University of California, Berkeley
U.C. Berkeley Division of Biostatistics Working Paper Series

Year 2004 *Paper* 162

Deletion/Substitution/Addition Algorithm for Partitioning the Covariate Space in Prediction

Annette Molinaro* Mark J. van der Laan†

*National Cancer Institute, National Institutes of Health, annette.molinaro@yale.edu

†Division of Biostatistics, School of Public Health, University of California, Berkeley, laan@berkeley.edu

http://biostats.bepress.com/ucbbiostat/paper162

Deletion/Substitution/Addition Algorithm for Partitioning the Covariate Space in Prediction

Annette Molinaro and Mark J. van der Laan

Abstract

We propose a new method for predicting censored (and non-censored) clinical outcomes from a highly-complex covariate space. Previously we suggested a unified strategy for predictor construction, selection, and performance assessment. Here we introduce a new algorithm which generates a piecewise constant estimation sieve of candidate predictors based on an intensive and comprehensive search over the entire covariate space. This algorithm allows us to elucidate interactions and correlation patterns in addition to main effects.

1 Introduction

By pinpointing and targeting specific early events in disease development clinicians aim toward a more preventative model of attacking cancer. These early events can be measured as genomic, epidemiologic, and/or clinical variables. Genomic variables are measured on expression or Comparative Genomic Hybridization (CGH) microarrays, epidemiologic variables with questionnaires, and clinical variables with pathology and histology reports. These measurements are then used to predict clinical outcomes such as primary occurrence, recurrence, metastases, or death.

To formalize the prediction of clinical outcomes (possibly censored) with high-dimensional covariate structures encompassing the aforementioned variables, a unified loss-based approach was suggested in Molinaro et al. (2004). This approach entails generating a sieve of candidate predictors, selecting a 'best' predictor from the sieve, and assessing its performance. In this 'roadmap' the user has two decisions to make. First, the investigator must choose which loss-function to implement (e.g., the squared error loss function). The choice of loss-function is specific to the parameter of interest, for example the conditional mean of the outcome given the covariates has as a corresponding loss function the L_2, or squared error. The second decision is how to generate an increasingly complex sieve (e.g., in tree estimation the sieve is obtained by recursive binary partitioning). The choice of how to generate this sieve is specific to the parameter of interest, the collected covariates, *and* the question the investigator is seeking to answer. For instance, if the outcome is continuous, the covariates include categorical and continuous variables, and the question is predicted time to outcome, the parameter of interest is the conditional mean of the outcome given the covariates. Thus, the user could select linear regression or recursive partitioning to generate the sieve.

In the previous manuscript (Molinaro et al., 2004) we were interested in exploring the individual contribution of various covariates as well as their interactions for the purposes of predicting survival outcomes. As such, we chose Classification and Regression Trees (CART) (Breiman et al., 1984), a binary recursive partitioning algorithm, for generating the sieve. The end product of CART is a list of *'and'* statements. For example, a CART tree is illustrated in Figure 1. In this tree there are two variables (diagnosis age, a continuous variable, and tumor grade, an ordered variable) and the outcome is time to recurrence, a continuous variable. Thus, the parameter of interest

is the conditional mean of the time to recurrence given the covariates and the chosen loss function is the squared error. The splitting and pruning (details can be found in Breiman et al. (1984)) are based on the loss function and result in a tree with three terminal nodes. The final predictor can be read as: if a patient is less than 50 her predicted time to recurrence is β_1; if a patient is over 50 *and* her tumor grade is less than or equal to 2 her predicted time to recurrence is β_2; if a patient is over 50 *and* her tumor grade is greater than 2 her predicted time to recurrence is β_3.

Although the importance of such statements is not in debate, there are settings where the *'and'* statement ordering does not accurately account for all of the observed biological phenomena. The motivating example for this is during the cell cycle there may be several regions of DNA which have been altered leading to a gain on one chromosome and a loss on another. Although these two mutations may be mutually exclusive the end result could be similar. In this situation, we would want to account for *'or'* orderings, e.g., 'loss at locus 1 *'or'* loss at locus 3 *'and'* gain at locus 2 predicts survival of y months', in addition to a list of *'and'* statements.

In order to accomplish this task of aggressively searching a highly-complex covariate space (e.g., thousands of BAC measurements in a array CGH study) we propose a new algorithm which builds *'and'* and *'or'* statements. We suggest that this will supersede previous approaches by being not only more aggressive but also more flexible. For example, CART utilizes a limited set of moves amounting to forward selection (node splitting) followed by backward elimination (tree pruning). In contrast, our proposed algorithm will not only split regions (nodes in tree estimation) it will also combine and substitute regions. These additional moves, more generally introduced in Sinisi and van der Laan (2004), will allow us to unearth intricate correlation patterns and further elucidate interactions in addition to main effects. As such, in this chapter we present an algorithm for generating a piecewise constant estimation sieve of candidate estimators based on an intensive and comprehensive search over the entire covariate space.

1.1 Motivation

The motivation for this algorithm came from the desire to predict clinical outcomes (possibly censored) with high-dimensional covariate structures (Figure 2). These structures can include covariates from CGH or expression microarrays in addition to histology, epidemiology and pathology variables.

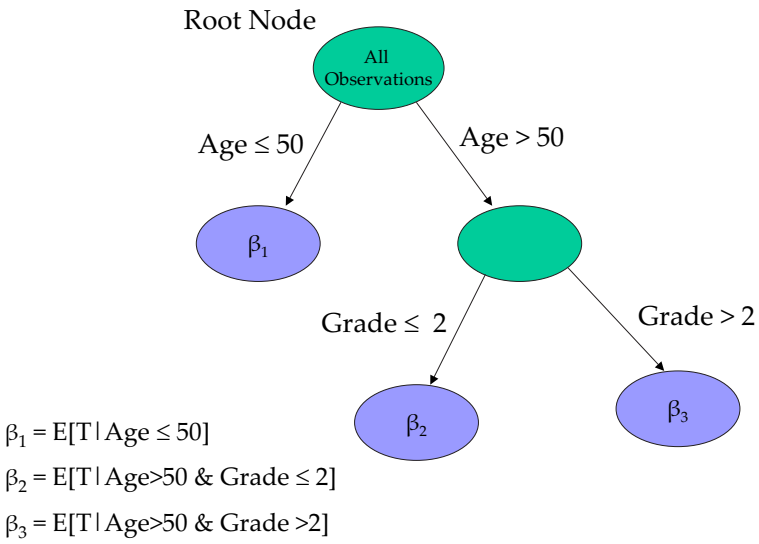

Figure 1: *Classification and Regression Tree Example.* This tree shows an example of recursive binary partitioning using CART with the variables 'Diagnosis Age' and 'Tumor Grade'. The terminal nodes, in blue, have predicted values $\beta_1, \beta_2,$ and, β_3.

It was the focus on array CGH that prompted the need for a new approach. The reason being that there could be several mutually exclusive chromosomal abberations occurring simultaneously that result in the same subsequent genomic alteration and/or clinical outcome. Binary recursive partitioning as described in Breiman et al. (1984) and Molinaro et al. (2004), results in a list of *'and'* statements which do not accommodate this biological circumstance. A better way to address this problem is with *'or'* in addition to *'and'* statements. In the statistical literature, Logic Regression has been introduced as an algorithm which attempts to construct predictors as Boolean combinations of binary covariates (Ruczinski et al., 2003; Kooperberg et al., 2001). Unfortunately, logic regression does not work in this setting because BACs are continuous, not binary, variables. Although a predetermined split for a 'gain' or 'loss' could be used to dichotomize the BACs, an interest in piecewise constant algorithms like CART is the ability to find a 'best' split along a continuous variables, not introduce said variables as binary.

1.2 Outline

In Section 2 we outline a unified loss based approach for generating candidate predictors, selecting a 'best' predictor, and assessing its performance. In this approach the user has two fundamental decisions: which loss function to use (Step 1) and which method or algorithm for generating an increasingly complex sieve of candidate estimators (Step 2). In Sections 2.1-2.4 the individual elements of the roadmap are detailed in the context of piecewise constant regression.

Section 2.2.2 reviews the mapping from the full data world to the observed data world which conveniently extends the new algorithm from the full, or uncensored, to the observed, or censored, data. Parameters of interest for univariate prediction and their corresponding loss functions are also explored. The reader is directed to Molinaro et al. (2004) for a description of the parameters of interest and corresponding loss functions for multivariate prediction and density estimation.

The algorithm is formally introduced in Section 3, including the main components of moves, ordering, and risk functions. The numerous simulations to evaluate the performance of the proposed algorithm as compared to CART and hybrids of the algorithm are shown in Section 4 and the conclusion in Section 5.

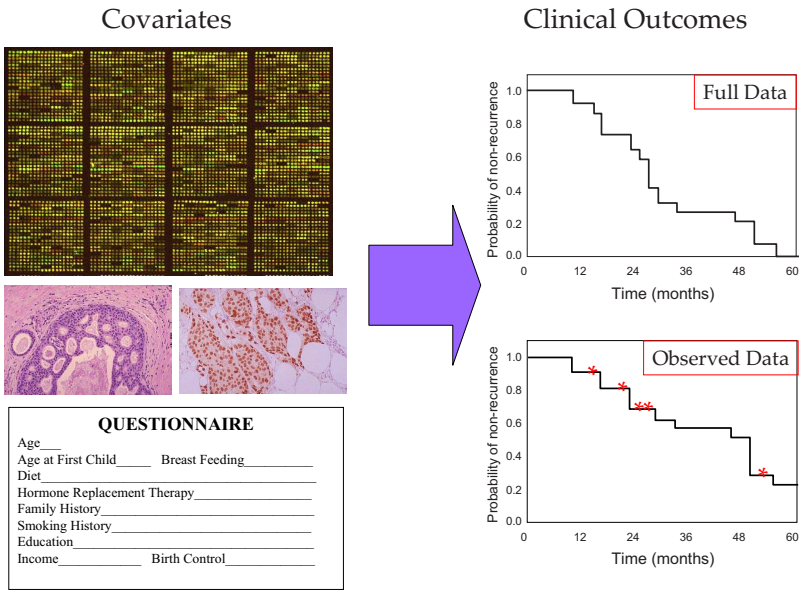

Figure 2: *Link observed covariates to clinical outcome of interest (Section 1.1).* The observed genomic, histopathologic, and epidemiologic covariates are explored for main effects, interactions, and underlying correlations in order to predict the uncensored (Full Data) or censored (Observed Data) clinical outcome of interest.

2 Estimation road map for censored data

Our proposed strategy for estimator construction, selection, and performance assessment is entirely driven by the choice of a *loss function* for the full, uncensored data structure. Censored data can be accommodated by mapping the full data loss function into an observed data loss function with the same expectation. The suggested strategy is outlined in these three steps:

1. *Definition of the parameter of interest in terms of a loss function for the observed data.* For the full data structure, define the parameter of interest as the minimizer of the expected loss, or *risk*, for a loss function chosen to represent the desired measure of performance (e.g., squared error loss for a population mean). Apply the general estimating function methodology of van der Laan and Robins (2002) to map the *full, uncensored* data loss function into an *observed, censored* data loss function having the same expected value and leading to an efficient estimator of this risk.

2. *Construction of candidate estimators based on a loss function for the observed data.* Define a finite collection of candidate estimators for the parameter of interest based on a sieve of increasing dimension approximating the complete parameter space. For each element of the sieve, the candidate estimator is defined as the minimizer of the empirical risk based on the observed data loss function (e.g., within-node sample mean for the squared error loss).

3. *Cross-validation for estimator selection and performance assessment based on a loss function for the observed data.* Use cross-validation to estimate risk based on the observed data loss function and to select an optimal estimator among the candidates in Step 2. This step relies on the unified cross-validation methodology of van der Laan and Dudoit (2003) and their finite sample and asymptotic optimality results for cross-validation estimator selection for general data generating distributions, loss functions (possibly depending on a nuisance parameter), and estimators.

2.1 Models

This section elaborates on the main steps of the *estimation roadmap for censored data* (Section 2) in the context of piecewise constant estimation. As such, the candidate estimators in Step 2 of the road map are generated by partitioning a suitably defined covariate space into disjoint and exhaustive regions. As illustrated in Molinaro et al. (2004), univariate prediction, multivariate prediction, and density estimation can be handled by specifying a suitable full data loss function for each of these problems. In this chapter we will focus solely on univariate prediction and refer the reader to the previous chapter for multivariate prediction and density estimation loss-functions and implementation.

2.1.1 Full data structure

In the *full data world*, we observe n i.i.d. observations $X_1, \ldots, X_n$, of a full data structure $X = (T, W)$. Let T denote the random survival time and $Z = \log(T)$. The baseline covariates, including the genomic, epidemiologic and histologic measurements are included in W. Denote the distribution of the full data structure X by F_X. Although the covariate process may contain both time-dependent and time-independent covariates we will focus on the time-independent W.

2.1.2 Observed data structure

In the observed data world, we rarely see all of the relevant variables in the full $X = (T, W)$. Rather, we observe the minimum, $\tilde{T} = \min(T, C)$, of the survival time T and a univariate censoring variable C. This missing, or *censored*, survival data situation can be due to drop out or the end of follow-up. The *observed data structure* can be written as $O = (\tilde{T} = \min(T, C), \Delta = I(T \leq C), X))$ and the censoring process as $A(t) = I(C < t)$. By convention, if T occurs prior to C, we set $C = \infty$; thus, C is always observed and we can now denote the observed data structure as $O = (C, X)$. The random variable O has a distribution $P_0 = P_{F_X,G}$, indexed by the full data distribution, F_X, and the conditional distribution, $G(\cdot|X)$, of the censoring variable C given X. Due to the fact that what we observe about X is determined by C, $G(\cdot|X)$ is referred to as the *censoring* or *coarsening mechanism*. The *survivor function* for the censoring mechanism is denoted by $\bar{G}(c \mid X) = Pr(C \geq c \mid X)$.

We assume that for $c < T$, the Lebesgue hazard corresponding to the censoring mechanism given the full data X is:

$$\lambda_C(c|X) = Pr(C = c \mid C \geq c, X) = m(c, \bar{X}(c)),$$

for some measurable function, m. This assumption on the censoring mechanism, referred to as *coarsening at random* (CAR), holds if the censoring distribution only depends on the observed data O. As X does not include time-dependent covariates, then, under CAR, the censoring time C is conditionally independent of the survival time T given baseline covariates W. An important consequence of CAR is that it implies the following factorization for the density of the observed data $O = (C, X)$ (with respect to a dominating measure satisfying CAR itself), into an F_X-part and a G-part,

$$p_{F_X,G}(o) = p_{F_X}(o)h(o),$$

where $h(o)$ is the density $g_{C|X}(c \mid x)$ and $p_{F_X}(o) = f_{F_X}(\bar{X}(t)) \mid_{t=c}$ only depends on the measure F_X. Denote by $\mathcal{G}(CAR)$ the set of all conditional distributions G of C given X satisfying CAR. Gill et al. (1997), van der Laan and Robins (2002) (Section 1.2.3, in particular), and Robins and Rotnitzky (1992) provide further, thorough explanations of CAR.

2.2 Loss Functions for Univariate Prediction

2.2.1 Full Data Loss Function

In the full data world where we observe $X_1, \ldots, X_n$, the parameter of interest, ψ_0, is a mapping $\psi : \mathcal{S} \to I\!R$, from a covariate space $\mathcal{S}$ into the real line $I\!R$. Denote the parameter space by Ψ. The parameter ψ_0 is defined in terms of a *loss function*, $L(X, \psi)$, as the minimizer of the expected loss, or *risk*. That is, ψ_0 is such that

$$\begin{aligned} E_{F_X}[L(X, \psi_0)] &= \int L(x, \psi_0) dF_X(x) \\ &\equiv \min_{\psi \in \Psi} \int L(x, \psi) dF_X(x) = \min_{\psi \in \Psi} E_{F_X}[L(X, \psi)]. \end{aligned}$$

The purpose of the loss function L is to quantify performance. Thus, depending on the parameter of interest, there could be numerous loss functions from which to choose. Frequently the parameter of interest is the conditional

mean $\psi_0(W) = E[Z \mid W]$ which has the corresponding squared error loss function, $L(X, \psi) = (Z - \psi(W))^2$. Another common parameter of interest is the conditional median $\psi_0(W) = Med[Z \mid W]$ which has the corresponding absolute error loss function, $L(X, \psi) =\mid Z - \psi(W) \mid$.

2.2.2 Observed data loss function

In the observed data world, we have a learning set of n i.i.d. observations, $O_1, \ldots, O_n$, from the right-censored data structure, $O_i \sim P_0 = P_{F_X,G}$. The empirical distribution of $O_1, \ldots, O_n$ is denoted by P_n. The goal remains to find an estimator for a parameter ψ_0 defined in terms of the risk for a full data loss function $L(X, \psi)$, e.g., a predictor of the log survival time Z based on covariates W. An immediate problem is that a loss function such as the quadratic loss, $L(X, \psi) = (Z - \psi(W))^2$, cannot be evaluated for an observation O with censored survival time ($\Delta = 0$). Risk estimators based on only uncensored observations, such as $\frac{1}{n}\sum_i L(X_i, \psi)\Delta_i$, are biased for $E_0[L(X, \psi)]$ and, in particular, estimate the quantity $E_0[L(X, \psi)\bar{G}_0(T|X)]$ which is not minimized by the parameter of interest ψ_0.

The general solution is to replace the full (uncensored) data loss function by an observed (censored) data loss function with the same expected value, i.e., risk. The *general estimating function methodology* of van der Laan and Robins (2002) can be used to link the observed data world to the full data world. Specifically, the methodology allows full data estimating functions, $D(X)$, to be mapped into observed data estimating functions, $IC(O \mid Q, G, D)$, indexed by *nuisance parameter* G and, possibly, $Q = Q(F_X)$. IC denotes *influence curve* as abbreviated in van der Laan and Robins (2002). The estimating functions satisfy

$$E_{P_0}[IC(O \mid Q, G, D)] = E_{F_X}[D(X)], \qquad \text{if } G = G_0 \text{ or } Q = Q_0 = Q(F_X).$$

In our specific application, the full data estimating function is the loss function, $D(X) = L(X, \psi)$, and the risk for a given estimator ψ is viewed as the full data parameter of interest, $\theta_0 = E_0[D(X)] = E_0[L(X, \psi)]$. Observed data loss functions are obtained from the estimating functions IC, that is, $L(O, \psi \mid \eta_0) = IC(O \mid Q_0, G_0, L(\cdot, \psi))$ is an observed data loss function with the same risk as the full data loss function $L(\cdot, \psi)$, where η_0 denotes the nuisance parameters (Q_0, G_0)

$$\int L(o, \psi \mid \eta_0) dP_0(o) = \int L(x, \psi) dF_{X,0}(x).$$

Inverse probability of censoring weighted loss function. The *inverse probability of censoring weighted* (IPCW) estimating function was introduced by Robins and Rotnitzky (1992). Its name derives from the fact that the full data function $D(X)$ is weighted by the inverse of a censoring probability. This estimating function is written as:

$$IC(O \mid G, D) = D(X)\frac{\Delta}{\bar{G}(T|X)}, \tag{1}$$

where $\bar{G}$ is a conditional survival function for C given X and $\Delta = \mathrm{I}(T \leq C)$ is the censoring indicator. Given that

$$E_0[\Delta|X] = Pr_0(C \geq T|X) = \bar{G}_0(T|X) > 0, \qquad F_{X,0}\text{-a.e.},$$

one has

$$E_0\left[\frac{D(X)\Delta}{\bar{G}_0(T|X)}\right] = E_0\left[E_0\left[\frac{D(X)\Delta}{\bar{G}_0(T|X)} \mid X\right]\right] = E_0\left[\frac{D(X)}{\bar{G}_0(T|X)}E_0[\Delta|X]\right] = E_0\left[D(X)\right].$$

This suggests the IPCW observed data loss function, $L(O, \psi \mid \eta_0) = IC(O \mid G_0, L(\cdot, \psi))$, with nuisance parameter $\eta_0 = G_0$. The corresponding risk estimator is the empirical mean

$$\hat{\theta}_n = \frac{1}{n}\sum_{i=1}^{n} L(O_i, \psi \mid \eta_n) = \frac{1}{n}\sum_{i=1}^{n} L(X_i, \psi)\frac{\Delta_i}{\bar{G}_n(T_i|X_i)},$$

where η_n represents $\bar{G}_n$, an estimator of the nuisance parameter $\bar{G}_0$ derived under the CAR assumption for the censoring mechanism, i.e., by considering censoring mechanisms $G \in \mathcal{G}(CAR)$. For such models, the estimator $\bar{G}_n(T|X)$ is a function of the observed data $O = (C, X)$ and thus the resulting risk estimator $\hat{\theta}_n$ depends only on the observed data structure, $O_1, \ldots, O_n$. Conditions for the IPCW estimating function to provide a consistent risk estimator are that $\bar{G}_0(T|X) > \delta > 0$, $F_{X,0}$-a.e., for some $\delta > 0$, and that $\bar{G}_n$ is a consistent estimator for $\bar{G}_0$. Given α^w as the endpoint of support of $F_{T|W}(\cdot \mid W)$, the first condition holds if $\bar{G}(\alpha^w \mid W) > 0$, F_W-a.e.. The plausibility of this assumption can be verified with standard diagnostic tools. Violation of this assumption will lead to a biased estimator.

If a Cox proportional hazards model is assumed for the censoring mechanism G, then $\lambda_C(t \mid X) = \lambda_0(t)\exp(\beta^T J(t))$, where $J(t) = f(L(t))$ is a set of covariates extracted from the process $\bar{L}(t) = \{L(s) : 0 \leq s \leq t\}$ for some

given $\mathbb{R}^k$-valued function f. Standard software can then be employed to obtain maximum (partial) likelihood estimators of the baseline hazard function λ_0 and the regression coefficients β (e.g., `coxph` function in R).

An alternative to the IPCW estimating function is the *doubly robust inverse probability of censoring weighted* (DR-IPCW). Under an identifiability condition, the DR-IPCW ensures consistency if at least one of two nuisance parameters are correctly specified and asymptotic efficiency if both nuisance parameters are consistently estimated. The DR-IPCW is illustrated for this application in Molinaro et al. (2004) and explicitly in van der Laan and Robins (2002).

In the absence of censoring, i.e., when $\Delta \equiv 1$ and $C \equiv \infty$, both the IPCW and DR-IPCW observed data loss functions reduce to the full data loss function, $L(O, \psi \mid \eta_0) = L(X, \psi)$. This ensures that the censored and full data estimators coincide when there is no censoring. In addition, one can estimate the nuisance parameter $\bar{G}_0$ in the IPCW and DR-IPCW loss functions using other covariates than those for ψ, in order to allow for informative censoring and a gain in efficiency. Properties of the IPCW and DR-IPCW estimating functions are discussed in detail in van der Laan and Robins (2002).

2.3 Generating Candidate Estimators with Piecewise Constant Regression Models

For Step 1 of the *estimation roadmap* we defined the full and observed data structures (Section 2.1), the full and observed data loss functions (Section 2.2), and outlined the mapping from the full data loss function to the observed to accommodate censored data (Section 2.2.2). In Step 2 the goal is to generate a sieve of candidate estimators. As such, in this section we detail how to approximate the parameter space by a sequence of subspaces of increasing dimension and generate candidate estimators for each subspace all in the framework of piecewise constant regression models.

In the full data world, we defined $X = (T, W)$, where T is the random survival time and W are the baseline covariates. Consider W to be a d-vector of baseline covariates. Define a countable set of *basis functions*, $\{\phi_j : j \in \mathbb{N}\}$, indexed by the non-negative integers $\mathbb{N}$. These basis functions are simply set indicators $\{\mathcal{R}_j : j \in I\}$ which form a partition of the covariate space $\mathcal{S}$, where I is an index set, $I \in \mathcal{I}$, and $\mathcal{I}$ is a collection of subsets of $\mathbb{N}$.

Here $\mathcal{R}_j$ denotes regions of $\mathcal{S}$ which are disjoint ($\mathcal{R}_j \cap \mathcal{R}_{j'} = \emptyset$, $j \neq j'$) and exhaustive ($\mathcal{S} = \cup_{j \in I} \mathcal{R}_j$). Now every parameter $\psi \in \Psi$ can be written (and approximated) as a finite linear combination of the basis functions:

$$\psi_{I,\beta}(\cdot) \equiv \sum_{j \in I} \beta_j \phi_j(\cdot),$$

where for a given index set $I \in \mathcal{I}$, the coefficients $\beta = (\beta_1, \ldots, \beta_{|I|})$ belong to $B_I \equiv \{\beta : \psi_{I,\beta} \in \Psi\} \subseteq I\!R^{|I|}$. These are of the form referred to as *piecewise constant regression models* (Härdle, 1989).

The complete parameter space Ψ can be written as the collection of basis functions $\{\phi_j : j \in I\!N\}$ and represented by

$$\Psi \equiv \{\psi_{I,\beta}(\cdot) = \sum_{j \in I} \beta_j \phi_j(\cdot) : \beta,\ I \in \mathcal{I}\}.$$

Define a *sieve*, $\{\Psi_k\}$, of subspaces $\Psi_k \subset \Psi$, of increasing dimension approximating the complete parameter space Ψ, such as,

$$\Psi_k \equiv \left\{ \psi_{I,\beta}(\cdot) = \sum_{j \in I} \beta_j \phi_j(\cdot) : \beta,\ I,\ |I| \leq k \right\},$$

where k denotes the index set size (i.e., how many basis functions). Now for every k we want to find the estimator which minimizes the empirical risk over the subspace Ψ_k. That can be done by initially optimizing over the regression coefficients $\beta \in B_I$ for a given index set I and then optimizing over the index sets I. We will postpone the full discussion of the latter until Section 3.

2.3.1 Estimation of regression coefficients β for a given subset of basis functions

Given index sets $I \in \mathcal{I}$, define I-specific subspaces

$$\Psi_I \equiv \{\psi_I, \beta : \beta \in B_I\}.$$

For each subspace Ψ_I, the regression coefficients β are estimated by minimizing the empirical risk, i.e.,

$$\begin{aligned} \hat{\beta}_I = \beta_I(P_n) &\equiv \text{argmin}_{\beta \in B_I} \int L(o, \psi_{I,\beta} \mid \hat{\nu}_n) dP_n(o) \\ &= \text{argmin}_{\beta \in B_I} \sum_{i=1}^{n} L(O_i, \psi_{I,\beta} \mid \hat{\nu}_n), \end{aligned}$$

where $\hat{\nu}_n$ is an estimator of the nuisance parameter. It is possible to write the I-specific estimators as $\hat{\psi}_I = \Psi_I(\cdot|P_n) \equiv \psi_{I,\beta_{I(P_n)}}, I \in \mathcal{I}$. An example of this is with the squared error loss function, $\hat{psi}_I$ is then the least squares linear regression estimator corresponding with the variables identified by the index set I.

2.4 Cross-Validation for estimator selection and performance assessment

In Step 3 of the *estimation roadmap* cross-validation is introduced for two purposes: to select an optimal estimator among the candidates generated in Step 2 (Section 2.3) and to assess the performance of the resulting estimator. Both of these are based on the observed data loss function chosen in Step 1. For selecting the optimal estimator, an alternative to cross-validation would be to minimize the empirical or cross-validated risk as a measure of error across the entire parameter space. However, this estimate would be highly variable and too data-dependent. van der Laan and Dudoit (2003) derive finite sample and asymptotic optimality results for the cross-validation selector for general data generating distributions, loss functions (possibly depending on a nuisance parameter), and estimators. The implication of these results is that selection via cross-validation is adequate in thorough searches of large parameter spaces. Thus, we shall rely on cross-validation, specifically v-fold cross-validation, for choosing the 'best' estimator and assessing its performance.

3 New Algorithm for Generating Candidate Estimators

As an alternative to previous methods for generating piecewise constant estimates in Step 2 of the *estimation roadmap*, we describe a completely data adaptive, aggressive, and flexible algorithm to search the entire covariate space. The fundamental steps, or moves, of this algorithm were introduced in Sinisi and van der Laan (2004). Here we define the moves in the context of both histogram regression and the motivating example detailed in Section 1.1.

In piecewise constant regression the basis functions ϕ_j defined in Section

2.3 are indicator functions of regions $\mathcal{R}_j$ of the covariate space $\mathcal{S}$, where these regions are disjoint ($\mathcal{R}_j \cap \mathcal{R}_{j'} = \emptyset$, $j \neq j'$) and exhaustive ($\mathcal{S} = \cup_{j \in I} \mathcal{R}_j$). We further define subsets of these regions as S_l. Thus, each region can be a union of several subsets or smaller regions of the covariate space. The subsets S_l themselves are disjoint ($S_l \cap S_{l'} = \emptyset$, $l \neq l'$) and exhaustive ($\mathcal{S} = \cup_{l \in I} S_l$). The predicted value for each of the regions is constant resulting in a histogram estimate of the regression surface.

The new partitioning algorithm utilizes three moves, or step functions, to generate index sets (i.e., different partitionings of the covariate space) with the goal of minimizing a risk function over all the generated index sets. These three moves are described in the following section.

3.1 Algorithm Moves

- **Deletion** A deletion move forms a union of two regions of the covariate space regardless of their spatial location, i.e., the two regions may not be contiguous. Formally, given a particular partitioning, i.e., an index set $I \in \mathcal{I}$, which consists of k basis functions representing indicator functions of regions, such that $|I| = k$, we define the set $DEL(I) \subset \mathcal{I}$ as that which contains all possible unions of two disparate regions. This new set, $DEL(I)$, is of size C_2^k.

- **Substitution** A substitution move divides two disparate regions into two subsets each and then forms combinations of the four subsets resulting in two new regions. Thus, this step forms unions of regions (or subsets within the regions) as well as divides regions. The motivation for this was to separate gains from losses in CGH data. The possible subsets of two regions and combinations thereof can be seen in Figure 3. Formally, given a particular partitioning, i.e., an index set $I \in \mathcal{I}$, which consists of k basis functions representing indicator functions of regions, such that $|I| = k$, we define the set $SUB(I) \subset \mathcal{I}$ by splitting all regions into two subsets each and subsequently form all unique combinations of the $2k$ subsets. This new set, $SUB(I)$, is of size $6 * C_2^k$, due to the six unique combinations for every two regions (i.e., four subsets) (see Figure 3).

- **Addition** An addition move splits one region into two distinct regions. Formally, given a particular partitioning, i.e., an index set $I \in \mathcal{I}$, which

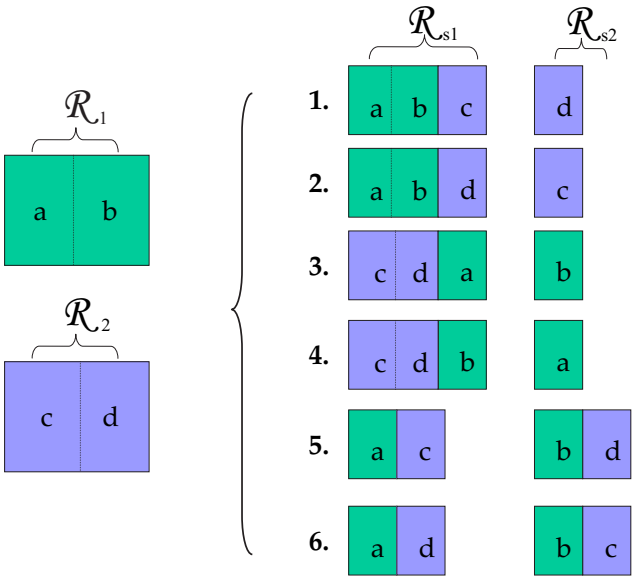

Figure 3: *Possible Substitutions Moves for two disparate regions.* The 'best' split on Region 1 ($\mathcal{R}_1$) is found and labeled a and b. The 'best' split on Region 2 ($\mathcal{R}_2$) is found and labeled c and d. All (six) unique combinations of $a, b, c,$ and d are formed.

consists of k basis functions representing indicator functions of regions, such that $|I| = k$, we define the set $ADD(I) \subset \mathcal{I}$ as that which contains two basis functions for every initial basis function. The new basis functions represent the best split of the original region into two distinct regions. As such $ADD(I)$ is of size $2k$.

'Best split' as referred to in both the Substitution and Addition Steps is the same notion as that used in CART. The best split of a region is the split which most decreases the residual sum of squares for the whole space. However, since all splits are examined individually it is simply the split which minimizes the within node (i.e., region) sum of squares.

A unique and highly important contribution of this algorithm is through the Deletion and Substitution Steps. It is during these steps that unions of regions (or subsets) are formed. These unions of disparate regions result in *'or'* statements. After these steps, the resulting basis functions may be comprised of numerous *'and'* and *'or'* statements. For computational

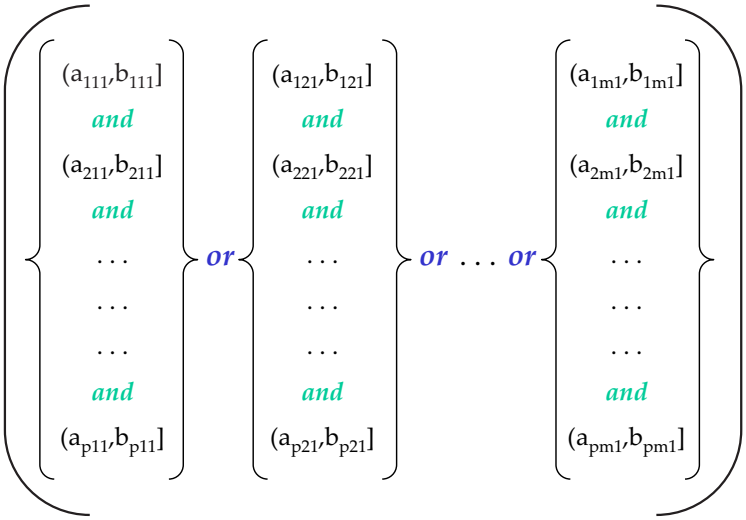

Figure 4: *A Region stored in a matrix.* The rows of the matrix represent the p covariates and the columns represent the m subsets of the kth region. The interval of the covariate within the corresponding subset is represented by $(a_{p,m,k}, b_{p,m,k}]$.

ease the regions are stored as individual matrices. The rows of each matrix represent individual variables (e.g., BACs) and the columns represent the subsets within the corresponding region. For example, a matrix representing one region can be seen in Figure 4. This representation allows p covariates (as rows) and m subsets (as columns) for the kth region. The columns are connected via unions, or *'or'* statements and the rows via *'and'* statements. Each variable (here assumed to be continuous) has intervals represented by $(a_{p,m,k}, b_{p,m,k}]$, which are indexed by the covariate (row), subset (column), and region (matrix).

3.2 Risk Functions

By utilizing the moves Deletion, Substitution, and Addition, the algorithm generates index sets with the goal of minimizing a risk function over all the generated index sets. There are two potential risk functions: the empirical

risk and the cross-validated empirical risk. The search resulting from the two risk functions is different as well as a slight modification to how the algorithm is run. As such, each is explained here, their effect on running the algorithm in the following section, and the resulting effect shown via simulations in Section 4.

As defined in Section 2.3.1, the I-specific estimators are represented $\hat{\psi}_I = \Psi_I(\cdot|P_n) \equiv \psi_{I,\beta_{I(P_n)}}, I \in \mathcal{I}$. In this context of histogram regression with the squared error loss function, $\hat{\psi}_I$ is the least squares linear regression estimator corresponding with the variables identified by the index set I.

For a particular partitioning, or index set, $I \in \mathcal{I}$, the *empirical risk* of the I-specific estimator is

$$\begin{aligned} I \to f_1(I) &\equiv \int \text{Ł}(o, \hat{\psi}_I \mid \hat{\nu}_n) dP_n(o) \\ &= \frac{1}{n} \sum_{i=1}^{n} L(O_i, \hat{\psi}_I | \hat{\nu}_n), \end{aligned}$$

where both $\hat{\psi}_I = \Psi_I(\cdot|P_n)$ and the estimator for the nuisance parameters $\hat{\nu}_n$ are estimators based on the empirical distribution P_n. With the empirical risk function, the algorithm searches to minimize it over all index sets I of size less equal to k, where $k = 1, \ldots, K$. For each k there is a best partitioning which can be denoted:

$$I_k^*(P_n) \equiv \text{argmin}_{\{I:|I|=k, I\in\mathcal{I}\}} f_1(I).$$

The algorithm searches for an approximation of $I_k^*(P_n)$ which is designated as $I_k(P_n)$ and the resulting estimator as $\hat{\psi}_k = \Psi_k(P_n)$. This results in a sieve of increasingly complex estimators indexed by k. To select the best partitioning, i.e., index set, v-fold cross-validation is implemented such that,

$$k(P_n) \equiv E_{S_n} \int \text{Ł}(o, \hat{\Psi}_k(P^0_{n,S_n}) | \hat{\nu}_{n,S_n^0}) dP^1_{n,S_n}(o),$$

where $\hat{\Psi}_k(P^0_{n,S_n})$ and $\hat{\nu}_{n,S_n^0}$ are based on the empirical distributions for the training sets.

A second option is the *cross-validated empirical risk function.* For a particular partitioning, or index set, $I \in \mathcal{I}$, the cross-validated empirical risk of

the I-specific estimator is

$$I \rightarrow f_2(I) \equiv E_{S_n} \int Ł(o, \hat{\psi}_I(o|P^0_{n,S_n})|\hat{\nu}_n) dP^1_{n,S_n}(o)$$

where $\hat{\Psi}_k(P^0_{n,S_n})$ and $\hat{\nu}_{n,S^0_n}$ are based on the empirical distributions for the training sets. With the cross-validated empirical risk function, the algorithm searches to minimize it over all index sets I. This does not result in a sieve as the previous risk function, but a single estimator.

3.3 Algorithm Ordering

Having outlined the three moves, Deletion, Substitution, and Addition, in Section 3.1 and the risk functions in Section 3.2, the ordering is the final step of the algorithm. The is will be explained with the empirical risk as the risk function and then at the end of this section the modification for the cross-validated risk estimate will be reviewed.

Minimizing the empirical risk function results in a sieve of estimators indexed by k. The vector $BEST(k)$ will be used to store the estimated empirical risk corresponding with the best partitioning of size k. Given the goal of minimizing $I \rightarrow f_1(I)$, there are three parts to this process:

1. **Initiate Algorithm.** The algorithm is initiated by setting the running partitioning, I_0, to the null set. For piecewise constant regression, the null set is that set which includes the entire covariate space as one region. Then $f_1(I_0)$ is evaluated and $BEST(1)$ is given its value. A stopping value indicating the maximum number of basis functions to consider is assigned as M.

2. **Move through Step Functions.**

   - * Set $k = |I_0|$. If $k > 1$ find an optimal updated I^- of size $k-1$ among all allowed **deletion** moves, where $I^- \equiv \text{argmin}_{I \in DEL(I_0)} f_1(I)$. If $f(I^-) < BEST(k-1)$ then $BEST(k-1) = f(I^-)$, $I_0 = I^-$ and return to *.
   - Else, find an optimal updated $I^=$ of size k among all allowed **substitution** moves, where $I^= \equiv \text{argmin}_{I \in DEL(I_0)} f_1(I)$. If $f(I^=) < BEST(k)$ then $BEST(k) = f(I^=)$, $I_0 = I^=$ and return to *.

- Else, find an optimal updated I^+ of size $k+1$ among all allowed **addition** moves, where $I^+ \equiv \text{argmin}_{I \in DEL(I_0)} f_1(I)$. If $f(I^+) < BEST(k+1)$ then $BEST(k+1) = f(I^+)$, $I_0 = I^+$ and return to *.

3. **Stop Algorithm.** If $|I| = M$ stop the algorithm.

When the algorithm is stopped there is a list of best estimators $I_k(P_n)$, where $k = 1, \ldots, M$. As detailed in Section 3.2, cross-validation is used to select the best k. The entire process is depicted in a flow chart in Figure 5. An example in two dimensions (i.e., with two BACs as covariates) is shown in Figures 6 and 7. The algorithm is initiated in the first graph of Figure 6 with all observations in one region. The second graph in the same figure shows that the 'best split' was found along BAC1 and now the patients with a 'loss' on BAC1 have a higher predicted time to recurrence (z-axis) than those with a 'gain' on BAC1. The algorithm continues splitting (Addition Step) as seen in the first graph in Figure 7. In the second graph of the same figure a Deletion Step is apparent as now the partitioning reads:

> If a loss on BAC1 **and** a loss on BAC2 the patient has the highest (longest) time to recurrence. If a gain on BAC1 **and** a gain on BAC2 the patient has the lowest (shortest) time to recurrence. If a loss on BAC1 **and** a gain on BAC2 **OR** a loss on BAC2 **and** a gain on BAC1 the patient has intermediate time to recurrence.

Thus far, the ordering of the algorithm has been with the empirical risk function $f_1(I)$, if the cross-validated empirical risk function $f_2(I)$ is used instead, $f_1(I)$ is replaced by $f_2(I)$ above and the vector $BEST(\cdot)$ is removed. At each step function the cross-validated empirical risk function is assessed and compared to that of I_0's. If at the conclusion of the Addition step $f_2(I^+)$ is not smaller than $f_2(I_0)$ the algorithm stops. At this point there is one estimator.

An additional way to limit the number of basis functions for either risk function is to restrict the minimum number of observation in a region $\mathcal{R}_j$ or subset S_l such that no more splits can occur if that minimum is reached.

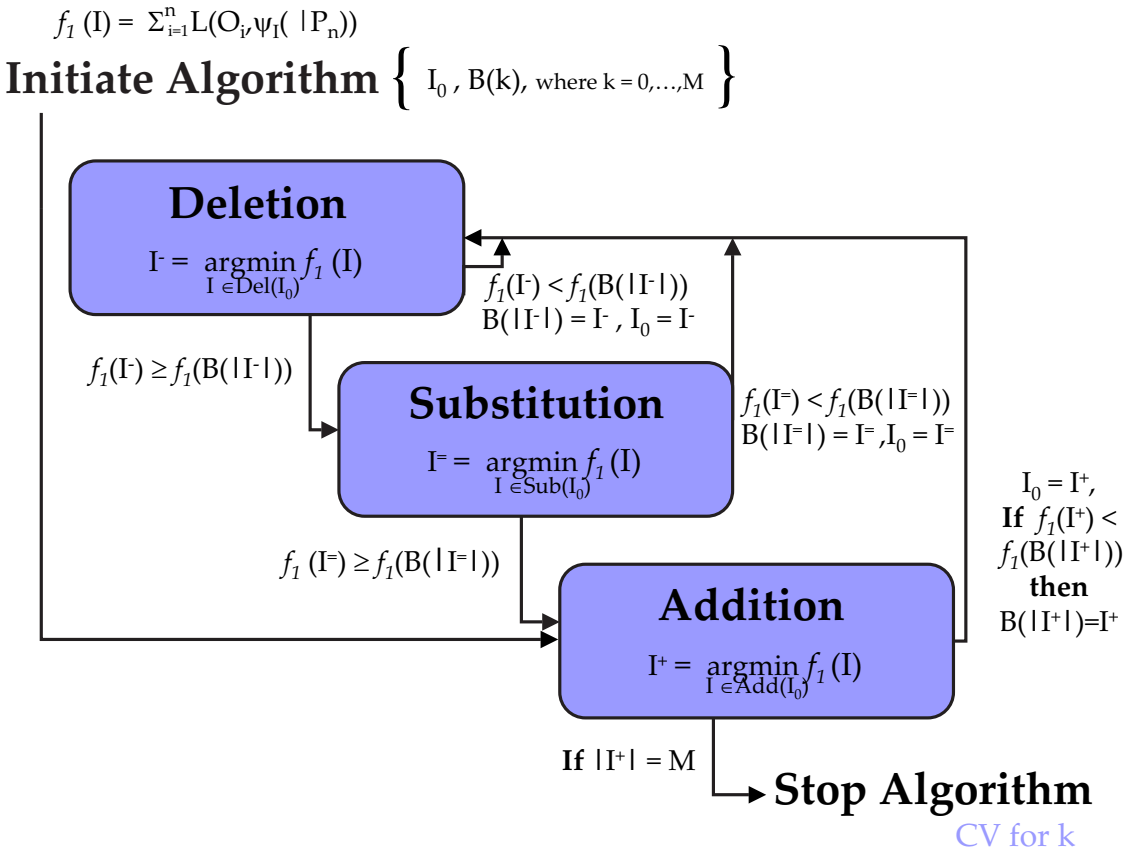

Figure 5: *Flowchart of Algorithm.* This flowchart shows the ordering of the algorithm with the empirical risk function $f_1(I)$.

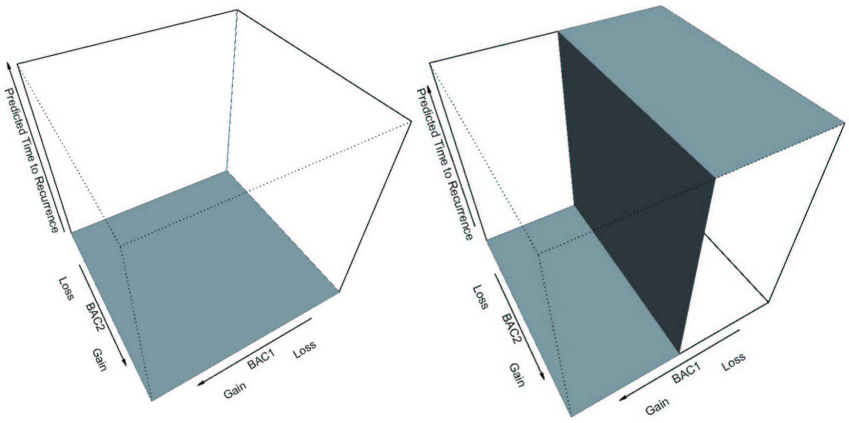

Figure 6: *Two dimensional display of Addition Step.* The algorithm is initiated on the left and the first 'best' split is found separating a 'loss' on BAC1 from a 'gain'.

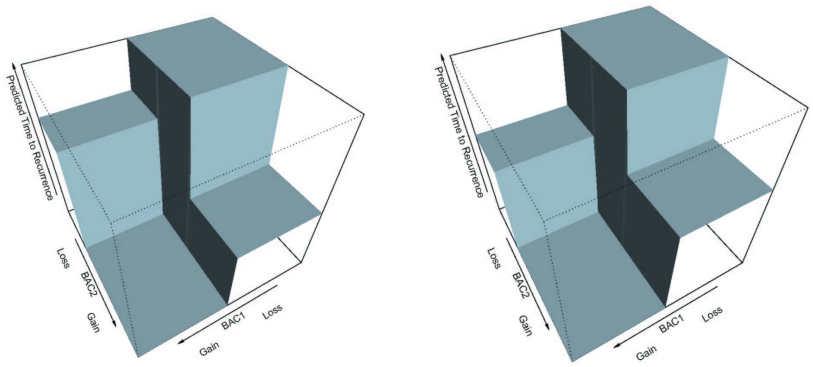

Figure 7: *Two dimensional display of Deletion Step.* Two disparate regions on the left form a union on the right. This union reads: "If a loss on BAC1 **and** a gain on BAC2 **OR** a loss on BAC2 **and** a gain on BAC1 the patient has intermediate time to recurrence."

4 Univariate Simulations

To understand the flexibility and aggressive nature of the proposed algorithm numerous stimulations studies were initiated. This section will relay the results from the simulations involving one covariate, hence the name 'Univariate Simulations'. Many of the results are shown in the following tables. For each of the simulations a description accompanies the table.

For each simulation several sample sizes, v-folds of cross-validation (e.g., 2-fold and 5-fold), risk functions, and underlying distributions (e.g., uniform) were utilized to get truly explore the potential of the algorithm. It is our belief that this algorithm will fare much better when there are underlying symmetric distributions as such the simulations involve both normal and uniform distributions.

4.1 Simulation Study 1

The first simulation study was designed implementing the cross-validated empirical risk function (CV-RISK). In order to evaluate this attempt at partitioning the covariate space for the purposes of prediction samples sizes of $250, 500,$ and 1000 were used as well as different underlying distributions. To explore difference in choices of v in v-fold cross-validation we looked at $v \in \{2, 5, 10\}$. In Table 1, the full data distribution was simulated from $y = x^2 + er$, where $x \sim N(0, 1)$ and $er \sim N(0, .0625)$. The small error was chosen to approximate the accuracy asymptotically. For each of 100 repetitions a training set was used to build a predictor with the CV-RISK and this partitioning with an independent test sample of 1000. The mean of the risk for each of the 100 repetitions is reported in the table along with the variance of the risk. In addition to 'our' algorithm, the same training set was partitioned with `rpart`, the implementation of CART in `R` (Therneau and Atkinson, 1997; Ihaka and Gentlemen, 1996). The choice of which is the 'best' tree as discussed in Breiman et al. (1984) and Therneau and Atkinson (1997) is to use the $1 - SE$ rule. Another option is to choose the subtree which simply minimizes the cross-validated risk. In the following tables, the first is labeled as 'rpart' and the second as 'rpart0'.

In this simulation, the minimum number of observations in a node or a region was set to 7 (the default for `rpart`). In the next section, different minimums for regions and nodes are compared.

As a way to measure how well the proposed algorithm is doing in com-

parison to `rpart`, we calculated an efficiency ratio ('ratio 2' in the tables). This ratio relays a sense of how fast our algorithm is approaching the truth as compared to `rpart`. For example, if the ratio is one then the two methods are doing equally well. However, if the ratio is less than one our method is getting to the truth faster.

In Table 1, we see that we are approaching the truth faster in almost all situations except with 2-fold cross-validation in sample sizes 500 and 1000. In addition to the mean of the risk being smaller for our algorithm, of importance is that it is using less basis functions than `rpart` is nodes. This means that the algorithm is coming up with the sparsest way of describing the data. By the time the sample size reaches 1000 that description is in half of the parameters that `rpart` is using (i.e., 12.06 basis functions to 28.72 nodes for 5-fold CV). Interestingly, there seems to be quite a gain from simply minimizing the CV risk as opposed to using the $1 - SE$ rule for `rpart`.

In Table 2, the exact same simulation is shown except for the fact that the underlying simulation is now uniform (i.e., $U(0, 1)$). In this simulation one can see that we are doing a small amount better than 'rpart0' up to $n = 1000$ and much better than 'rpart'. Because both methods are piecewise constant estimators there is no reason to think that asymptotically our algorithm will supersede CART with an underlying uniform distribution.

4.2 Simulation Study 2

The second simulation study was designed again implementing the cross-validated empirical risk function (CV-RISK). This simulation is set-up the same as Simulation 1 except for two additional comparisons. First, the variance of the underlying model. In Simulation 1 it was 0.0625, here there are two variances 0.5625 and 1. In Table 3, the full data distribution was simulated from $y = x^2 + er$, $x \sim NORM(0, 1)$ and $er \sim N(0, 0.5625)$. In Table 4, the full data distribution was simulated from $y = x^2 + er$, $x \sim N(0, 1)$ and $er \sim N(0, 1)$. In Table 5, the full data distribution was simulated from $y = x^2 + er$, $x \sim unif(0, 1)$ and $er \sim N(0, 0.5625)$. In Table 6, the full data distribution was simulated from $y = x^2 + er$, where $x \sim U(0, 1)$ and $er \sim N(0, 1)$.

The second and most substantial difference is that in addition to selecting over the best partitioning indexed by k the algorithm is also selecting over δ, the minimum number to include in a region (or node) before it can be split. There are four estimators in this simulation: 'ours(k, δ)' chooses over k and

Table 1: *Simulation study* **CV-RISK**. Full data simulated from $y = x^2 + er$, where $x \sim N(0,1)$ and $er \sim N(0, .0625)$. Candidate estimator was chosen over 100 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2) for both our algorithm and rpart (col 3). The results (cols 4-7) in table are based on an independent test sample of $n = 1000$. col 4 is the average of the 100 risks (with the L_2 loss function) for each method, col 5 is the variance of the risks over the 100 reps, col 6 is the average size (number of basis functions for ours and number of splits for rpart) and col 7 is the ratio of averaged risks (col 4) (ours/rparts).

Sample			100 Repetitions				
Size	$v-fold$	Method	mean	std dev	avg size	ratio	ratio2
250	2	ours	0.32188	0.14985	6.32	1	
		rpart	0.55843	0.18393	4.67	.576	.523
		rpart0	0.36851	0.09927	13.32	.873	.848
	5	ours	0.26125	0.09384	7.44	1	
		rpart	0.45305	0.14195	5.69	.577	.509
		rpart0	0.35172	0.09927	14.45	.743	.687
	10	ours	0.27828	0.14513	7.46	1	
		rpart	0.46529	0.15208	5.76	.598	.536
		rpart0	0.36282	0.11626	14.94	.767	.719
500	2	ours	0.24002	0.12891	8.02	1	
		rpart	0.31778	0.10918	7.31	.755	.695
		rpart0	0.21727	0.06906	20.66	1.105	1.147
	5	ours	0.18935	0.07318	9.95	1	
		rpart	0.27216	0.08574	9.55	.696	.606
		rpart0	0.22187	0.07544	21.26	.853	.796
	10	ours	0.18111	0.06535	10.18	1	
		rpart	0.25474	0.07448	10.26	.711	.617
		rpart0	0.21756	0.07088	21.44	.832	.765
1000	2	ours	0.15648	0.06088	10.73	1	
		rpart	0.19593	0.06042	11.51	.799	.704
		rpart0	0.14695	0.04875	25.78	1.065	1.113
	5	ours	0.14080	0.04016	12.06	1	
		rpart	0.18489	0.05206	13.02	.762	.640
		rpart0	0.15403	0.04916	28.44	.914	.855
	10	ours	0.13791	0.04503	12.44	1	
		rpart	0.18301	0.05667	13.62	.754	.626
		rpart0	0.15434	0.04817	28.72	.894	.821

Table 2: *Simulation study* **CV-RISK**. Full data simulated from $y = x^2 + er$, where $x \sim U(0,1)$ and $er \sim N(0, .0625)$. Candidate estimator was chosen over 100 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2) for both our algorithm and rpart (col 3). The results (cols 4-7) in table are based on an independent test sample of $n = 1000$. col 4 is the average of the 100 risks (with the L_2 loss function) for each method, col 5 is the variance of the risks over the 100 reps, col 6 is the average size (number of basis functions for ours and number of splits for rpart) and col 7 is the ratio of averaged risks (col 4) (ours/rparts).

Sample			100 Repetitions				
Size	$v-fold$	Method	mean	std dev	avg size	ratio	ratio2
250	2	ours	0.07014	0.00371	5.08	1	
		rpart	0.07416	0.00537	3.27	.946	.655
		rpart0	0.07064	0.00466	5.42	.993	.939
	5	ours	0.07016	0.00356	5.42	1	
		rpart	0.07322	0.00504	3.51	.958	.715
		rpart0	0.07058	0.00427	5.55	.994	.948
	10	ours	0.06948	0.00367	5.78	1	
		rpart	0.07237	0.00486	3.54	.960	.707
		rpart0	0.07028	0.00404	5.66	.989	.897
500	2	ours	0.06724	0.00351	6.11	1	
		rpart	0.07039	0.00391	3.67	.955	.601
		rpart0	0.06775	0.00356	5.16	.992	.903
	5	ours	0.06763	0.00305	6.55	1	
		rpart	0.06967	0.00375	4.04	.971	.715
		rpart0	0.06746	0.00317	5.94	1	1.03
	10	ours	0.06780	0.00339	6.96	1	
		rpart	0.06908	0.00359	4.15	.981	.805
		rpart0	0.06719	0.00312	6.2	1.009	1.13
1000	2	ours	0.06623	0.00332	7.02	1	
		rpart	0.06819	0.00332	4.42	.971	.656
		rpart0	0.06639	0.00319	6.06	.998	.959
	5	ours	0.06584	0.00328	8.01	1	
		rpart	0.06743	0.00369	4.54	.976	.677
		rpart0	0.06541	0.00345	6.84	1.006	1.148
	10	ours	0.06633	0.00337	8.63	1	
		rpart	0.06729	0.00324	4.73	.986	.7996
		rpart0	0.06553	0.00309	7.31	1.012	1.26

δ; 'ours(k)' chooses only over k with δ set at 7; 'add only' only utilizes the Addition Step in the algorithm with δ set at 7 (this is a type of forward selection); 'rpart0' is the CART estimator that chooses the tree thatminimizes the cross-validated risk. The other details of the simulation are exactly the same as in Section 4.1.

In Table 3, we see that 'ours(k, δ)' is approaching the truth faster ('ratio 2') in almost all situations compared to the other three estimators except for 'add only' in $n = 1000$. Importantly, in all situations the size of 'ours(k, δ)' is less. This means that 'ours(k, δ)' is coming up with the sparsest way of describing the data. In Table 4, we see similar results even with a larger variance in the model. Here the gain in efficiency and accuracy is not as great as in the previous table; however, the number of parameters for 'ours(k, δ)' still remains low in comparison to 'add only' and 'rpart0'.

In Tables 5 and 6, 'ours(k, δ)' does negligibly better at any point except for a small gain in $n = 1000$ in Table 5. Similar to Simulation 1, our assumption is that the proposed algorithm will not significantly supersede other histogram regression methods when confronted with underlying uniform distribution.

4.3 Simulation Study 3

The third simulation is designed to compare cross-validation selection methods to CART (via `rpart`) and *oracle* estimators. The oracle estimators are built similarly to the others estimators except an independent sample of $10,000$ evaluates the estimator instead of v-fold cross-validation. Thus, the oracle estimator approximates the truth. In this simulation the estimators under consideration are:

- OURS. The proposed algorithm minimizing over k, the number of basis functions, and δ the minimum number of observations in a region (or subset) in order to split it. The empirical risk function is minimized with a restriction of $M \in 10, 15, 20$ employed. Minimum of 5-fold cross-validation risk ($CV(k)$) is used to choose the 'best' of the candidate estimators, $k = 1, \ldots, M$.
- Sparse .0x The proposed algorithm as described in **OURS** except $x\%$ is added to the minimum $CV(k)$. The smallest k which falls in th interval $CV(k) + x\%$ is chosen.

Table 3: *Simulation study* **CV-RISK**. Full data simulated from $y = x^2 + er$, $x \sim NORM(0, 1)$ and $er \sim N(0, 0.5625)$. 50 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2). The results (cols 4-7) in table are based on an independent test sample of $n = 1000$. col 4 is the average of the 50 risks (with the L_2 loss function) for each method, col 5 is the variance of the risks over the 50 reps, col 6 is the average size.

Sample			50 Repetitions				
Size	$v - fold$	Method	mean	std dev	avg size	ratio	ratio2
250	2	ours(k, δ)	0.91890	0.12443	5.6(6)		
		ours(k)	1.02020	0.16043	4.6	0.90	0.78
		add only	0.94866	0.12981	14.1	0.95	0.92
		rpart0	1.01403	0.15204	7.9	0.91	0.79
	5	ours(k, δ)	0.88729	0.12735	6.2(5.6)		
		ours(k)	0.96799	0.18040	5.1	0.92	0.80
		add only	0.92756	0.12585	14.6	0.96	0.89
		rpart0	0.98300	0.15122	8.4	0.90	0.77
	10	ours(k, δ)	0.90082	0.12205	6.5(5.5)		
		ours(k)	0.96908	0.14049	5.5	0.93	0.83
		add only	0.95173	0.12876	15.1	0.95	0.87
		rpart0	0.99345	0.14993	8.7	0.91	0.79
500	2	ours(k, δ)	0.77999	0.09932	7(6.7)		
		ours(k)	0.80779	0.10913	6.3	0.97	0.89
		add only	0.75785	0.07929	15.2	1.03	1.11
		rpart0	0.78901	0.09653	10.4	0.99	0.96
	5	ours(k, δ)	0.75375	0.080503	7.9(5.7)		
		ours(k)	0.77583	0.08900	6.9	0.97	0.896
		add only	0.75748	0.07453	16.4	0.995	0.98
		rpart0	0.77456	0.08416	11.7	0.97	0.90
	10	ours(k, δ)	0.74842	0.06409	8.5(5.6)		
		ours(k)	0.77202	0.06919	7.4	0.97	0.87
		add only	0.75956	0.07667	17.4	0.99	0.94
		rpart0	0.77790	0.09251	12.1	0.96	0.86
1000	2	ours(k, δ)	0.68110	0.04817	8.9(6.3)		
		ours(k)	0.70124	0.06083	7.9	0.97	0.85
		add only	0.67018	0.04975	16.9	1.02	1.10
		rpart0	0.68681	0.05731	12.9	0.99	0.95
	5	ours(k, δ)	0.68142	0.02747	10.5(6)		
		ours(k)	0.69303	0.02694	9.5	0.98	0.91
		add only	0.67750	0.02394	17.9	1.01	1.03
		rpart0	0.69691	0.02666	15.2	0.98	0.88
	10	ours(k, δ)	0.68081	0.05745	10.5(5.6)		
		ours(k)	0.70599	0.069005	9.4	0.96	0.82
		add only	0.693293	0.067235	18.2	0.98	0.90
		rpart0	0.71248	0.074105	15.5	0.96	0.79

Table 4: *Simulation study* **CV-RISK**. Full data simulated from $y = x^2 + er$, $x \sim N(0,1)$ and $er \sim N(0,1)$. 50 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2). The results (cols 4-7) in table are based on an independent test sample of $n = 1000$. col 4 is the average of the 50 risks (with the L_2 loss function) for each method, col 5 is the variance of the risks over the 50 reps, col 6 is the average size.

Sample			50 Repetitions				
Size	$v-fold$	Method	mean	std dev	avg size	ratio	ratio2
250	2	ours(k,δ)	1.42876	0.19039	5.1(6)		
		ours(k)	1.50388	0.21861	4.3	0.95	0.85
		add only	1.46023	0.16760	12.9	0.98	0.93
		rpart0	1.49847	0.14625	6.2	0.95	0.86
	5	ours(k,δ)	1.37600	0.15532	5.5(6)		
		ours(k)	1.41195	0.15068	5.2	0.97	0.91
		add only	1.43679	0.16052	14.5	0.96	0.86
		rpart0	1.46440	0.16455	7.7	0.939	0.81
	10	ours(k,δ)	1.38398	0.14863	5.8(5.3)		
		ours(k)	1.41520	0.15748	5.3	0.98	0.93
		add only	1.45299	0.14411	15.4	0.95	0.85
		rpart0	1.46560	0.16638	7.2	0.94	0.82
500	2	ours(k,δ)	1.24121	0.07730	6.8(6.2)		
		ours(k)	1.29458	0.14130	5.9	0.96	0.82
		add only	1.25467	0.08901	15.5	0.99	0.95
		rpart0	1.27564	0.10481	9.4	0.97	0.88
	5	ours(k,δ)	1.23782	0.09230	7.2(5.7)		
		ours(k)	1.26498	0.09105	6.1	0.98	0.897
		add only	1.25763	0.08390	17	0.98	0.92
		rpart0	1.26562	0.09809	10	0.98	0.895
	10	ours(k,δ)	1.24538	0.10267	7.4(6)		
		ours(k)	1.24789	0.10408	6.9	0.99	0.99
		add only	1.25758	0.10143	17.1	0.99	0.95
		rpart0	1.26257	0.11444	10.1	0.99	0.93
1000	2	ours(k,δ)	1.15565	0.07665	8.4(6.4)		
		ours(k)	1.17479	0.08406	7.3	0.98	0.89
		add only	1.14613	0.07644	16.7	1.01	1.07
		rpart0	1.17308	0.083109	10.9	0.99	0.899
	5	ours(k,δ)	1.14230	0.06129	10(5.7)		
		ours(k)	1.14511	0.064496	8.5	0.997	0.98
		add only	1.13967	0.06021	17.9	1.0	1.02
		rpart0	1.15039	0.063386	12.6	0.99	0.95‘
	10	ours(k,δ)	1.15745	0.06233	9.95(6.1)		
		ours(k)	1.15737	0.07351	8.4	1.0	1.0
		add only	1.14641	0.06678	18	1.01	1.08
		rpart0	1.16241	0.07268	13.5	0.996	0.969

Table 5: *Simulation study* **CV-RISK**. Full data simulated from $y = x^2 + er$, $x \sim unif(0,1)$ and $er \sim N(0, 0.5625)$. 50 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2). The results (cols 4-7) in table are based on an independent test sample of $n = 10000$. col 4 is the average of the 50 risks (with the L_2 loss function) for each method, col 5 is the variance of the risks over the 50 reps, col 6 is the average size.

Sample			50 Repetitions				
Size	$v-fold$	Method	mean	std dev	avg size	ratio	ratio2
250	2	ours(k,δ)	0.61252	0.02184	3.5(.05)		
		ours(k)	0.60452	0.01641	3.2	1.01	1.19
		add only	0.65218	0.03192	12.3	.94	.56
		rpart0	0.60094	0.02216	2.4	1.02	1.3
	5	ours(k,δ)	0.62821	0.03091	4.1(.04)		
		ours(k)	0.62299	0.03154	3.6	1.01	1.09
		add only	0.64963	0.02993	11.3	.97	.75
		rpart0	0.59377	0.02307	2.7	1.06	1.98
	10	ours(k,δ)	0.63574	0.03293	4.0(.04)		
		ours(k)	0.62026	0.03042	3.8	1.02	1.26
		add only	0.66294	0.02795	13.9	.96	.73
		rpart0	0.59590	0.02281	2.7	1.07	2.19
500	2	ours(k,δ)	0.59330	0.01815	4.3(.05)		
		ours(k)	0.60105	0.01970	4.1	.99	.80
		add only	0.63093	0.01735	15.1	.94	.45
		rpart0	0.59277	0.01988	2.7	1	1.02
	5	ours(k,δ)	0.60360	0.01819	4.3(.05)		
		ours(k)	0.60788	0.02163	4.3	.99	.91
		add only	0.63555	0.01742	16.7	.95	.56
		rpart0	0.58673	0.01021	2.8	1.03	1.7
	10	ours(k,δ)	0.61003	0.02301	4.7(.03)		
		ours(k)	0.61198	0.02232	4.4	1.0	.96
		add only	0.63542	0.01850	16.9	.96	.65
		rpart0	0.58496	0.01602	3.1	1.011.04	2.1
1000	2	ours(k,δ)	0.57554	0.00979	4.9(.05)		
		ours(k)	0.58079	0.01043	4.5	.99	.71
		add only	0.59863	0.01156	15.5	.96	.36
		rpart0	0.57837	0.01115	2.8	.995	.82
	5	ours(k,δ)	0.58326	0.01166	4.9(.03)		
		ours(k)	0.59161	0.01478	5.1	.99	.71
		add only	0.60428	0.01063	17.2	.97	.50
		rpart0	0.57642	0.00951	3.4	1.01	1.5
	10	ours(k,δ)	0.58728	0.01372	5.1(.03)		
		ours(k)	0.59778	0.01729	5.1	.98	.70
		add only	0.60405	0.01349	17.8	.97	.60
		rpart0	0.57516	0.01122	3.6	1.02	1.96

Table 6: *Simulation study* **CV-RISK**. Full data simulated from $y = x^2 + er$, $x \sim unif(0,1)$ and $er \sim N(0,1)$. 50 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2). The results (cols 4-7) in table are based on an independent test sample of $n = 10000$. col 4 is the average of the 50 risks (with the L_2 loss function) for each method, col 5 is the variance of the risks over the 50 reps, col 6 is the average size.

Sample			50 Repetitions				
Size	$v-fold$	Method	mean	std dev	avg size	ratio	ratio2
250	2	ours(k,δ)	1.07887	0.03831	3.4(.05)	1	
		ours(k)	1.06419	0.03912	3.1(.03)	1.01	1.23
		add only	1.16395	0.04685	12.6	.93	.48
		rpart0	1.05864	0.02579	1.8	1.02	1.34
	5	ours(k,δ)	1.10572	0.05783	3.5(.05)	1	
		ours(k)	1.09595	0.05060	3.5	1.01	1.10
		add only	1.15806	0.04919	11.6	.95	.66
		rpart0	1.05771	0.05126	2.5	1.05	1.83
	10	ours(k,δ)	1.13338	0.05617	3.8(.03)	1	
		ours(k)	1.09249	0.05152	3.3	1.04	1.44
		add only	1.18075	0.05103	14.8	.96	.74
		rpart0	1.05373	0.05204	2.5	1.08	2.48
500	2	ours(k,δ)	1.04841	0.02776	3.8(.06)		
		ours(k)	1.05374	0.029934	4.0(.01)	.99	.90
		add only	1.11661	0.03643	15.0	.94	.42
		rpart0	1.03259	0.02104	2.1	1.02	1.49
	5	ours(k,δ)	1.06194	0.03128	3.8(.04)		
		ours(k)	1.06652	0.03523	3.7	.996	.93
		add only	1.12845	0.02870	17.2	.94	.48
		rpart0	1.02831	0.01544	2.2	1.08	2.19
	10	ours(k,δ)	1.07972	0.03252	4.1(.03)	1	
		ours(k)	1.09095	0.04195	3.9	.99	.88
		add only	1.13839	0.03059	17.4	.95	.58
		rpart0	1.03799	0.02161	2.4	1.04	2.1
1000	2	ours(k,δ)	1.02659	0.02041	4.2(.05)	1	
		ours(k)	1.02726	0.01729	4.1(.01)	1.0	.98
		add only	1.06692	0.02313	16	.96	.40
		rpart0	1.02242	0.016063	2.2	1.0	1.19
	5	ours(k,δ)	1.03701	0.02485	4.6(.04)		
		ours(k)	1.04759	0.03211	4.5	.99	.78
		add only	1.07384	0.024011	17	1.11	.50
		rpart0	1.01959	0.02417	3.1	1.02	1.89
	10	ours(k,δ)	1.04208	0.02203	4.3(.03)	1	
		ours(k)	1.05722	0.02389	4.4	.99	.74
		add only	1.07512	0.02016	17.2	.97	.56
		rpart0	1.02166	0.01482	3	1.02	1.94

- 1-SE The proposed algorithm as described in **OURS** except 'best' estimator is chosen finding the smallest k thats in the range of $min(CV(k))+ SE(min(CV(k)))$.
- MinCVrisk The proposed algorithm with the cross-validated empirical risk as the risk function.
- MINBUCK7 The proposed algorithm as described in **OURS** except only minimized over k with $\delta = 7$.
- ADDONLY A hybrid of the proposed algorithm whereas only the Addition Step is implemented. This is a type of forward selection approach.
- RPARTS The CART algorithm as implemented in `R` as `rpart`. Restriction of complexity parameter value as an argument in `rpart`. Minimizes over both k and δ.

In the following three figures, column 1 indicates the sample size 250, column 2 the error added to the underlying symmetric model, and column 3 shows any restrictions on OURS or RPARTS, e.g., $BF = 10$ means that OURS was limited to a maximum of $M = 10$ basis functions. Columns 4 through 7 are the results of the estimators chosen using 5-fold cross-validation. Column 4 indicates which method, Column 5 has the mean of the risk as assessed by an independent test sample of $10,000$ over 50 repetitions. Column 6 shows the average number of basis functions (over 50 repetitions) and column 7 indicates the δ chosen. Columns 8 through 10 are the results of the oracle estimators. These were chosen with an independent test sample of $10,000$ instead of 5-fold cross-validation. Column 9 is the relative efficiency ratio as described previously.

Figure 8 shows the results of the smallest sample size, $n = 250$. For the smaller error, OURS does quite well compared to it's own oracle estimator and to the others. By increasing M from 10 to 15 there is a marginal increase in accuracy. The biggest difference is when the error in the model is increased. Now OURS is not doing as well, ADDONLY and $1-SE$ are doing quite well in comparison to RPARTS though. Interestingly, The oracle estimator for OURS is doing better than the other estimators and oracles. This indicates that there is a problem with the selection of k and δ by cross-validation. If those problems are accommodated OURS could approach the accuracy of its oracle and thus, do the best.

Figure 9 shows very similar results at a much smaller scale. Now the discrepancy between OURS and RPARTS is not as big in the smaller error models and grows rapidly with a larger error. The second smallest oracle estimator is that of OURS. This indicates that if the cross-validation difficulty is remedied OURS would be potentially one of the best of these estimators. Interestingly, ADDONLY does the best in both of the larger error simulations.

Figure 10 the results are quite similar. However, OURS is doing quite badly in the largest error compared to RPART. It definitely does better when the number of basis functions is restricted to 15. OURS oracle estimator is also doing a slight bit worse than RPARTS'. When the error added to the model is smaller, OURS does marginally better.

5 Discussion

We have proposed a new partitioning algorithm for generating a piecewise constant estimation sieve of candidate estimators based on an intensive and comprehensive search over the entire covariate space. This new algorithm builds *'and'* and *'or'* statements. This allows combinations and substitutions of regions for the purpose of discovering intricate correlations patterns and interactions in addition to main effects. We suggest that this approach will supersede previously suggested methods by being not only more aggressive but also more flexible. In addition, we are currently implementing a bagging scheme with the intention of further improving prediction.

Sample size	Error	Restrict	Estimator Chosen by Cross-Validation Method	Mean	Size	Delta	Oracle Estimator Method	Mean	Size	Relative Efficiency
250	0.25	BF=10	OURS	0.283933	9.34	0.0208	OURS	0.274081	9.44	0.797
			Sparse .01	0.285164	8.9					
			Sparse .02	0.286492	8.52					
			Sparse .03	0.291409	7.86					
			1-SE	0.363773	5.10					
			MinCVrisk	0.291676	13.38		MinCVrisk	0.290050	13.24	0.825
			MINBUCK7	0.348052	8.82		MINBUCK7	0.341099	8.84	
			ADDONLY	0.304824	9.98		ADDONLY	0.299695	10	
		Cp=.001	RPARTS	0.340204	12.48		RPARTS	0.331355	12.96	
		BF=15	OURS	0.279152	12.94	0.0206	OURS	0.269621	11.86	0.786
			Sparse .01	0.279406	11.46					
			Sparse .02	0.279729	10.66					
			Sparse .03	0.283235	9.9					
			1-SE	0.359579	5.42					
			MinCVrisk	0.291676	13.38		MinCVrisk	0.290050	13.24	0.832
			MINBUCK7	0.344667	11.58		MINBUCK7	0.338233	11.32	
			ADDONLY	0.277417	14.94		ADDONLY	0.272266	14.8	
		CP=0	RPARTS	0.337966	15.84		RPARTS	0.328718	15.64	
	1	BF=10	OURS	1.441586	9	0.0211	OURS	1.309177	5.44	1.102
			Sparse .01	1.441670	8.36					
			Sparse .02	1.436106	8.08					
			Sparse .03	1.429405	7.78					
			1-SE	1.379621	5.78					
			MinCVrisk	1.409958	7.86		MinCVrisk	1.384494	7.18	1.023
			MINBUCK7	1.457210	8.22		MINBUCK7	1.363833	5.16	
			ADDONLY	1.334329	9.6		ADDONLY	1.300900	8.84	
		Cp=.005	RPARTS	1.400555	7.56		RPARTS	1.341487	7.92	
		BF=15	OURS	1.508310	10.86	0.0204	OURS	1.336559	5.5	1.179
			Sparse .01	1.504889	10					
			Sparse .02	1.498654	9.48					
			Sparse .03	1.493648	9.16					
			1-SE	1.431853	6.26					
			MinCVrisk	1.455152	7.74		MinCVrisk	1.404150	7.1	1.056
			MINBUCK7	1.501373	9.08		MINBUCK7	1.397916	5.26	
			ADDONLY	1.388528	13.14		ADDONLY	1.309970	8.98	
		CP=0	RPARTS	1.431067	7.58		RPARTS	1.357756	7.84	

Figure 8: *Simulation study* **Oracle vs. Cross-validation selector**. Full data simulated from $y = x^2 + er$, $x \sim N(0,1)$ and $er \sim N(0, error)$. 50 repetitions of sample size = 250 with error (col 2), basis functions (BF) and cp (col 3), selection method (col 4), average risk with L_2 loss function (col 5), average number of basis functions (or nodes) (col 6), chosen δ (col 7), and oracle estimator results based on independent test set of $10,000$ (col 8-10).

Sample size	Error	Restrict	Estimator Chosen by Cross-Validation Method	Mean	Size	Delta	Oracle Estimator Method	Mean	Size	Relative Efficiency
500	0.25	BF=10	OURS	0.189608	9.66	0.0105	OURS	0.186198	9.76	0.977
			Sparse .01	0.189873	9.58					
			Sparse .02	0.190672	9.18					
			Sparse .03	0.191562	8.92					
			1-SE	0.224406	6.84					
			MinCVrisk	0.181053	19.18		MinCVrisk	0.180600	19.04	0.911
			MINBUCK7	0.214364	9.46		MINBUCK7	0.211434	9.86	
			ADDONLY	0.243516	10		ADDONLY	0.239421	10	
		Cp=.001	RPARTS	0.192565	19.1		RPARTS	0.188683	19.42	
		BF=15	OURS	0.179309	13.48	0.0102	OURS	0.174786	14.2	0.899
			Sparse .01	0.179923	12.64					
			Sparse .02	0.180917	12.08					
			Sparse .03	0.181666	11.48					
			1-SE	0.216560	7.38					
			MinCVrisk	0.181053	19.18		MinCVrisk	0.180600	19.04	0.913
			MINBUCK7	0.206912	13.34		MINBUCK7	0.202296	13.8	
			ADDONLY	0.191173	15		ADDONLY	0.188642	15	
		CP=0	RPARTS	0.192409	21.6		RPARTS	0.186567	22.4	
	1	BF=10	OURS	1.288866	9.48	0.0107	OURS	1.201862	6.56	1.226
			Sparse .01	1.290389	8.82					
			Sparse .02	1.290850	8.66					
			Sparse .03	1.288236	8.52					
			1-SE	1.267996	7.08					
			MinCVrisk	1.289258	10.08		MinCVrisk	1.255939	9.26	1.228
			MINBUCK7	1.284121	9.3		MINBUCK7	1.229240	6.88	
			ADDONLY	1.210023	9.96		ADDONLY	1.202851	9.84	
		Cp=.005	RPARTS	1.235523	9.68		RPARTS	1.204612	10.88	
		BF=15	OURS	1.385647	13.54	0.0100	OURS	1.202157	6.5	1.637
			Sparse .01	1.384786	12.72					
			Sparse .02	1.381353	12.26					
			Sparse .03	1.375156	11.98					
			1-SE	1.334312	10.18					
			MinCVrisk	1.289258	10.08		MinCVrisk	1.255939	9.26	1.228
			MINBUCK7	1.354310	12.92		MINBUCK7	1.229620	6.8	
			ADDONLY	1.213414	14.44		ADDONLY	1.182773	12.06	
		CP=0	RPARTS	1.235523	9.68		RPARTS	1.204612	10.88	

Figure 9: *Simulation study* **Oracle vs. Cross-validation selector**. Full data simulated from $y = x^2 + er$, $x \sim N(0,1)$ and $er \sim N(0, error)$. 50 repetitions of sample size = 500 with error (col 2), basis functions (BF) and cp (col 3), selection method (col 4), average risk with L_2 loss function (col 5), average number of basis functions (or nodes) (col 6), chosen δ (col 7), and oracle estimator results based on independent test set of $10,000$ (col 8-10).

Sample size	Error	Restrict	Estimator Chosen by Cross-Validation Method	Mean	Size	Delta	Oracle Estimator Method	Mean	Size	Relative Efficiency
1000	0.25	BF=15	OURS	0.137484	14.61	0.0051	OURS	0.136517	14.86	0.857
			Sparse .01	0.138429	13.86					
			Sparse .02	0.140330	13.29					
			Sparse .03	0.140851	12.96					
			1-SE	0.169069	9.11					
			MinCVrisk	0.139129	25.11		MinCVrisk	0.138034	25.11	0.876
			MINBUCK7	0.153104	14.50		MINBUCK7	0.151968	15.00	
			ADDONLY	0.160150	15.00		ADDONLY	0.158697	15.00	
		Cp=.0001	RPARTS	0.149979	28.71		RPARTS	0.144944	30.46	
	1	BF=10	OURS	1.183230	9.74	0.0057	OURS	1.142713	7.98	1.155
			Sparse .01	1.186038	8.86					
			Sparse .02	1.186785	8.62					
			Sparse .03	1.188314	8.30					
			1-SE	1.182286	7.38					
			MinCVrisk	1.239488	14.02		MinCVrisk	1.190828	12.00	1.510
			MINBUCK7	1.176304	9.78		MINBUCK7	1.155677	7.86	
			ADDONLY	1.178627	10.00		ADDONLY	1.172320	9.96	
		Cp=.002	RPARTS	1.158613	12.76		RPARTS	1.136611	13.14	
		BF=20	OURS	1.328907	18.56	0.0050	OURS	1.141729	8.13	2.090
			Sparse .01	1.326708	17.29					
			Sparse .02	1.323793	16.85					
			Sparse .03	1.318888	16.23					
			1-SE	1.295150	14.83					
			MinCVrisk	1.240015	14.08		MinCVrisk	1.189328	11.98	1.525
			MINBUCK7	1.288648	17.96		MINBUCK7	1.154195	7.96	
			ADDONLY	1.147161	19.90		ADDONLY	1.120470	14.90	
		Cp=.002	RPARTS	1.157388	12.94		RPARTS	1.133392	13.77	

Figure 10: *Simulation study* **Oracle vs. Cross-validation selector**. Full data simulated from $y = x^2 + er$, $x \sim N(0,1)$ and $er \sim N(0, error)$. 50 repetitions of sample size = 1000 with error (col 2), basis functions (BF) and cp (col 3), selection method (col 4), average risk with L_2 loss function (col 5), average number of basis functions (or nodes) (col 6), chosen δ (col 7), and oracle estimator results based on independent test set of $10,000$ (col 8-10).

References

L. Breiman, J. H. Friedman, R.A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks/Cole, Monterey, CA, 1984.

R. D. Gill, M. J. van der Laan, and J. R. Robins. Coarsening at random: Characterizations, conjectures and counter-examples. In D. Y. Lin and T. R. Fleming, editors, *Proceedings of the First Seattle Symposium in Biostatistics, 1995*, Springer Lecture Notes in Statistics, pages 255–294, 1997.

W. Härdle. *Applied nonparametric regression*. Number 17 in Econometric Society Monographs. Cambridge University Press, 1989.

R. Ihaka and R. Gentlemen. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299–314, 1996.

C. Kooperberg, I. Ruczinski, M. LeBlanc, and L. Hsu. Sequence analysis using logic regression. *Genetic Epidemiology*, S1:626–631, 2001.

A. M. Molinaro, S. Dudoit, and M. J. van der Laan. Tree-based multivariate regression and density estimation based on right-censored data. *Journal of Multivariate Analysis*, 90:154–177, 2004.

J. Robins and A. Rotnitzky. *Recovery of information and adjustment for dependent censoring using surrogate markers*, chapter AIDS Epidemiology, Methodological issues. Bikhauser, 1992.

I. Ruczinski, C. Kooperberg, and M. LeBlanc. Logic regression. *Journal of Computational and Graphical Statistics*, 12(3):474–511, 2003. URL `biostat.jhsph.edu/~iruczins/publications/publications.html`.

S. Sinisi and M. J. van der Laan. Loss based cross-validated deletion/substitution/addition algorithm in estimation. Technical Report 143, Division of Biostatistics, University of California, Berkeley, 2004. URL `www.bepress.com/ucbbiostat/paper143/`.

T. Therneau and E. Atkinson. An introduction to recursive partitioning using the rpart routine. Technical Report 61, Section of Biostatistics, Mayo Clinic, Rochester, 1997.

M. J. van der Laan and S. Dudoit. Unified cross-validation methods for selection among estimators: Finite sample results, asymptotic optimality, and applications. Technical Report 130, Division of Biostatistics, University of California, Berkeley, 2003. URL `www.bepress.com/ucbbiostat/paper130/`.

M. J. van der Laan and J. Robins. *Unified Methods for Censored Longitudinal Data and Causality*. Springer, 2002.