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

Year 2004 *Paper* 143

Loss-Based Cross-Validated Deletion/Substitution/Addition Algorithms in Estimation

Sandra E. Sinisi* Mark J. van der Laan†

*Division of Biostatistics, School of Public Health, University of California, Berkeley, sinisi54@alum.berkeley.edu

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

This working paper is hosted by The Berkeley Electronic Press (bepress) and may not be commercially reproduced without the permission of the copyright holder.

http://biostats.bepress.com/ucbbiostat/paper143

Copyright ©2004 by the authors.

Loss-Based Cross-Validated Deletion/Substitution/Addition Algorithms in Estimation

Sandra E. Sinisi and Mark J. van der Laan

Abstract

In van der Laan and Dudoit (2003) we propose and theoretically study a unified loss function based statistical methodology, which provides a road map for estimation and performance assessment. Given a parameter of interest which can be described as the minimizer of the population mean of a loss function, the road map involves as important ingredients cross-validation for estimator selection and minimizing over subsets of basis functions the empirical risk of the subset-specific estimator of the parameter of interest, where the basis functions correspond to a parameterization of a specified subspace of the complete parameter space. In this article we first review this approach. Then we propose a general deletion/substitution/addition algorithm for minimizing over subsets of variables (e.g., basis functions) the empirical risk of subset-specific estimators of the parameter of interest. In particular, in the regression context, this algorithm corresponds to minimizing over subsets of variables the sum of squared residuals of the subset-specific linear regression estimator. This algorithm provides us with a new class of loss-based cross-validated algorithms in prediction of univariate and multivariate outcomes, conditional density and hazard estimation, and we generalize it to censored outcomes such as survival. In the context of regression, using polynomial basis functions, we study the properties of the deletion/substitution/addition algorithm in simulations and apply the method to detect binding sites in yeast gene expression experiments.

1 Introduction.

A dominating feature of statistical estimation and inference problems in genomics is that they involve parameter estimation for high-dimensional multivariate distributions, with typically unknown and intricate correlation patterns among variables. Accordingly, statistical models for the data generating distribution correspond to large parameter spaces. For instance, for the prediction of clinical outcomes using whole genome microarray measures of gene expression (i.e., thousands of gene expression measures), the parameter space may consist of the set of all possible linear combinations of tensor products of univariate (e.g., polynomial, wavelets, splines) basis functions of the explanatory variables, in order to allow for arbitrary subsets of explanatory variables and their functional form. Even if it were possible to minimize a suitable empirical mean of a loss function (e.g., empirical mean of residual sum of squared errors or log-likelihood) over the entire parameter space, the resulting estimators would be too variable and ill-defined.

A unified framework to approach these problems and its theoretical foundations are established in van der Laan and Dudoit (2003). This earlier technical report proposes a unified loss function based methodology for estimator construction, selection and performance assessment, and in particular provides finite sample results and asymptotic optimality results concerning v-fold cross-validation estimator selection for general data generating distributions, loss functions (possibly depending on a nuisance parameter), and estimators. These new theoretical results have the important practical implication that cross-validation selection among many candidate estimators can be used in intensive searches of large parameter spaces, even in finite sample situations. In particular, it is shown that explicit estimators of the type as presented in this article are minimax and adaptive to the truth due to such aggressive searches among candidate estimators. Special cases and applications are described in a collection of related articles: estimator selection and performance assessment based on uncensored data (Dudoit and van der Laan, 2003), estimator selection with right censored data (Keleş et al., 2003), likelihood-based cross-validation (van der Laan et al., 2003), tree-based estimation with censored data (Molinaro et al., 2003). An overall description of the estimation methodology with examples of the deletion/substitution/addition algorithms as presented in this article and in the context of histogram regression is available in (Dudoit et al., 2003).

Informally, the approach can be summarized as follows. First, one devel-

Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

ops an algorithm that is capable of adapting to the data completely, that is, it is an algorithm that is capable of minimizing the empirical mean of the loss function over an arbitrarily good approximation of the complete parameter space. Due to the number of variables involved and limited sample size, the non-parametric version of this algorithm is too variable: the estimator will fit the data perfectly. Consequently, this algorithm must be indexed by a number of "brakes," which correspond to specified subspaces of the complete parameter space. Natural collection of brakes are obtained by parameterizing the complete parameter space in terms of linear combinations of basis functions, so that the choice of a basis, the number of basis functions, a complexity measure of the basis functions, and a constraint on the vector of coefficients (e.g., norm), provide natural choices for brakes. Based on our cross-validation results, even when one implements a large number of brakes, the resulting estimator will perform asymptotically exactly as well as the estimator corresponding to the oracle selector of brakes. As a consequence, the estimator adapts at an asymptotically increasing level to the truth when more and more brakes are applied. In van der Laan and Dudoit (2003) this is completely formalized in terms of a finite sample inequality and corresponding asymptotics for a cross-validated adaptive ϵ-net estimator, which is of the type described above, so that the constraint on the vector of coefficients is discrete valued on a δ-grid, where δ represents the brake which needs to be chosen by cross-validation.

Currently available algorithms used in (e.g.) regression are not aiming to minimize empirical risk over specified parameter spaces (e.g, forward variable selection, recursive partitioning in classification and regression trees, (Breiman et al., 1984), multivariate adaptive regression splines, (Friedman, 1991; Hastie et al., 2001)). These algorithms do not adhere to the theoretically validated estimation road map of Dudoit and van der Laan (2003), which has motivated us to construct deletion/substitution/addition algorithms which search aggressively over subsets of basis functions. These algorithms aim to minimize over all allowed linear combinations of maximally k basis functions, while one still selects k, among possibly other fine-tuning parameters, with cross-validation.

Overview. In this article we will first define in detail (and discuss) the general road map for loss-function based estimation. The fundamental ingredients are 1) defining subspaces of the parameter space, 2) minimizing empirical risk of the loss function over each given subspace, and 3) using cross-validation to choose among the candidate estimators. Subsequently,

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

we propose a deletion/substitution/addition (D/S/A) algorithm for minimizing empirical risk over a subspace (i.e., ingredient 2). In the context of regression, we will study the performance of the D/S/A algorithm in terms of asymptotically achieving the global minimum over the complete parameter space and the properties of the resulting data adaptive estimator in simulation studies. Finally, we apply the methodology to a yeast data set to detect binding sites.

2 Unified loss-function based methodology.

In this section we will present a statistical framework which allows a unified treatment of the high dimensional estimation problems in genomics and other fields. Let $O_1, \ldots, O_n$ be n i.i.d. observations of $O \sim P_0$, where P_0 is known to be an element of a statistical model $\mathcal{M}$. Let $\Psi : \mathcal{M} \to D(\mathcal{S})$ be a mapping, a so-called parameter, from the model $\mathcal{M}$ into a space of real valued functions from a Euclidean set $\mathcal{S} \subset \mathbb{R}^d$, and let $\psi_0 = \Psi(P_0)$ be the true parameter value (i.e., a function). We will denote the parameter space of this parameter with $\mathbf{\Psi} = \{\Psi(P) : P \in \mathcal{M}\} \subset D(\mathcal{S})$. Each parameter value $\psi : \mathcal{S} \to \mathbb{R}$ is a function from a certain d-dimensional Euclidean set $\mathcal{S} \subset \mathbb{R}^d$ to the real line. Note that the use of upper case Ψ and lower case ψ allows us to distinguish between the mapping $\Psi : \mathcal{M} \to \mathcal{D}(\mathcal{S})$ and actual realizations of this mapping ψ, which are functions from $\mathcal{S}$ to the real line.

Estimators. Let P_n denote the *empirical distribution* of $O_1, \ldots, O_n$, where P_n places probability $1/n$ on each realization O_i. Our goal is to use the sample to estimate the parameter $\psi_0 = \Psi(P_0)$ of the unknown data generating distribution P_0. An *estimator* $\hat{\Psi}$ is a mapping from empirical distributions to the parameter space $\mathbf{\Psi}$, and its realization will be denoted with $\hat{\psi} = \hat{\Psi}(P_n)$. Note that estimators are viewed as algorithms one can apply to any empirical distribution, not as the actual realizations at the observed P_n.

In the next subsection we will present the general road map for loss function based estimation (van der Laan and Dudoit (2003)), and then we will discuss the ingredients of the road map, which will motivate the need for a new class of algorithms as presented in this article.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

2.1 Estimation road map.

Our general road map for estimation is driven by the choice of loss function and relies on cross-validation for estimator selection and performance assessment. We will now describe the steps of this road map and illustrate them with general examples.

Defining the parameter of interest in terms of a loss function.

Let $(O, \psi) \rightarrow L(O, \psi) \in \mathbb{R}$ be a (loss) function which maps a candidate parameter value $\psi \in \mathbf{\Psi}$ and observation O into a real number, whose expectation is minimized at ψ_0:

$$\begin{aligned} \psi_0 &= \text{argmin}_{\psi \in \mathbf{\Psi}} \int L(o, \psi) dP_0(o) \\ &= \text{argmin}_{\psi \in \mathbf{\Psi}} E_0 L(O, \psi). \end{aligned} \tag{1}$$

We will adopt terminology from the prediction literature. The risk of a candidate $\psi \in \mathbf{\Psi}$ is defined as $E_{P_0} L(O, \psi)$, and we will refer to the function $\psi \rightarrow E_0 L(O, \psi)$ as the risk function. The difference between the risk at ψ and the minimal risk at ψ_0 will be called the risk difference at ψ:

$$d(\psi, \psi_0) \equiv E_0\{L(O, \psi) - L(O, \psi_0)\}.$$

The conditional risk of the estimator $\hat{\Psi}(P_n)$ (given P_n) is defined as

$$\int L(o, \hat{\Psi}(P_n)) dP_0(o).$$

The marginal risk is the expectation of this conditional risk. Finally, $\int L(o, \hat{\Psi}(P_n)) dP_n(o)$ is called the empirical risk estimator, which can be viewed as an estimator of both conditional risk as well as marginal risk.

The estimation problem covers, in particular, multivariate regression and multivariate density estimation. For example, if $O = (Y, W) \sim P_0$, Y is an outcome, W is a vector of covariates, $\psi_0(W) = E_{P_0}(Y \mid W)$ is the parameter of interest, then we can set the loss function equal to the quadratic loss function:

$$L(Y, W, \psi) = (Y - \psi(W))^2.$$

Note that for this choice of loss function $d(\psi, \psi_0) = \int (\psi(W) - \psi_0(W))^2 dP_0(W)$.

COBRA
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Similarly, if $O = (Y = (Y_1, \ldots, Y_l), W)$, Y is a multivariate random outcome vector and W a vector of covariates, the multivariate conditional expectation $\psi_0(W) \equiv E_{P_0}(Y \mid W) = (E_0(Y_1 \mid W), \ldots, E_0(Y_l \mid W))$ is the parameter of interest, then we can define the loss function as

$$L(O, \psi) \equiv (Y - \psi(W))^\top \eta(W)(Y - \psi(W)),$$

where η is a symmetric $l \times l$-matrix function of W. Note that for any symmetric matrix function $\eta(W)$ we have

$$\psi_0 = \text{argmin}_{\psi \in \Psi} E_0 L(O, \psi). \tag{2}$$

Here η could denote an approximation of an unknown matrix such as

$$\left[E_0\left(\{Y - E_0(Y \mid W)\}\{Y - E_0(Y \mid W)\}^\top \mid W\right)\right]^{-1}.$$

It is straightforward to show that for this choice of loss function we have $d(\psi, \psi_0) = \int \parallel \eta^{0.5}(W)(Y - \psi(W)) \parallel^2 dP_0(W)$, where $\parallel \cdot \parallel$ denotes the Euclidean norm of the l-dimensional vector (van der Laan and Dudoit (2003)).

If $O \sim f_0 \equiv \frac{dP_0}{d\mu}$, where μ is a dominating measure of the data generating distribution P_0, $\psi_0 = f_0$ is the parameter of interest, then we can define the loss function as

$$L(O, \psi) = -\log(\psi(O)),$$

so that $d(\psi, \psi_0) = \int \log(\psi_0(O)/\psi(O))\psi_0(O)d\mu(O)$ is the Kullback-Leibler divergence.

This same loss function applies to the case that $O = (Y, W)$, and ψ_0 is the conditional density of an outcome Y, given V, where $V \subset W$, but, obviously, now the parameter space consists of conditional densities. Similarly, if the parameter of interest is the conditional hazard $\lambda_0(\cdot \mid W)$ of a failure time Y, given W, then we can choose as loss function

$$L(O, \psi) = -\log(\psi(Y \mid W)) + \int_0^Y \psi(u \mid W)d\mu(u).$$

For this choice of loss function we have that

$$d(\psi, \psi_0) = \int \log(f_0(O)/f(O))f_0(O)d\mu(O),$$

where f, f_0 are the densities corresponding with the conditional hazards ψ, ψ_0, respectively. That is, the risk distance between the hazard and the true hazard reduces to the Kullback-Leibler divergence for their corresponding densities.

BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

Censored data. In censored data situations, we apply the general *estimating function* methodology in van der Laan and Robins (2003) 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. We will present the details for the general right-censored data structure in Section 6. The unified loss function based methodology in van der Laan and Dudoit (2003) covers the censored data case as a special case of loss functions $L(O, \psi \mid \upsilon)$ depending on a nuisance parameter $\Upsilon(P)$, where $\psi_0 = \text{argmin}_{\psi \in \mathbf{\Psi}} E_0 L(O, \psi \mid \upsilon_0)$, and $\upsilon_0 = \Upsilon(P_0)$ is the true nuisance parameter value. In addition, van der Laan and Dudoit (2003) provide various general censored data examples and theorems specific to censored data loss functions.

Selecting a sieve.

The minimum empirical risk estimator (e.g., maximum likelihood estimator or least squares estimator in the examples above)

$$\text{argmin}_{\psi \in \mathbf{\Psi}} \int L(O, \psi) dP_n(O)$$

might not be well defined and/or suffer from the curse of dimensionality. A general approach advocated in the theoretical literature for dealing with this problem is too construct a sequence of subspaces approximating the whole parameter space $\mathbf{\Psi}$, a so-called sieve, and select the actual subspace whose corresponding minimum empirical risk estimator minimizes an appropriately penalized empirical risk (Barron et al., 1999) or a cross-validated empirical risk. In most multivariate algorithms used in practice (e.g., machine learning) this approach is not followed closely in the sense that one uses algorithms which do not aim to minimize empirical risk for specified subspaces.

We propose the following approach.

- Parameterize the parameter space $\mathbf{\Psi}$ in terms of a known transformation $g(\cdot)$ (e.g., identity, the logit function in binary classification, or the exponential function in the Cox proportional hazards models, or more general, multiplicative intensity models) of linear combinations of basis functions:

$$\mathbf{\Psi} = \left\{ g\left(\sum_{j \in I} \beta_{1j} \Phi_{j, \beta_{2j}} \right) : I \subset \mathcal{I}, \beta \right\},$$

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

where $\{\Phi_{j,\beta_{2j}} : j\}$ denotes a collection of basis functions, possibly indexed by a parameter β_{2j}, I is a particular finite or countable subset of indices, $\mathcal{I}$ denotes the collection of indices of the allowed subsets of basis functions, and β here ranges over Euclidean sets guaranteeing that the linear combinations are contained in the parameter space: $\sum_{j\in I} \beta_{1j}\Phi_{j,\beta_{2j}} \in \mathbf{\Psi}$.

This class of parameterizations and corresponding sieves covers most of the examples in the literature. In projection pursuit regression one chooses $\Phi_{j,\beta_{2j}}(w) = \Phi_j(\beta_{2j}w)$, where $\beta_{2j}w = \sum_{l=1}^{d} \beta_{2j,l}w_l$ represents linear combinations of the original variables w and Φ_j are known functions. In particular, single layer neural networks correspond with $\Phi_{j,\beta_{2j}}(w) = \Phi_0(\beta_{2j}w)$, where Φ_0 is a known cumulative distribution function, which is often called a sigmoidal function (see Ripley (1996), Hastie et al. (2001)). We refer to Barron (1993) for universal approximation results for such neural networks.

Tensor products. In many examples the basis functions Φ_j are not indexed by parameters β_{2j}. A general class of basis functions can be obtained as tensor products of univariate basis functions. Let $e_1 = 1, e_2, \ldots$ be a collection of univariate basis functions such as the polynomial powers (i.e., $1, x, x^2, \ldots$), spline basis functions of fixed degree with corresponding fixed set of knot points, or wavelet basis functions. This choice of a class of basis functions can itself be indexed by a continuous parameter (which can be chosen with cross-validation) such as the smoothness degree of splines, width between the knot points of the splines, or the smoothness degree of the wavelets. Given a vector $\vec{p} = (p_1, \ldots, p_d) \in \mathbb{N}^d$, let $\phi_{\vec{p}} = e_{p_1}(W_1) \times \ldots \times e_{p_d}(W_d)$ denote the tensor product of univariate basis functions identified by $\vec{p}$. For instance, if we use polynomial basis functions, then $\phi_{\vec{p}}(W) = W_1^{p_1} \ldots W_d^{p_d}$. The collection $\{\phi_{\vec{p}} : \vec{p}\}$ provides now a basis for the complete parameter space $\mathbf{\Psi}$: in particular, if e_j, $j = 1, \ldots$, is an orthonormal basis in a Hilbert space $(L^2(\lambda), \langle\cdot,\cdot\rangle_\lambda)$, where $\langle f, g\rangle_\lambda = \int f(u)g(u)d\lambda(u)$, then the corresponding tensor products provide an orthonormal basis w.r.t. the corresponding Hilbert space for multivariate real valued functions endowed with the inner product $\langle\psi_1, \psi_2\rangle = \int \psi_1(s)\psi_2(s) \prod_{j=1}^{d} d\lambda(s_j)$.

In general, it appears that an index $\vec{p} \in \mathbb{N}^d$ (or $\mathbb{N}^{d_1}$ for some integer d_1) provides a natural way of indexing basis functions for classes of

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

d-dimensional real valued functions. For example, even in single layer neural networks, we can define for each $\vec{p} \in \{0,1\}^d$ $\Phi_{\vec{p},\beta_2} = \Phi_0(\beta_2 \vec{p} w)$, where $\beta_2 \vec{p} w = \sum_{l=1}^d \beta_{2,l} p_l w_l$, so that $\vec{p}$ indicates which of the d variables can be used in the linear combination. This results in a parameterization of single layer neural networks of the form $\sum_{\vec{p} \in I} \beta_{1,\vec{p}} \Phi_{\vec{p},\beta_{2,\vec{p}}}$. Therefore, for concreteness of our D/S/A algorithm, in the remainder of this article we will consider basis functions $\{\Phi_{\vec{p},\beta_{2\vec{p}}} : \vec{p} \in \mathbb{N}^d\}$ indexed by d-dimensional vectors of integers. Thus the index set I represents a set of elements in $\mathbb{N}^d$.

For each index set I consisting of a collection of vectors $\vec{p}$, let

$$\psi_{I,\beta} \equiv g\left(\sum_{\vec{p} \in I} \beta_{1,\vec{p}} \Phi_{\vec{p},\beta_{2,\vec{p}}}\right)$$

be the function in the parameter space corresponding to the variables and model identified by I and $\beta = (\beta_{\vec{p}} = (\beta_{1\vec{p}}, \beta_{2,\vec{p}}) : \vec{p} \in I)$. For a given index set I, let $B_I = \{\beta : \psi_{I,\beta} \in \mathbf{\Psi}\}$ be the corresponding parameter space for β.

- Given this parameterization, one defines a collection of subspaces $\mathbf{\Psi}_s \subset \mathbf{\Psi}$, indexed by an s ranging over a set $\mathcal{A}_n$ possibly depending on sample size. Such subspaces are obtained by restricting the subsets I of basis functions to be contained in $\mathcal{I}_s \subset \mathcal{I}$, and/or restricting the values for the corresponding coefficients $(\beta_{\vec{p}} : \vec{p} \in I)$ to be contained in $B_{I,s} \subset B_I$:

$$\mathbf{\Psi}_s = \{\psi_{I,\beta} : I \in \mathcal{I}_s \subset \mathcal{I}, \beta \in B_{I,s} \subset B_I\}.$$

For example, such a subspace could be defined as all linear combinations of maximally s_1 basis functions, where the "dimension" of each basis function indexed by $\vec{p}$, as measured by some function $m(I)$ (e.g., number of terms in the tensor product defining the basis function), is bounded by s_2, and the norm of the β-vector is bounded by a value s_3:

$$\mathbf{\Psi}_s = \left\{\sum_{\vec{p} \in I} \beta_{1,\vec{p}} \Phi_{\vec{p},\beta_{2,\vec{p}}} \in \mathbf{\Psi} :\mid I \mid \leq s_1, \max_{\vec{p} \in I} m(I) \leq s_2, \parallel \beta \parallel \leq s_3\right\}. \quad (3)$$

For example in the polynomial case allowing $m(I)$ to be bivariate makes sense where $m_1(I) = \sum_{j=1}^d I(p_j \neq 0)$ is the number of non-zero components in $\vec{p}$ and $m_2(I) = \sum_{j=1}^d p_j$, and $\parallel \vec{\beta} \parallel \equiv \sum_j \mid \beta_j \mid$ is the Manhattan norm of the coefficient vector.

Collection of Biostatistics Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Additional subspaces are obtained by also varying the collection of basis functions, where the additional different choices of basis functions do not necessarily generate the complete parameter space. For example, the above construction of subspaces can be carried out for polynomial basis, wavelet basis with different degrees of smoothness, and linear spline basis functions.

Construction of candidate estimators.

For each $s \in \mathcal{A}_n$, compute (or approximate as best as one can) the minimizer of the empirical risk over the subspace $\mathbf{\Psi}_s$:

$$\psi_s(P_n) \equiv \text{argmin}_{\psi \in \Psi_s} \int L(O, \psi) dP_n(O).$$

This minimization problem is naturally split into two sequential steps. Given each possible subset $I \in \mathcal{I}_s$ of basis functions, compute the corresponding minimum risk estimator of β:

$$\beta(P_n \mid I, s) \equiv \text{argmin}_{\beta \in B_{I,s}} \int L\left(O, \psi_{I,\beta}\right) dP_n(O).$$

For example, in regression with $g(x) = x$ and parameter-free basis functions $\Phi_{\vec{p}}$, this corresponds to minimizing the sum of the squared residuals over the linear regression model in the basis functions $\Phi_{\vec{p}}$ indexed by $\vec{p} \in I$. If the basis functions are indexed by a parameter, then computing $\beta(P_n \mid I, s)$ corresponds to a non-linear minimization problem. For each I this results in an estimator $\hat{\psi}_{I,s} = \Psi_{I,s}(P_n) \equiv \psi_{I,\beta(P_n|I,s)}$.

Now, it remains to minimize the empirical risk over all allowed subsets $I \in \mathcal{I}_s$ of basis functions. Specifically, one needs to minimize the function $f_{E,s} : \mathcal{I}_s \to \mathbb{R}$ defined by

$$f_{E,s}(I) \equiv \int L\left(O, \Psi_{I,s}(P_n)\right) dP_n(O). \tag{4}$$

Suppose that $s = (s_1, s_2)$, where s_1 denotes the maximal number of basis functions, and s_2 represents the additional constraints to be fine-tuned (e.g., see example (3) above). Let

$$I_s(P_n) \equiv \text{argmin}_{I \in \mathcal{I}_s} f_{E,s}(I)$$

be the minimizer. In Section 3 we propose a deletion/substitution/addition algorithm, where one run of the algorithm for a fixed s_2 seeks to calculate $I_{s_1,s_2}(P_n)$ for all $s_1 = 1, \ldots$.

Collection of Biostatistics Research Archive

Hosted by The Berkeley Electronic Press

Selection among candidate estimators: Cross-validation.

This provides us now with the empirical risk minimizer $\hat{\psi}_s = \hat{\Psi}_s(P_n)$ for each choice of subspace s, where

$$\hat{\Psi}_s(P_n) = \Psi_{I_s(P_n),s}(P_n) = \psi_{I_s(P_n),\beta(P_n|I_s(P_n),s)}.$$

Finally, we select s with cross-validation. Let $B_n \in \{0,1\}^n$ be a random vector whose observed value defines a split of the observed data $O_1, \ldots, O_n$, also called learning sample, into a validation sample and a training sample. If $B_n(i) = 0$ then observation i is placed in the training sample and if $B_n(i) = 1$, it is placed in the validation sample. The choice of distribution of B_n now corresponds to different possible cross-validation schemes presented in the literature such as v-fold cross-validation and Monte-Carlo cross-validation. We will denote the empirical distribution of the data in the training sample and validation sample with P^0_{n,B_n} and P^1_{n,B_n}, respectively. The proportion of observations in the validation sample is denoted with $p = \sum_i B_n(i)/n$. The bootstrap cross-validation scheme can also be included by defining $B_n(i)$ as the number of times observation i occurs in the bootstrap sample, P^1_{n,B_n} is the empirical distribution of all O_i with $B_n(i) > 0$, and P^0_{n,B_n} is the weighted empirical distribution of the remaining sample with weights being the counts $B_n(i)$. The cross-validation selector of s is now defined as

$$\begin{aligned} s(P_n) &\equiv \text{argmin}_{s\in\mathcal{A}_n} E_{B_n} \int L(O, \hat{\Psi}_s(P^0_{n,B_n}))dP^1_{n,B_n}(O) \\ &= \text{argmin}_{s\in\mathcal{A}_n} E_{B_n} \frac{1}{np} \sum_{i=1}^{n} I(B_n(i) = 1)L(O_i, \hat{\Psi}_s(P^0_{n,B_n})). \end{aligned}$$

Our final estimator of ψ_0 is given by $\hat{\psi} = \hat{\Psi}(P_n) \equiv \Psi_{s(P_n)}(P_n)$.

In case the loss function depends on a nuisance parameter υ_0, as in the right-censored data case presented in Section 6, then one also estimates the nuisance parameter $\upsilon_0 = \Upsilon(P_0)$ based on the training samples. That is, given an estimator $\hat{\Upsilon}$ of the nuisance parameter υ_0, we have

$$s(P_n) \equiv \text{argmin}_{s\in\mathcal{A}_n} E_{B_n} \int L(O, \hat{\Psi}_s(P^0_{n,B_n}) \mid \hat{\Upsilon}(P^0_{n,B_n}))dP^1_{n,B_n}(O).$$

For the finite sample inequalities comparing the risk distance of the cross-validation selected estimator with the risk distance of the estimator chosen by the oracle selector $\tilde{s}(P_n) \equiv \text{argmin}_{s\in\mathcal{A}_n} E_{B_n} \int L(O, \psi_s(P^0_{n,B_n}))dP_0(O)$, and its

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

asymptotic implications, we refer to van der Laan and Dudoit (2003) for results for general loss functions, and Dudoit and van der Laan (2003), van der Laan et al. (2003) for the corresponding results in regression and likelihood cross-validation. The practical message of these results is that for quadratic (e.g., convex) uniformly bounded loss functions the cross-validation selector performs as well in risk distance as the oracle selector up to a term smaller than $C\log(K(n))/(np)$, while for non-quadratic loss functions, this last term is replaced by $C\sqrt{\log(K(n)/(np)}$. Thus, as long as the number $K(n)$ of estimators we consider is such that $\log(K(n))/(np)$ is of smaller order than the actual minimal risk distance $\min_{s\in\mathcal{A}_n} d(\hat{\psi}_s, \psi_0)$ of the candidate estimators to ψ_0, then the cross-validation selector is asymptotically equivalent (in risk distance) to the oracle selector. That is, in estimation problems which do not allow the parametric $1/\sqrt{n}$-rate of convergence, the number $K(n)$ of candidate estimators can be a polynomial power in n.

2.2 Discussion of road map.

Note that the *cross-validated D/S/A algorithm* estimating ψ_0, corresponding to the above road map, is completely defined by the following choices: the loss function; the basis functions ϕ_j defining the parameterization $\psi_{\beta,I}$ of the parameter space; and the sets of deletion, substitution, and addition moves which define the D/S/A algorithm (see Section 3). Consequently, the road map provides a unified treatment of (e.g.) multivariate prediction and density/hazard estimation based on either uncensored or censored data. Each of these problems can be dealt with according to the road map by the choice of a suitable loss function.

A number of commonly used estimation procedures chooses a particular loss function in the full data situation and follows the road map up to a degree, but depart completely from this type of loss function and the preferred estimator selection procedure when faced with the obstacle of evaluating the loss function in the presence of censoring (e.g., classification and regression trees, where one "minimizes" empirical risk by recursive binary partitioning of the covariate space, see Molinaro et al. (2003)).

We show in Section 6 that one can, and should, also adhere to the above estimation road map in censored data situations. All that is required is to replace the preferred full (uncensored) data loss function by an observed (censored) data loss function with the same expected value, i.e., with the

A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

same risk.

We also would like to note that existing regression methods for creating candidate estimators are not following the road map. In order to account for higher-order interactions among many variables (e.g., thousands of gene expression measures in microarray experiments), one needs to consider large parameter spaces. Many standard approaches either only accommodate variable main effects or are rigid in their search among candidate estimators. For example, while regression trees (e.g., CART, (Breiman et al., 1984)) and multivariate adaptive regression splines (MARS, (Friedman, 1991)) allow interactions among variables, the candidate estimators are generated according to a limited set of moves, amounting to forward selection (node splitting) followed by backward elimination (tree pruning). In neural networks the current types of algorithms do not select data adaptively the dimension of the linear-predictors and the number of linear predictors. Moreover, as we remarked before, such algorithms (e.g., neural networks, CART, MARS) just correspond to a particular type of parameterization, which are not necessarily the most suitable one for the true underlying regression, so that it is advisable to select the actual parameterization (i.e., the basis functions) with cross-validation as well.

2.3 The loss-based cross-validated D/S/A algorithm.

The estimation road map of Section 2.1 corresponds to an algorithm which maps a set of arguments into the estimator $\hat{\Psi}(P_n)$. Given these arguments, the D/S/A algorithm, which is described in Section 3, in particular tries to minimize $f_{E,s}(I)$ [equation (4)] and seeks to calculate $I_s(P_n) \equiv \text{argmin}_{I \in \mathcal{I}_s} f_{E,s}(I)$ for a given s.

In function notation, we refer to the *cross-validated D/S/A* algorithm as $CV - DSA(\cdot)$ and it takes as arguments the following: (1) the choice of loss function $L(\cdot)$; (2) the parameterization of the parameter space (choice of basis), $(I, \beta) \rightarrow \psi_{I,\beta}$; (3) a mapping from a constraint s to allowed subsets of basis functions $s \rightarrow \mathcal{I}_s$; (4) a mapping from a constraint s and a subset of basis functions I to the parameter space $B_{I,s}$ for β; (5) a set $\mathcal{A}_n$ of constraints over which to search; (6) the cross-validation scheme; (7) what moves to use (i.e. *deletions*, *substitutions*, and/or *additions*); and (8) the data or *learning set*, P_n.

Let's review these arguments in a general setting and then define the arguments in the context of polynomial regression. Refer to Section 4 for the

Collection of Biostatistics Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

specified arguments used in the simulation studies.

The first step is to define the objective function in terms of a loss function (Argument 1). If we focus on the class of basis functions obtained by using tensor products of univariate basis functions, then these univariate basis functions can be, for example, polynomials, splines, or wavelets (Argument 2). Recall that for each index set I consisting of a collection of vectors $\vec{p} = (p_1, \ldots, p_d) \in \mathbb{N}^d$, we have $\psi_{I,\beta}$ which is a function in the parameter space identified by I and β. With this, we need to define s which indexes a collection of subspaces $\mathbf{\Psi}_s \subset \mathbf{\Psi}$ over a set $\mathcal{A}_n$ (Argument 3). For example, $\mathbf{\Psi}_s = \{\psi_{I,\beta} : I \in \mathcal{I}_s, \beta \in B_{I,s} \subset B_I\}$. To be more specific, given a q-valued function $m(I) = (m_1(I), \ldots, m_q(I))$, $\mathcal{I}_s$ can be defined as $\mathcal{I}_s = \{I : m(I) \leq s\}$ where $m_1(I) \leq s_1, \ldots, m_q(I) \leq s_q$. For example, $m_1(I) = |I|$ represents the number of tensor products or the size of the index sets, $m_2(I) = \max_{\vec{p} \in I} \sum_{j=1}^{d} I(p_j \neq 0)$ represents the maximum order of interaction of tensor products (the number of non-zero components in $\vec{p}$), and $m_3(I) = \max_{\vec{p} \in I} \sum_{j=1}^{d} p_j$ represents the maximum sum of powers of tensor products, etc. Furthermore, in addition to requiring that $\psi_{I,\beta} \in \mathbf{\Psi}$, we can define constraints on β where we are restricting β such that it lies on a δ-grid, or limit its norm so that it is less than a specified value, $B_{I,s} = \{\beta \in \mathbb{R}^{|I|} : \psi_{I,\beta} \in \mathbf{\Psi}, \| \beta \| \leq \delta_s\}$ (Argument 4). Next, we decide on which sets we would like to search, $\mathcal{A}_n$ where $s \in \mathcal{A}_n$ (Argument 5).

The next step is to define the cross-validation scheme (Argument 6) to use, whereby we have been using v-fold cross-validation so you can define the level v under which to run v-fold cross-validation. Then we can define indicators identifying whether or not to use *deletion* and/or *substitution* moves. If Argument 7 is $(1, 1)$, then this corresponds to using *deletion* and *substitution* moves, which is what is done by the D/S/A algorithm. However, if we wanted to run pure forward selection, then Argument 7 is $(0, 0)$. Note that *addition* moves are always made. Finally, we identify the data to be used (Argument 8).

In the context of polynomial regression, $CV - DSA(\cdot)$ always uses the L_2-loss function, polynomial basis functions, and v-fold cross-validation, that is $CV - DSA\{$loss = L_2; basis = poly; $\ldots$; $v = \cdot$; moves = (1,1); data = P_n $\}$.

Note the distinction between $DSA(\cdot)$ and $CV - DSA(\cdot)$ where $DSA(\cdot)$ finds the *best* subset $I(P_n)$ for a given s as described in Box 1. While $CV - DSA(\cdot)$ finds the *best* subsets for the set s on the training sample, evaluates these *best* subsets on the validation sample which yields the cross-validation

Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

selector $s(P_n)$, and then finds the *best* final subset on the learning sample given $s(P_n)$.

In the next section we describe our proposed D/S/A algorithm.

3 D/S/A algorithm for minimizing over subsets of basis functions.

We propose an aggressive and flexible algorithm for generating a sequence of index sets I, according to three types of moves for the elements of I: deletions, substitutions, and additions. We refer to this general algorithm as the *Deletion/Substitution/Addition algorithm*, or *D/S/A algorithm*. The main features of this novel approach are summarized below for the case where the index sets are subsets of $\mathbb{N}^d$ (e.g., as is the case for tensor product basis functions ϕ_j such as polynomial basis functions in polynomial regression). Adaptations to histogram regression with partition-specific indicator basis functions are provided in Molinaro and van der Laan (2004). Because Section 4 implements the D/S/A algorithm by using tensor products of polynomial basis functions, it is important to briefly mention what has been said about polynomial bases. Barron and Xiao offer a description of polynomial methods when discussing the multivariate adaptive regression splines, or MARS, method of Friedman (1991, pg. 67-82). Barron and Xiao argue that polynomial methods provide a computationally faster algorithm than using spline bases, for example. They also refer to approximation results by Canuto and Quarteroni (1982) which show that polynomial approximations are adaptive to underlying smoothness, so that polynomial methods have a higher rate of convergence than that of splines for classes of smooth functions (see Barron (1989); Cox (1988); Canuto and Quarteroni (1982)).

To simplify notation, in this section we will suppress dependence of quantities on s. Let $\mathcal{I}$ denote the collection of allowed index sets; in our road map, $\mathcal{I}$ now plays the role of $\mathcal{I}_s$ for a specified subspace indexed by $s \in \mathcal{A}_n$. We assume that $s = (s_1, s_2)$, where s_1 denotes the upper bound on the size of the index sets (i.e, the number of basis functions allowed), while s_2 represents the remaining fine tuning parameters. The D/S/A algorithm below aims to calculate $I_{s_1,s_2}(P_n)$ (see (3)) for each choice of s_1, given s_2. Thus, one only has to carry out this algorithm for each value of s_2 to obtain all optimal

A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

index set $\{I_{s_1,s_2}(P_n) : s_1, s_2\}$ and thereby our collection of s-specific estimators $\hat{\Psi}_s(P_n)$, $s \in \mathcal{A}_n$. Throughout the remainder of this article, we use k to represent s_1 where $s_1 = |I|$ always and $s_2, \ldots, s_q$ can represent additional constraints on the tensor products and vector of coefficients.

The D/S/A algorithm for minimizing over index sets I is defined in terms of three functions, $DEL(I)$, $SUB(I)$, and $ADD(I)$, which map an index set $I \in \mathcal{I}$ of size k into *sets of index sets* of size $k-1$, k, and $k+1$, respectively.

Deletion/Substitution/Addition moves.
Consider index sets $I \subset \mathbb{N}^d$ and let $\mathcal{I}$ denote a collection of subsets of $\mathbb{N}^d$.
Deletion moves. Given an index set $I \in \mathcal{I}$ of size $k = |I|$, define a set $DEL(I) \subset \mathcal{I}$ of index sets of size $k-1$, by deleting individual elements of I. This results in k possible deletion moves, i.e., $|DEL(I)| = k$.
Substitution moves. Given an index set $I \in \mathcal{I}$ of size $k = |I|$, define a set $SUB(I) \subset \mathcal{I}$ of index sets of size k, by replacing individual elements $\vec{p} \in I$ by one of the $2d$ vectors created by adding or subtracting 1 to any of the d components of $\vec{p}$. That is, for each $\vec{p} \in I$, consider moves $\vec{p} \pm \vec{u}_j$, where $\vec{u}_j$ denotes the unit d-vector with one in position j and zero elsewhere, $j = 1, \ldots, d$. This results in up to $k \times (2d)$ possible substitution moves, i.e., $|SUB(I)| = k \times (2d)$. In case the allowed index sets require that each $\vec{p}$ has at most s_2 non-zero components, then we propose to add to these substitution moves the so called swap-substitution moves. The swap-substitution moves correspond to adding or subtracting the unit vectors as above, but if that results in a $\vec{p}$ with more than s_2 non-zero components, then we replace it by the s_2 vectors one obtains by setting one of the (original) non-zero components equal to zero. This augmentation of the set of substitution moves results in maximally $k \times s_2 \times (2d)$ substitution moves.
Addition moves. Given an index set $I \in \mathcal{I}$ of size $k = |I|$, define a set $ADD(I) \subset \mathcal{I}$ of index sets of size $k+1$, by adding to I an element of $SUB(I)$ or one of the d unit vectors $\vec{u}_j$, $j = 1, \ldots, d$. This results in up to $k \times (2d) + d$ (or $k \times s_2 \times (2d) + d$) possible addition moves, i.e., $|ADD(I)| = k \times (2d) + d$.

Thus the substitution moves (excluding the swap-substitution moves) can

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

be described as

$$SUB(I) \rightarrow \begin{cases} (p_1+1, p_2, p_3, \ldots, p_d) \\ (p_1, p_2+1, p_3, \ldots, p_d) \\ \vdots \\ (p_1, p_2, p_3, \ldots, p_d+1) \\ (p_1-1, p_2, p_3, \ldots, p_d) \\ (p_1, p_2-1, p_3, \ldots, p_d) \\ \vdots \\ (p_1, p_2, p_3, \ldots, p_d-1) \end{cases}$$

for each $\vec{p} \in I$, and the addition moves as adding $\vec{p}_{k+1}$ described by

$$ADD(I) = \begin{cases} (1, 0, \ldots, 0) \\ \vdots \\ (0, \ldots, 0, 1) \\ (p_1+1, p_2, p_3, \ldots, p_d) \\ \vdots \\ (p_1, p_2, p_3, \ldots, p_d+1) \\ (p_1-1, p_2, p_3, \ldots, p_d) \\ \vdots \\ (p_1, p_2, p_3, \ldots, p_d-1) \end{cases}$$

Clearly, each of these sets $DEL(I)$, $SUB(I)$, and $ADD(I)$ of possible moves can be enlarged (or modified) to enforce this algorithm to search the parameter space more aggressively, but our current results do not indicate an obvious need for this.

Next, we describe how the three basic moves of the D/S/A algorithm can be used to generate index sets $I_k(P_n)$, that seek to minimize the empirical risk function, $f_E(I)$, over all index sets I of size less than or equal to k, $k = 1, \ldots, K_n$ (Box 1). In the context of general loss functions depending on a nuisance parameter, the *empirical risk* of the I-specific estimator $\hat{\Psi}_I(P_n)$ (as defined in Section 2) is given by

$$\begin{aligned} f_E(I) &\equiv \int L(o, \hat{\Psi}_I(P_n) \mid \hat{\Upsilon}(P_n)) dP_n(o) \\ &= \frac{1}{n} \sum_{i=1}^{n} L(O_i, \hat{\Psi}_I(P_n) \mid \hat{\Upsilon}(P_n)), \end{aligned} \tag{5}$$

where $\hat{\Upsilon}(P_n)$ is an estimator of υ_0 based on the whole learning sample. $f_E : \mathcal{I} \rightarrow \mathbb{R}$ defines an empirical risk function.

CORRA
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

In the special case of the squared error loss function, with full data, the empirical risk function is simply the mean squared error (cf. residual sum of squares) for $\hat{\Psi}_I(P_n)$

$$f_E(I) = \frac{1}{n}\sum_{i=1}^{n}(Z_i - \hat{\Psi}_I(P_n)(W_i))^2.$$

Denote the best (in terms of empirical risk) index set I of size less than or equal to k, $k = 1, \ldots, K_n$, by

$$I_k^\star(P_n) \equiv \underset{\{I:|I|\leq k,\ I\in\mathcal{I}\}}{\operatorname{argmin}} f_E(I).$$

The D/S/A algorithm below returns for each k, an index set $I_k(P_n)$ that aims to approximate (or equal) $I_k^\star(P_n)$.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

Deletion/Substitution/Addition algorithm for optimizing the empirical risk function.

1. **Initialization.** Set $I_0 = \emptyset$ and $BEST(k) = \infty$, $k = 1, 2, \ldots$, where $BEST(k)$ represents the current lowest value of the objective function $f = f_E$ for index sets I of size k. Let $BEST.SET(k)$ represent the actual index sets so that $f(BEST.SET(k)) = BEST(k)$.

2. **Algorithm (*).** Let $k = |I_0|$. Find an optimal updated index set I^- of size $k-1$, among all allowed **deletion** moves: $I^- \equiv \text{argmin}_{I \in DEL(I_0)} f(I)$. If $f(I^-) < BEST(k-1)$, then set $I_0 = I^-$, $BEST(k-1) = f(I^-)$, $BEST.SET(k-1) = I_0$, and go back to (*).

   Otherwise, find an optimal updated index set $I^=$ of the same size k as I_0, among all allowed **substitution** moves: $I^= \equiv \text{argmin}_{I \in SUB(I_0)} f(I)$. If this update improves on I_0, that is, $f(I^=) < f(I_0)$, then set $I_0 = I^=$, $BEST(k) = f(I^=)$, $BEST.SET(k) = I_0$, and go back to (*).

   Otherwise, find an optimal updated index set I^+ of size $k+1$, among all allowed **addition** moves: $I^+ \equiv \text{argmin}_{I \in ADD(I_0)} f(I)$. Set $I_0 = I^+$. If this update improves on I_0, that is, $f(I^+) < f(I_0)$, then set $BEST(k+1) = f(I^+)$, and $BEST.SET(k+1) = I_0$. Go back to (*).

3. **Stopping rule.** Run the algorithm until the current index set size $k = |I_0|$ is larger than a user-supplied max. size or until $f(I^+) - f(I_0) < \Delta$ for a user-specified $\Delta > 0$. Denote the last set I by $I_{\text{final}}(P_n)$.

Note that the D/S/A algorithm is such that $BEST(k)$ is decreasing in k, since addition moves only occur when they result in a decrease in risk over the current index set size. Thus, the best subset of size k is also the best subset of size less than or equal to k. We also note that this algorithm gives priority to moves which make the fit smaller, and it avoids getting trapped in a local maximum by always carrying out the addition move.

Unlike previously proposed forward/backward selection approaches, this D/S/A algorithm performs an extensive search of the parameter space, truly aimed at minimizing the empirical risk function over all index sets of a given size.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

3.1 Simple example to illustrate D/S/A algorithm.

Consider the regression setting so that $L(O,\psi) = (Y - \psi(W))^2$, and suppose that we parameterize each allowed regression function as linear combinations of tensor products of the polynomial powers. Suppose that $W = (W_1, \ldots, W_4)$ (i.e., $d = 4$) and that the current model (i.e., I_0) in the D/S/A algorithm is given by $Y = W_1W_2W_3 + W_2W_4^5$. Note that the current size is $k = 2$, the corresponding indices are $\vec{p}_1 = (1,1,1,0)$, $\vec{p}_2 = (0,1,0,5)$, and $I_0 = \{\vec{p}_1, \vec{p}_2\}$.

A *deletion* move simply means removing one of the terms of the current model and fitting a model of size $k-1$. Thus, the deletions set, $DEL(I_0)$, contains two index sets of size $k = 1$

$$DEL(I_0) = \Big\{\{\vec{p}_1\}, \{\vec{p}_2\}\Big\} = \Big\{\{(1,1,1,0)\}, \{(0,1,0,5)\}\Big\}.$$

The *substitution* moves involve replacing the j^{th} term for $j = 1, \ldots, k$ with a new term, keeping the size of the model fixed at k. The possible substitution moves are given by:

$$SUB(I_0) = \begin{cases} W_1^2W_2W_3 + W_2W_4^5 & \vec{p}_1 = (2,1,1,0) \\ W_1W_2^2W_3 + W_2W_4^5 & \vec{p}_1 = (1,2,1,0) \\ W_1W_2W_3^2 + W_2W_4^5 & \vec{p}_1 = (1,1,2,0) \\ W_1W_2W_3W_4 + W_2W_4^5 & \vec{p}_1 = (1,1,1,1) \\ W_2W_3 + W_2W_4^5 & \vec{p}_1 = (0,1,1,0) \\ W_1W_3 + W_2W_4^5 & \vec{p}_1 = (1,0,1,0) \\ W_1W_2 + W_2W_4^5 & \vec{p}_1 = (1,1,0,0) \\ W_1W_2W_4^5 + W_1W_2W_3 & \vec{p}_2 = (1,1,0,5) \\ W_2^2W_4^5 + W_1W_2W_3 & \vec{p}_2 = (0,2,0,5) \\ W_2W_3W_4^5 + W_1W_2W_3 & \vec{p}_2 = (0,1,1,5) \\ W_2W_4^6 + W_1W_2W_3 & \vec{p}_2 = (0,1,0,6) \\ W_4^5 + W_1W_2W_3 & \vec{p}_2 = (0,0,0,5) \\ W_2W_4^4 + W_1W_2W_3 & \vec{p}_2 = (0,1,0,4) \end{cases}$$

We also note that, if the total number of terms in the tensor products is bounded by $s_2 = 3$, then the substitution move which would not be allowed, $W_1W_2W_3W_4 + W_2W_4^5$, would be replaced by these swap moves: $W_2W_3W_4 + W_2W_4^5$, $W_1W_3W_4 + W_2W_4^5$, $W_1W_2W_4 + W_2W_4^5$.

A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

If none of these substitution moves improve RSS, then the D/S/A algorithm finds the best fit among the following *addition* moves:

$$ADD(I_0) = \begin{cases} W_1 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (1,0,0,0) \\ W_2 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,1,0,0) \\ W_3 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,0,1,0) \\ W_4 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,0,0,1) \\ W_1^2W_2W_3 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (2,1,1,0) \\ W_1W_2^2W_3 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (1,2,1,0) \\ W_1W_2W_3^2 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (1,1,2,0) \\ W_1W_2W_3W_4 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (1,1,1,1) \\ W_2W_3 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,1,1,0) \\ W_1W_3 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (1,0,1,0) \\ W_1W_2 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (1,1,0,0) \\ W_1W_2W_4^5 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (1,1,0,5) \\ W_2^2W_4^5 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,2,0,5) \\ W_2W_3W_4^5 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,1,1,5) \\ W_2W_4^6 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,1,0,6) \\ W_4^5 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,0,0,5) \\ W_2W_4^4 + W_1W_2W_3 + W_2W_4^5 & \vec{p}_3 = (0,1,0,4) \end{cases}$$

4 Simulations.

A variety of simulations were conducted to see how the algorithm performs in different settings. The D/S/A algorithm first is implemented without constraints (brakes), and using the notation of Section 2.3, we are running $DSA\{\text{loss} = L_2;\ \text{basis} = \text{poly};\ m(I) = \{m_1(I)\};\ \beta_{I,s} = \text{everything};\ k = \infty;\ v = \text{NULL};\ \text{moves} = (1,1);\ \text{data} = P_\infty\}$. In all of the simulations, we are not placing additional constraints on β. $m_1(I) = |I|$ and we are not restricting the number of tensor products ($k = \infty$). We therefore are not concerned with using cross-validation to select k. We are running $DSA(\cdot)$ until we reach a k that gives a minimal residual sum of squared errors (RSS). Next, the cross-validated D/S/A algorithm is implemented with a cross-validated constraint placed on the number of tensor products in the regression, i.e., selecting k via cross-validation. Let *DSA1-CV* be defined by $CV - DSA\{\text{loss} = L_2;\ \text{basis} = \text{poly};\ m(I) = \{m_1(I)\};\ \beta_{I,s} = \text{everything};\ k = \{1,\ldots,K_n\};\ v = \{2,5,10\};$

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

moves = (1,1); data = P_n }. This means that we are running $CV - DSA(\cdot)$ under 2, 5, and/or 10-fold cross-validation where we are using cross-validation to select the number of tensor products ($m_1(I)$). β is unrestricted as usual. The algorithm will yield candidate estimators of size $k = 1, 2, \ldots, K_n$ based on the training sample. Then we find amongst the estimators of size k the one that gives the minimal cross-validated risk over the v validation samples. Using the $\hat{k}$ obtained via cross-validation, find the final estimator of size $\hat{k}$ on the learning sample. Note that the specific preset options for K_n and v will be given for each individual simulation study. The cross-validated D/S/A algorithm (*DSA1-CV*) is then compared to: the R function `stepAIC()`, forward selection with cross-validation (*fscv*), and logic regression. Finally, the algorithm which places a second brake on the complexity of each tensor product ($m_2(I) = \max_{\vec{p} \in I} \sum_{j=1}^{d} \mathrm{I}(p_j \neq 0)$; $m_2(I) \leq s_2$) and thereby incorporates the *swap-substitution* moves is implemented (*DSA2-CV*) and used in the logic regression comparisons. Let *DSA2-CV* be defined by $CV - DSA\{$loss = L_2; basis = poly; $m(I) = \{m_1(I), m_2(I)\}$; $\beta_{I,s}$ = everything; $s_1 = \{1, \ldots, K_n\}$, $s_2 = \{1, \ldots\}$; $v = \{2, 5, 10\}$; moves = (1,1); data = P_n }. Selecting s_2 via cross-validation is used in Section 5.

In each of the simulations, an $n \times d$ covariate matrix, W, is generated from a given probability distribution, e.g. normal, uniform, Bernoulli. The true mean linear polynomial regression model, $E(Y|W)$, is either manually or randomly generated with the details described for each simulation. The outcome Y is then generated from the true mean linear polynomial regression model with no noise or a Gaussian noise with mean 0 and standard deviation σ. The D/S/A algorithm is then used to minimize $f_{E,s}(I)$, and the procedure may be repeated a number of times.

4.1 Implementation without constraints.

The first set of simulations explore the performance of the D/S/A algorithm itself. Therefore, cross-validation is not yet employed, and the D/S/A algorithm is run on the *learning* set without using cross-validation to select the size k.

The purpose of these simulations is to establish to what degree the D/S/A algorithm is truly capable of finding the global minimum (i.e., the optimal predictor $W \to \psi_0(W) = E_0(Y|W)$) when n is large enough. In the following simulations, the true regression model is randomly generated where the first set of simulations, Table 1, uses a relatively simpler true model compared to

Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

the second set of simulations, Table 2.

4.1.1 Simple True Regression Models.

The protocol of this simulation can briefly be described as follows:

1. Generate the covariate matrix $W_{n,d}$ of n i.i.d. observations of d variables W_i, $i = 1, \ldots, d$ using either a uniformly distributed distribution between 0 and 1 or a Bernoulli distribution with a fixed or random probability of event.

2. Randomly define the true polynomial regression model, $E(Y|W)$, such that the number of terms in the regression along with the variables used and the degree with which they enter the regression model is random. Specifically, $E[Y|W]$, is a sum of tensor-product polynomial basis functions: $Y = \sum_{s=1}^{size} \beta_s \prod_{j=1}^{d} W_j^{p_s(j)} + \varepsilon$, $E(\varepsilon|W) = 0$. $p_s(j)$ is the j^{th} element of $\vec{p}_s$ and is used to identify each basis function. The true regression is randomly generated as follows:

$$\begin{aligned} &size \sim \mathcal{U}\{1, \ldots, 5\} \\ &\sum_{j=1}^{d} p_s(j) \sim \mathcal{U}\{1, \ldots, 5\} \\ &\vec{p}_s \sim \text{Multinomial}(\textstyle\sum_{j=1}^{d} p_s(j), d, (\frac{1}{d}, \ldots, \frac{1}{d})) \end{aligned}$$

   An example of a true regression used in a particular simulation (see Table 1) when $d = 5$ is:

$$E[Y|W] = W_0 W_1^2 W_2^2 + W_0 W_1 W_2^2 W_3 + W_2^3 + W_4^4$$

3. Generate the outcome Y using the previously randomly generated true regression model with no noise, $\varepsilon = 0$.

4. Use the D/S/A algorithm to minimize over I the residual sum of squared errors for the I-specific least squares estimator.

5. Repeat this procedure up to nreps = 1000 times. Report the mean of the proportion of correctly fitted terms among the terms in the true regression or the sensitivity, the mean of the proportion of correctly

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

fitted terms among the terms in the fitted regression model or the specificity, and the mean residual sum of squares of the regression fit selected by $DSA(\cdot)$ over the nreps iterations, $\overline{\text{RSS}}$.

Using the notation of Section 2.3, we are running $DSA\{\text{loss} = L_2; \text{basis} = \text{poly}; m(I) = \{m_1(I)\}; \beta_{I,s} = \text{everything}; k = \infty; v = \text{NULL}; \text{moves} = (1,1); \text{data} = P_\infty\}$ in this initial set of simulations. Some of the numerical results obtained from this simulation are given in Table 1. Again, the idea of these simulations is to see if the D/S/A algorithm can get to the truth. The results indicate that in each case the algorithm succeeded in minimizing $f_{E,s}(I)$, $\overline{RSS} = 0$ (col 7, table 1). Both sensitivity and specificity of $DSA(\cdot)$ is 100% (cols 5-6, table 1), indicating that the algorithm is successful in fitting true *simple* regressions in the case of zero error which corresponds to choosing a very large sample size. These results are encouraging and led to further exploration of the algorithm's capabilities.

W	n	d	$nreps$	$sens$	$spec$	$\overline{RSS}$
$\mathcal{U}(0,1)$	1000	5	1000	100%	100%	0.0000
$\mathcal{U}(0,1)$	1000	100	500	100%	100%	0.0000
Bernoulli(random)	1000	5	100	100%	100%	0.0000
Bernoulli(random)	2000	10	100	100%	100%	0.0000
Bernoulli(random)	1000	25	100	100%	100%	0.0000
Bernoulli(0.6)	500	5	100	100%	100%	0.0000
Bernoulli(0.6)	500	25	100	100%	100%	0.0000

Table 1: Zero error added to the true simple model. $DSA\{\text{loss} = L_2; \text{basis} = \text{poly}; m(I) = \{m_1(I)\}; \beta_{I,s} = \text{everything}; k = \infty; v = \text{NULL}; \text{moves} = (1,1); \text{data} = P_\infty\}$. *nreps*: number of repetitions, *sens*: average sensitivity across the number of repetitions, *spec*: average specificity across the number of repetitions, $\overline{RSS}$: average residual sum of square of fitted model across the number of repetitions.

4.1.2 Not as Simple True Regression Models.

To convince ourselves that the estimator is truly consistent non-parametrically, this set of simulations involves examples where the true model is more complicated. The simulation protocol is similar, though not identical, to the

Collection of Biostatistics Research Archive

Hosted by The Berkeley Electronic Press

previous one: $W_{n,d}$ is generated from either a uniform or normal probability distribution, with $d = 100$ in all cases. The true regression model again is randomly generated, but now we have increased both the size and the allowed complexity of each tensor product such that:

$$size \sim \mathcal{U}\{1, \ldots, 10\}$$
$$\sum_{j=1}^{d} p_s(j) \sim \mathcal{U}\{1, \ldots, 20\}$$

The outcome Y again is generated with no added noise, and the procedure is run only once with $RSS \leq 0.000001$ used as a stopping criterion.

The following four true regression models were generated:

$$E_1[Y|W] = W_1W_{12}W_{13}^2W_{22}W_{24}W_{54}W_{79}W_{83}W_{95}$$
$$+ W_{15}W_{18}W_{37}W_{42}W_{68} + W_6W_{22}W_{33}^3W_{40}W_{58}W_{75}W_{82}W_{87}$$
$$+ W_{15}W_{31}$$

$$E_2[Y|W] = W_7W_{25}W_{31}W_{59}W_{63}W_{68}W_{70}W_{83}W_{88}W_{98}$$
$$+ W_0W_{32}W_{47}W_{54}W_{66}W_{72}W_{73}W_{77} + W_{82} + W_7W_{49}W_{55}W_{73}W_{80}$$
$$+ W_{33}W_{40} + W_{18}W_{21}W_{40}W_{56}W_{59}W_{71}W_{91}$$
$$+ W_9W_{13}W_{18}W_{20}W_{41}W_{53}W_{69}W_{95}$$
$$+ W_3W_{38}W_{78}W_{96} + W_0W_{20}W_{64}W_{88}W_{91}W_{96}$$
$$+ W_2W_6W_{16}W_{37}W_{45}W_{46}W_{61}W_{68}W_{91}W_{95}$$

$$E_3[Y|W] = W_0W_1^2W_2W_3^2W_4 \cdots W_{98}W_{99}^2$$

$$E_4[Y|W] = W_0W_1^2W_4^4W_{99}^{10} + W_{45}$$
$$+ W_2^2W_8W_{14}W_{20}W_{22}W_{29}W_{36}W_{39}W_{41}W_{44}W_{48}W_{56}W_{62}W_{63}W_{65}W_{87}$$
$$+ W_{27}W_{48}W_{63}W_{77}W_{78}W_{93}W_{94} + W_{71}$$
$$+ W_{12}W_{18}W_{22}W_{44}W_{50}W_{55}W_{57}W_{64}W_{73}^2W_{80}W_{83}W_{93}W_{94}W_{96}$$
$$+ W_{69}W_{91} + W_2W_4W_{22}W_{23}W_{28}W_{36}W_{53}W_{79}W_{88} + W_{48}W_{70}W_{82}W_{97}$$
$$+ W_3W_{24}W_{29}W_{54}W_{64}W_{80}$$

The true model is selected in the first three cases (col 5, rows 1-3, table 2), but in the third case the stopping criterion is reached before the algorithm has the chance to fit the correct 10 tensor products. It did manage to fit 80% of the true terms (col 5, row 3, table 2) but achieved an RSS equal to that

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

of our stopping criterion with a fitted model of 11 terms.

$E[Y\|W]$	W	n	d	*sens*	*spec*	*RSS*
$E_1[Y\|W]$	$\mathcal{U}(0,1)$	1000	100	100%	100%	0.000000
$E_2[Y\|W]$	$\mathcal{U}(0,1)$	1000	100	100%	100%	0.000000
$E_3[Y\|W]$	$\mathcal{N}(1.1,0.1)$	5000	100	100%	100%	0.000000
$E_4[Y\|W]$	$\mathcal{U}(0,1)$	1000	100	80%	73%	0.000001

Table 2: Zero error added to the true complex model. $DSA\{\text{loss} = L_2$; basis = poly; $m(I) = \{m_1(I)\}$; $\beta_{I,s} =$ everything; $k = \infty$; $v =$ NULL; moves = (1,1); data = $P_\infty\}$. *sens*: sensitivity, *spec*: specificity, RSS: residual sum of square of fitted model.

The next step is to see what happens when some noise is added to these regression models. We used a normal distribution with mean 1 and standard deviation 0.5 to generate $W_{n,d}$ for $E_5[Y|W]$ and $E_6[Y|W]$. The outcome Y is generated from the randomly chosen true regression model with zero error or Gaussian error with mean 0 and standard deviation 1. These too were run only once.

The following two models were generated, first with $\varepsilon = 0$ and then with $\varepsilon \sim \mathcal{N}(0,1)$.

$$E_5[Y|W] = W_0W_1^2W_2^2 + W_0W_1W_2^2W_3 + W_2^3 + W_4^4$$

$$E_6[Y|W] = W_0 + W_0W_{49}W_{99} + W_{24}^5 + W_{17}W_{30}W_{53}W_{62}W_{78}W_{88}$$

Table 3 displays the results of this simulation which compares two models with zero error and a Gaussian error. The truth was identified in all cases where $\varepsilon = 0$ or $\varepsilon \sim \mathcal{N}(0,1)$. The number of moves the algorithm needed to make in order to converge are displayed as well for this simulation. Based on Table 3, a large number of covariates does not affect the convergence rate since the number of moves performed when $d = 100$ is less than the number of moves performed when $d = 5$.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

$E[Y\|W]$	n	d	*sens*	*spec*	RSS_n	RSS_0	moves	subs	adds	dels
$E_5[Y\|W]$	1000	5	100%	100%	0.0000	0.0000	30	22	6	2
$E_5[Y\|W]^*$	1000	5	100%	80%	1.074	1.080	30	23	6	1
$E_6[Y\|W]$	1000	100	100%	100%	0.0000	0.0000	21	17	4	0
$E_6[Y\|W]^*$	1000	100	100%	100%	0.9576	0.9572	21	17	4	0

Table 3: Comparing $\varepsilon = 0$ and $\varepsilon \sim \mathcal{N}(0,1)^*$. $DSA\{\text{loss} = L_2\text{; basis} = \text{poly; } m(I) = \{m_1(I)\}\text{; } \beta_{I,s} = \text{everything; } k = \infty\text{; } v = \text{NULL; moves} = (1,1)\text{; data} = P_\infty\}$. *sens*: sensitivity, *spec*: specificity, RSS_n: $RSS/(n-b)$ represents the estimate of the variance of the error where b is the number of independent variables in fitted model, RSS_0: true variance of the error, *moves*: number of moves made by the algorithm, *subs*: number of substitution moves made, *adds*: number of addition moves made, *dels*: number of deletion moves made, *: indicates the model for which $\varepsilon \sim \mathcal{N}(0,1)$.

4.2 D/S/A algorithm with cross-validated size versus the `stepAIC()` function in R.

The next set of simulations address the performance of cross-validation in making sure that the algorithm does not select too many variables, and thereby over-fits, by comparing it to the R function `stepAIC()`.

Before describing the simulation protocol, let's review how cross-validation fits into the D/S/A algorithm. $CV - DSA(\cdot)$ yields a sequence of models of size $k = 1$, another sequence of models of size $k = 2$, ..., and finally a sequence of models of size $k = K_n$. Given the "best" model of *each* size for sizes ranging from $k = 1$ to $k = K_n$, calculate the cross-validated risk for each model, and then choose the best (i.e. the model with the minimal cross-validated risk) size via cross-validation. We are using v-fold cross-validation to split the sample. Recall that we defined a random vector $B_n \in \{0,1\}^n$ for splitting the sample into a validation and a training sample:

$$B_n(i) = \begin{cases} 0 & \text{if } i^{\text{th}} \text{ observation is in the training sample} \\ 1 & \text{if } i^{\text{th}} \text{ observation is in the validation sample} \end{cases}$$

Let P^0_{n,B_n}, P^1_{n,B_n} be the empirical distributions of the training and validation sample, respectively.

- Run the algorithm on the training sample and obtain $\hat{\Psi}_k(P^0_{n,B_n})$ for

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

$k = 1, \ldots, K_n$ (Note: $k = |I|$).

- Given $\hat{\Psi}_k(P^0_{n,B_n})$, compute the risk on the validation sample and choose $\hat{k}$ such that:

$$\hat{k} \equiv \text{argmin}_k E_{B_n} \int L(O, \hat{\Psi}_k(P^0_{n,B_n}))dP^1_{n,B_n}(O)$$

- For example, in these regression simulations, $\hat{k} \equiv \sum_{i:B_n(i)=1} \{Y_i - \hat{\Psi}_k(P^0_{n,B_n})\}^2$.

- Our final estimator of ψ_0 is given by $\psi_n = \hat{\Psi}(P_n) \equiv \Psi_{\hat{k}}(P_n)$.

With that in mind, the simulation protocol is:

1. Generate the covariate matrix $W_{n,d}$ of n i.i.d. observations of d variables W_i, $i = 1, \ldots, d$ from a uniformly distributed distribution between 1 and 10.

2. Manually generate the following three true regression models:
   $E_1[Y|W] = W_1 + W_2^2$

   $E_2[Y|W] = W_1 W_3$

   $E_3[Y|W] = W_1 W_3 + W_5^2 + W_7 W_{10}$

3. Generate the outcome Y using the previously randomly generated true mean regression model with Gaussian error with mean 0 and standard deviation 1.

4. Let *DSA1-CV* be defined by $CV - DSA\{$loss $= L_2$; basis = poly; $m(I) = \{m_1(I)\}$; $\beta_{I,s}$ = everything; $k = \{1, \ldots, 10\}$; $v = 2$; moves = (1,1); data $= P_n$ $\}$. The procedure is run once with *DSA1-CV* and `stepAIC`. Report the final size of the fitted model chosen by each method, $\hat{k}$, and an estimate of the true risk, $\hat{r}$.

The D/S/A algorithm creates variables data-adaptively and therefore does not require enumeration of all potential variables. `StepAIC` does require enumeration of all variables. To compare the two black-box algorithms

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

(data → predictor), we enumerated all main terms, squared terms, and pairwise interactions. This is a preliminary simulation to compare the D/S/A algorithm with a cross-validated constraint on the size of the model with the forward selection algorithm (enumerating all terms) using AIC to select the size of the model. (In Section 4.3, we will compare our *DSA1-CV* algorithm to forward selection with cross-validation.) For this simulation, we are interested in whether or not each method fits the true model and the true risk of the selected model. The true risk is estimated by setting aside a large sample of independent observations, a *test set*, and calculating the risk based on the fitted model on this set of observations. In this particular case, a test set of size 20,000 was used to estimate the true risk.

In the first simulation (row 1, table 4), both our method and `stepAIC` selected the exact true model. However, in the next two simulations (rows 2-3, table 4), our method fitted the truth exactly while `stepAIC` heavily over-fitted the model (col 4, table 4). The over-fitting did not hurt the risk estimate because the estimated risk from the fitted model produced by `stepAIC` is nearly the same as the estimated risk given by our method's fitted model.

$E[Y\|W]$	n	d	$\hat{k}_{AIC}$	$\hat{k}_{DSA1-CV}$	$\hat{r}_{AIC}$	$\hat{r}_{DSA1-CV}$
$E_1[Y\|W]$	5000	3	2	2	0.9963	0.9963
$E_2[Y\|W]$	5000	10	19	1	0.9995	0.9932
$E_3[Y\|W]$	5000	10	22	3	1.0174	1.0106

Table 4: Comparing `stepAIC` to DSA-CV algorithm with cross-validated constraint on size under 2-fold cross-validation. $\hat{k}$: size of the final fitted model for each method, $\hat{r}$: estimate of the true risk, based on 20,000 independent observations, of the final model chosen by both methods.

4.3 Comparison to forward selection with cross-validation.

These simulation studies (Tables 5 - 7) compare the D/S/A algorithm, such that *DSA1-CV* is defined by $CV - DSA\{$loss $= L_2$; basis = poly; $m(I) = \{m_1(I)\}$; $\beta_{I,s}$ = everything; $k = \{1, \ldots, 10\}$; $v = \{2, 5, 10\}$; moves = (1,1); data $= P_n$ $\}$, to a type of forward selection with cross-validation (fscv) algorithm. The forward selection algorithm makes all the same *addition* moves as

Collection of Biostatistics Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

our method but does not carry out the deletion and substitution moves. Let *fscv* be identified by $CV - DSA\{\text{loss} = L_2;\ \text{basis} = \text{poly};\ m(I) = \{m_1(I)\};\ \beta_{I,s} = \text{everything};\ k = \{1, \ldots, 10\};\ v = \{2, 5, 10\};\ \text{moves} = (0,0);\ \text{data} = P_n\}$.

In the first simulation, Table 5, the true model is $y = \sum_{j=1}^{5} c_j w^{2j} + \varepsilon$, where $w \sim U(-1, 1)$, ε is normal with mean 0 and standard deviation 0.25, and $c_j = 1/j^2$. This is a difficult regression problem because it involves a covariate which is symmetric and narrowly spread about zero and coefficients in the range of 0.04 to 1, rendering a relatively low signal-to-noise ratio.

The risks (col 4, table 5) are roughly similar for both methods. For $n = 500$ and $n = 1000$, *fscv* has a better risk than *DSA1-CV*. Both methods have poor sensitivity and can fit a model different from the truth but with a low risk estimate. *DSA1-CV* has greater specificity in this particular simulation.

The second simulation, Table 6, has the same true model as that of the previous simulation, but $w \sim U(-3, 3)$ and ε is normal with mean 0 and standard deviation 1. This is an extreme simulation in the sense that when $w = 3$, $c_5 w^{10}$ gives a relatively large value for y. A plot of the data would show occasional extreme values for y where $y \in [-4 \times 10^{10}, 4 \times 10^{10}]$ with the majority centered on 0.

fscv yields models that are too big. It always picks a model of size 10, where 10 is the preset maximal allowed terms for all methods. Because it is picking such a large model, it has a sensitivity of 100%, but a specificity of 50%. And naturally, a large model will have a low risk estimate. If the maximal allowed number of terms was set to 5 instead of 10 (i.e., if the argument k was set to {1,. . . ,5} instead of {1,. . . ,10}), then *fscv*, for example when $n = 1000$ and $v = 2$, yields a mean risk estimate of 17841, an average size of 4.5, 40% sensitivity, and 45% specificity. The D/S/A algorithm is more aggressive than the fscv algorithm, and it is trying to fit the extreme observations which occur on occasion. *DSA1-CV* always tries to fit a model of about size 5 (col 6, table 6), where the truth contains five terms. But when it tries to fit extreme values, it results in a poor estimator. The true risk estimates are very spread out (col 5, table 6). For example, the median risk estimate when $n = 1000$ and $v = 10$ is 4.051 while the mean is 765. The aggressiveness of the D/S/A algorithm leads to variable estimators when it is trying to account for outliers, or extreme observations, which occur in this particular simulation.

The true model, for the third simulation (Table 7), is $y = 4w + 3w^3 - 2w^5 + \varepsilon$, where $w \sim U(1, 5)$, ε is normal with mean 0 and standard deviation

UCBerkeley
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

1, and $y \in (-6000, 8)$. The fscv algorithm, naturally, picks a model with about 5 or more terms, not having the deletion or substitution step to get rid of terms involving even powers, and thus has a sensitivity of 100% in all cases. *DSA1-CV* picks a smaller model on average with a higher specificity as expected. The risk estimates are approximately the same for both methods. The D/S/A algorithm seems to be more efficient for smaller sample sizes than the fscv algorithm. The implications of these simulations will be described further in Section 7.

4.4 Logic Regression.

Logic Regression (Ruczinski et al., 2003), like the D/S/A algorithm, is an adaptive regression methodology that attempts to construct predictors. However, the goal of Logic Regression is to find predictors that are Boolean (logical) expressions, and thus is applied when the covariates in the data to be analyzed are primarily binary. The D/S/A algorithm can handle any combination of continuous and discrete covariates. It is important to compare the two methods when applied to binary variables, and this simulation is an initial attempt at comparing the two. Logic Regression uses a cross-validated constraint on the complexity of each tree, which corresponds to the complexity of our tensor products (implemented by *DSA2-CV*, Tables 8 - 10). Thus, Logic Regression is compared to two implementations of the D/S/A algorithm, (*DSA1-CV* and *DSA2-CV*) referred to as dsa1 and dsa2, respectively, in Tables 8 - 10 where *DSA1-CV* is defined by $CV - DSA\{\text{loss} = L_2;\ \text{basis} = \text{poly};\ m(I) = \{m_1(I)\};\ \beta_{I,s} = \text{everything};\ k = \{1, \ldots, K_n\};\ v = \{2, 5, 10\};\ \text{moves} = (1,1);\ \text{data} = P_n\ \}$ and *DSA2-CV* is defined by $CV - DSA\{\text{loss} = L_2;\ \text{basis} = \text{poly};\ m(I) = \{m_1(I), m_2(I)\};\ \beta_{I,s} = \text{everything};\ s_1 = \{1, \ldots, K_n\},\ s_2 = \{1, \ldots\};\ v = \{2, 5, 10\};\ \text{moves} = (1,1);\ \text{data} = P_n\ \}$, where $m_2(I) = \max_{\vec{p} \in I} \sum_{j=1}^{d} I(p_j \neq 0)$. The first implementation was used in the previous simulations (Tables 4 - 7), and the second implementation uses cross-validation to select the number of tensor products and places a brake on the complexity of each tensor product thereby involving the swap moves (i.e., it is limiting the order of interactions to be no greater than a specified value, s_2).

When running Logic Regression, the preset maximum number of allowed trees matched the number of terms in the true model. The results of both Logic Regression and DSA-CV depend on the fine tuning parameters such as number of folds, number of trees or maximum number of tensor products, and

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Table 5: *Simulation study 1* **RISK COMPARISON**. Full data simulated from $y = \sum_{j=1}^{5} c_j w^{2j} + er$, where $w \sim U(-1,1)$, $er \sim N(0,\sigma)$, $\sigma = 0.25$, and $c_j = 1/j^2$. Candidate estimator was chosen over 50 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2) for all algorithms (col 3). The results (cols 4-9) in the table are based on an independent test sample of $n = 10000$. col 4 is the average of the 50 risks for each method, col 5 is the standard deviation of the risks over the 50 reps, col 6 is the average size (number of basis functions), col 7 is the sensitivity, col 8 is the specificity, and col 9 is the ratio of averaged risks (col 4) − optimal risk, (ours/fscv).

Sample			50 Repetitions					
Size	$v-fold$	Method	mean	std dev	avg size	sens	spec	ratio
250	2	ours	.0648	.002	2.88	35%	73%	1
		fscv	.0651	.001	4.16	38%	47%	.881
	5	ours	.0651	.002	3.58	39%	64%	1
		fscv	.0649	.001	4.04	38%	48%	1.055
	10	ours	.0649	.002	3.88	41%	62%	1
		fscv	.0649	.001	4.18	40%	49%	.987
500	2	ours	.0642	.002	3.54	40%	69%	1
		fscv	.0640	.001	4.34	41%	48%	1.106
	5	ours	.0640	.002	3.88	43%	64%	1
		fscv	.0638	.001	4.28	41%	48%	1.135
	10	ours	.0639	.003	4.28	48%	63%	1
		fscv	.0638	.001	4.32	42%	49%	1.077
1000	2	ours	.0636	.001	3.92	42%	61%	1
		fscv	.0634	.001	5.42	52%	48%	1.232
	5	ours	.0637	.001	4.06	42%	58%	1
		fscv	.0633	.001	4.84	46%	48%	1.415
	10	ours	.0636	.001	3.92	40%	59%	1
		fscv	.0634	.001	5.18	50%	49%	1.291

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

Table 6: *Simulation study 2* **RISK COMPARISON**. Full data simulated from $y = \sum_{j=1}^{5} c_j w^{2j} + er$, where $w \sim U(-3, 3)$, $er \sim N(0, 1)$, and $c_j = 1/j^2$. Candidate estimator was chosen over 50 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2) for all algorithms(col 3). The results (cols 4-9) in the table are based on an independent test sample of $n = 10000$. col 4 is the average of the 50 risks for each method, col 5 is the standard deviation of the risks over the 50 reps, col 6 is the average size (number of basis functions), col 7 is the sensitivity, col 8 is the specificity, and col 9 is the ratio of averaged risks (col 4) $-$ optimal risk, (ours/fscv).

Sample			50 Repetitions					
Size	$v-fold$	Method	mean	std dev	avg size	sens	spec	ratio
250	2	ours	11.266	22	4.5	52%	60%	1
		fscv	1.052	.031	10.0	100%	50%	197
	5	ours	27.898	125	4.82	53%	56%	1
		fscv	1.052	.031	10.0	100%	50%	517
	10	ours	9.041	21	5.0	55%	55%	1
		fscv	1.052	.031	10.0	100%	50%	155
500	2	ours	5.209	8	4.64	60%	64%	1
		fscv	1.027	.012	10.0	100%	50%	156
	5	ours	362.2	2525	4.48	60%	66%	1
		fscv	1.027	.012	10.0	100%	50%	13377
	10	ours	362.3	2525	4.6	62%	66%	1
		fscv	1.027	.012	10.0	100%	50%	13381
1000	2	ours	376.5	2499	4.74	47%	50%	1
		fscv	1.015	.005	10.0	100%	50%	25033
	5	ours	1104.9	4240	4.76	48%	51%	1
		fscv	1.015	.005	10.0	100%	50%	73593
	10	ours	765.4	3502	5.06	51%	50%	1
		fscv	1.015	.005	10.0	100%	50%	50960

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Table 7: *Simulation study 3* **RISK COMPARISON**. Full data simulated from $y = 4w+3w^3-2w^5+er$, where $w \sim U(1,5)$ and $er \sim N(0,1)$. Candidate estimator was chosen over 50 repetitions of three sample sizes (col 1), three v-fold cross-validations (col 2) for all algorithms (col 3). The results (cols 4-9) in the table are based on an independent test sample of $n = 10000$. col 4 is the average of the 50 risks for each method, col 5 is the standard deviation of the risks over the 50 reps, col 6 is the average size (number of basis functions), col 7 is the sensitivity, col 8 is the specificity, and col 9 is the ratio of averaged risks (col 4) − optimal risk, (ours/fscv).

Sample			50 Repetitions					
Size	$v-fold$	Method	mean	std dev	avg size	sens	spec	ratio
250	2	ours	1.043	.018	3.62	83%	71%	1
		fscv	1.045	.019	5.36	100%	57%	.950
	5	ours	1.044	.018	3.64	82%	71%	1
		fscv	1.046	.019	5.46	100%	56%	.960
	10	ours	1.043	.018	3.62	82%	72%	1
		fscv	1.046	.019	5.52	100%	55%	.939
500	2	ours	1.031	.006	3.24	91%	86%	1
		fscv	1.031	.007	5.28	100%	57%	.980
	5	ours	1.030	.006	3.30	89%	84%	1
		fscv	1.031	.007	5.30	100%	57%	.965
	10	ours	1.030	.006	3.34	89%	83%	1
		fscv	1.031	.007	5.34	100%	57%	.967
1000	2	ours	1.026	.005	3.54	85%	75%	1
		fscv	1.026	.005	5.28	100%	57%	1.014
	5	ours	1.027	.005	3.46	84%	75%	1
		fscv	1.026	.005	5.18	100%	58%	1.023
	10	ours	1.026	.005	3.44	85%	77%	1
		fscv	1.026	.005	5.28	100%	57%	1.015

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

number of leaves or tensor product complexity measure. These simulations show that the D/S/A algorithm is competitive with Logic Regression since the risk ratios are roughly one in each simulated setting.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Table 8: *Simulation study 1* **Logic Regression Comparison**. Full data simulated from $y = w_1 + w_{10} + er$, where $w_i \sim B(0.7)$, $1 \leq i \leq 10$, and $er \sim N(0, 1)$. Candidate estimator was chosen over 10 repetitions of two sample sizes (col 1), three v-fold cross-validations (col 2) for both our algorithm and logic regression (col 3). The results (cols 4-7) in the table are based on an independent test sample of $n = 10000$. col 4 is the average size (number of basis functions for ours and number of leaves for logic), col 5 is the average of the 10 risks (with the L_2 loss function) for each method, col 6 is the variance of the risks over the 10 reps, and col 7 is the ratio of averaged risks (col 5) − optimal risk, (dsa/logic).

Sample			10 Repetitions			
Size	$v - fold$	Method	avg size	mean risk	std dev	ratio
250	2	dsa1	1.6	1.713	.072	.975
		dsa2	2.0	1.745	.073	1.019
		logic	2.0	1.731	.077	1
	5	dsa1	2.0	1.741	.073	1.019
		dsa2	1.9	1.724	.086	.996
		logic	1.1	1.727	.082	1
	10	dsa1	2.0	1.741	.073	1.019
		dsa2	2.0	1.745	.073	1.025
		logic	1.1	1.727	.082	1
1000	2	dsa1	1.8	1.708	.055	.990
		dsa2	2.0	1.723	.041	1.011
		logic	2.0	1.715	.031	1
	5	dsa1	2.0	1.723	.041	1.011
		dsa2	2.0	1.723	.041	1.011
		logic	2.0	1.715	.031	1
	10	dsa1	2.0	1.723	.041	1.011
		dsa2	2.0	1.723	.041	1.011
		logic	2.0	1.715	.031	1

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

Table 9: *Simulation study 2* **Logic Regression Comparison**. Full data simulated from $y = \beta_1(w_1 w_3(1-w_2))+\beta_2((1-w_1)w_3(1-w_2))+\beta_3(w_7 w_{10})+er$, where $w_i \sim \mathcal{B}(0.7)$, $1 \leq i \leq 10$, $\beta \sim \mathcal{N}(1,1)$, and $er \sim N(0,1)$. Candidate estimator was chosen over 10 repetitions of two sample sizes (col 1), three v-fold cross-validations (col 2) for both our algorithm and logic regression (col 3). The results (cols 4-7) in the table are based on an independent test sample of $n = 10000$. col 4 is the average size (number of basis functions for ours and number of leaves for logic), col 5 is the average of the 10 risks (with the L_2 loss function) for each method, col 6 is the variance of the risks over the 10 reps, and col 7 is the ratio of averaged risks (col 5) − optimal risk, (dsa/logic).

Sample			10 Repetitions			
Size	$v-fold$	Method	avg size	mean risk	std dev	ratio
250	2	dsa1	4.1	4.647	.248	.995
		dsa2	4.0	4.636	.254	.992
		logic	5.6	4.664	.255	1
	5	dsa1	4.8	4.705	.319	1.013
		dsa2	4.7	4.723	.297	1.018
		logic	6.0	4.657	.246	1
	10	dsa1	4.8	4.705	.319	1.014
		dsa2	4.7	4.723	.297	1.019
		logic	6.0	4.654	.278	1
1000	2	dsa1	5.0	4.738	.101	1.001
		dsa2	4.9	4.731	.105	.999
		logic	7.0	4.734	.105	1
	5	dsa1	5.0	4.738	.101	1.001
		dsa2	5.0	4.743	.101	1.002
		logic	7.0	4.734	.105	1
	10	dsa1	5.0	4.738	.101	1.001
		dsa2	5.0	4.743	.101	1.002
		logic	7.0	4.734	.105	1

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Table 10: *Simulation study 3* **Logic Regression Comparison**. Full data simulated from $y = \beta_1 w_5 w_{15} w_{25} w_{35} + \beta_2 w_{70} w_{80} w_{90} + \beta_3 w_{51} w_{61} + \beta_4 w_{12} w_{19} w_{47} w_{69} w_{83} + \beta_5 w_{100} + er$, where $w_i \sim \mathcal{B}(0.5)$, $1 \leq i \leq 100$, $\beta \sim \mathcal{N}(1,1)$, and $er \sim N(0,1)$. Candidate estimator was chosen over 1 repetition of two sample sizes (col 1), three v-fold cross-validations (col 2) for both our algorithm and logic regression (col 3). The results (cols 4-7) in the table are based on an independent test sample of $n = 10000$. col 4 is the size (number of basis functions for ours and number of leaves for logic), col 5 is the risk (with the L_2 loss function) for each method, and col 6 is the ratio of averaged risks (col 5) − optimal risk, (dsa/logic).

Sample			1 Repetition		
Size	$v-fold$	Method	size	risk	ratio
250	2	dsa1	2.0	3.028	1.008
		dsa2	2.0	3.028	1.008
		logic	3.0	3.011	1
	5	dsa1	5.0	3.275	1.011
		dsa2	5.0	3.275	1.011
		logic	3.0	3.250	1
	10	dsa1	5.0	3.275	1.011
		dsa2	5.0	3.275	1.011
		logic	3.0	3.250	1
1000	2	dsa1	3.0	3.094	1.013
		dsa2	3.0	3.103	1.017
		logic	3.0	3.068	1
	5	dsa1	5.0	3.128	.990
		dsa2	5.0	3.167	1.008
		logic	7.0	3.149	1
	10	dsa1	5.0	3.128	.990
		dsa2	5.0	3.167	1.008
		logic	7.0	3.149	1

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

5 Data Analysis.

An important problem in contemporary biology is transcription factor binding site identification. Transcriptors are proteins that bind to regions in the vicinity of genes and as a result regulate the activities of the genes. Identification of these sites is a crucial problem as understanding the components of regulation is a step toward understanding how genes are expressed at all times in the cell life. This biological problem has been put by Keleş et al. (2002) into a statistical framework by formulating it as a model selection problem. Keleş et al. (2002) model gene expression as a function of short oligonucleotides that represent potential binding sites and use pentamers as an initial set of covariates, adopting a stepwise cross-validation methodology with forward selection and backward deletion to choose the most predictive pentamers. This selection approach also searches for two-way interaction terms that improves the prediction of the multivariate linear regression model.

Similar to the approach taken by Keleş et al. (2002), two implementations of the D/S/A algorithm were applied to a publicly available dataset of the yeast *Saccharomyces cerevisiae*, focusing on the 800 basepairs in the upstream control region (UCR). The D/S/A algorithm is used to regress gene expression on the indicators (binary scores) of the presence of a length five motif (pentamer) at various time points of the cell cycle data by Cho et al. (1998). The set of explanatory variables were the scores of 512 pentamers for 2836 genes obtained from Keleş et al. (2002). As described in Keleş et al. (2002), cells were collected at 17 time points with 10 minute intervals to cover two full cell cycles in the experiments of Cho et al. (1998). Just as in (Keleş et al., 2002), two time points (90 min. and 100 min.) were discarded because of less efficient labeling of their mRNA prior to hybridization (Tavazoie et al., 1999), and normalized expression profiles of the most variable, 2836 ORFs, were used.

The two implementations of the D/S/A algorithm were: (1) select the number of tensor products in the fitted model via cross-validation and leave the complexity measure on the tensor products, $m_2(I) = \max \sum_{j=1}^{d} \mathrm{I}(p_j \neq 0)$, unconstrained, i.e., $CV - DSA\{\text{loss} = L_2;\ \text{basis} = \text{poly};\ m(I) = \{m_1(I)\};\ \beta_{I,s} = \text{everything};\ k = \{1,\ldots,5\};\ v = 2;\ \text{moves} = (1,1);\ \text{data} = P_n\ \}$ or, (2) select both s_1 and s_2 via cross-validation, i.e., $CV - DSA\{\text{loss} = L_2;\ \text{basis} = \text{poly};\ m(I) = \{m_1(I), m_2(I)\};\ \beta_{I,s} = \text{everything};\ s_1 = \{1,\ldots,5\},\ s_2 = \{1,\ldots,5\};\ v = 2;\ \text{moves} = (1,1);\ \text{data} = P_n\ \}$.

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Results given by *DSA1-CV* and *DSA2-CV* for the 15 time points are given in Tables 11 - 12. In the interest of saving time, the algorithm ran up until a maximal size of five. The average runtime of the second implementation of the D/S/A algorithm was about 2.5 hours for the 15 time points.

Let's look in particular at the first results for T = 0 minutes. The final model is composed of four terms ($\hat{s}_1 = 4$, $s_2 = \infty$): a four-way interaction, two five-way interactions including pentamers *AGGGG* and *ATAAA* and/or their reverse complements (N.B. a pentamer can refer to itself and/or its reverse complement), and a 12-way interaction including pentamers *AAACA*, *AACGC*, *AATAA*, *ACAAA*, and *ACGCG*. Wolfsberg et al. (1999) identified pentamers and hexamers as potential regulatory motifs, and we compared our findings to the significant pentamers they found ($p \leq 0.05$). The pentamers involved in the interaction terms for the first time point coincide with their significant pentamers. For example, *AACGC/GCGTT* overlaps with MCB at four of the five positions. MCB is known to control the expression of genes expressed in late G_1 phase of the cell cycle (Wolfsberg et al., 1999). Hence the method was able to select biologically relevant pentamers, and perhaps, other identified pentamers play a role in predicting gene expression. These first set of results, however, indicate the importance of applying a second brake to the algorithm, namely limiting the number of terms within each tensor product which led us to determine s_2 via cross-validation for each of the given time points. This would avoid the problem of fitting a model with such higher-order interactions, e.g. 12-way interactions and higher. For instance, at T = 20 minutes, *DSA1-CV* yielded a final model with 5 terms, including a 16-way interaction. Yet when we let cross-validation choose s_2 (*DSA2-CV*), we result in a model with four main terms. Keleş et al. (2002) show that the three pentamers with the highest rank at T = 20 minutes are: *ACGCG/CGCGT* (mcb), *CGCGA/TCGCG* (scb), and *AGGGG/CCCCT* (stre) which are three of the main terms picked by *DSA2-CV*. It should be noted that there are instances where the results given by *DSA1-CV* and *DSA2-CV* coincide, for example, both methods picked the same model with one 2-way interaction at T = 120 minutes.

6 Right censored data loss functions.

Suppose that $X = \bar{X}(T) \equiv (X(t) : t \leq T)$ is the full-data structure of interest which ends at a possibly random time T (such as a survival time),

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

Table 11: *Yeast Data Analysis* **DSA1-CV vs. DSA2-CV**, 2-fold cross-validation, applied to yeast cell cycle data of (Cho et al., 1998), time point 1.

T=0 min
$(\hat{s}_1 = 4,\, s_2 = \infty)$ $I(AAGAA)I(ACCCC)I(AGGGG[stre])I(ATAAA)I(CAGTA)$ $+ \quad I(AAACA[ste12])I(AAAGG)I(AACGC)I(AATAA)I(AATAT)I(AATCT)$ $I(ACAAA)I(ACCGT)I(ACGCG[mcb])I(ACTCT)I(AGTAA)I(CCACG)$ $+ \quad I(AAGGA)I(AAGGG)I(AGAAA)I(CTTAA)$ $+ \quad I(AAAAA)I(AGAAA)I(ATGAG)I(CATCG)I(GATGA)$
$(\hat{s}_1 = 5,\, \hat{s}_2 = 3)$ $I(ACCCC)I(AGGGG[stre])I(CAGTA) \quad + \quad I(ATGAG)I(CATCG)I(GATGA)$ $+ \quad I(AAGGA)I(AAGGG)I(CTTAA) \quad + \quad I(ACAAA)I(ACGCG[mcb])I(CGAAA)$ $+ \quad I(CACTA)I(CCAGG)I(GCCCC)$

but that the observed data structure is given by

$$O \equiv \left(\tilde{T} = \min(T, C),\ \Delta = I(T \leq C),\ \bar{X}(\tilde{T})\right),$$

for a right-censoring variable C with conditional distribution $G_0(\cdot|X)$, given the full data structure X.

By convention, if $T < C$, let $C = \infty$. One can then rewrite the observed data structure as $O = (\bar{X}(C), C)$. The distribution, $P_0 = P_{F_{X,0},G_0}$, of the observed data structure O is indexed by the full data distribution $F_{X,0}$ and the conditional distribution $G_0(\cdot|X)$ of the censoring variable C. Let $\mathcal{M}^F$ be the model for the full data distribution. We assume that G_0 satisfies the coarsening at random (CAR) assumption:

$$Pr_0(C = t \mid C \geq t, \bar{X}(T)) = Pr_0(C = t \mid C \geq t, \bar{X}(t)), \qquad \text{for } t < T.$$

If X does not include time-dependent covariates (e.g., $X = (W, Z)$), then, CAR is equivalent with assuming that C is conditionally independent of the survival time T, given baseline covariates W.

The parameter of interest $\Psi : \mathcal{M}^F \to D(\mathcal{S})$, and as in Section 2 the parameter $\psi_0 = \Psi(P_0)$ is defined in terms of a loss function, $L(X, \psi)$, as (one

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

T=10 min
$(\hat{s}_1 = 5,\ s_2 = \infty)$ $I(AAACA[ste12])I(AAACC)I(AAAGA)I(AAAGG)I(AAGAG)I(AATCA)I(AATTG)$ $I(ACAAC)I(ACAGG)I(AGAAA)I(ATAAA)I(ATGTC)I(ATTGA)I(CAAAA)$ $I(CAGGA)I(GAAAA[ecb])I(GCAAA)I(TAAAA)I(TGAAA)I(TTCAA)$ $+ \quad I(AAAAA)I(AAACA[ste12])I(AAAGC)I(AACAC)I(AAGAA)I(AATGA)$ $I(AGAAA)\ I(AGGGG[stre])I(ATAAA)I(ATAGA)I(ATTTG)I(CTAAA)I(GAAAA[ecb])$ $I(GGAAA)I(GTTTA[sff])I(TAGAA)I(TATAA)$ $+ \quad I(AAAAT)I(AAGAG)I(AAGTG)I(AATGA)I(ATCAA)I(ATGAA)I(TATAA)$ $I(TTCAA) \quad + \quad I(AAACT)I(AAAGA)I(AAGAC)I(AATCA)I(ACAAT)$ $I(ACTCT)I(ATAAA)I(ATAAC)\ I(ATGGC)I(CCAGC)I(CTTTA)I(GATAA)$ $I(GATCC)I(TAAGA)I(TGAAA)I(TTCAA)$ $+ \quad I(AAAAA)I(AAAAC)I(AAACA[ste12])I(AAACT)I(AAAGC)I(AAATT)$ $I(AACAC)\ I(AAGAA)I(AATGA)I(AGAAA)I(AGGGG[stre])I(ATAAA)I(ATAAT)$ $I(ATAGA)\ I(ATGCA)I(ATTAA)I(ATTTA)I(ATTTG)I(CAAGC)I(GAAAA[ecb])$ $I(GACAC)\ I(GGAAA)I(GTTTA[sff])I(TAGAA)I(TATAA)I(TCAAA)I(TCTCA)$
$(\hat{s}_1 = 3,\ \hat{s}_2 = 2)$ $I(AACAC)I(GGGGA) \quad + \quad I(CAGGA)I(CGAGA)$ $+ \quad I(GATGA)I(TTCAA)$

T=20 min
$(\hat{s}_1 = 5,\ s_2 = \infty)$ $I(ACGCG[mcb]) \quad + \quad I(AAAAC)I(AAACA[ste12])I(AAAGT)I(AAGGA)I(AATGG)$ $I(AGGGG[stre])I(AGTAA)I(ATTAC)I(ATTTG)I(CCAAA)I(CCTGA)I(GATTA)$ $I(GGAAA)\ I(GTAAA)I(TATAA)I(TCAAA)$ $+ \quad I(AACTT)I(AAGCA)I(AATAG)I(ACACA)I(AGAAT)I(AGAGA)$ $I(AGCAA)\ I(AGGAC)I(ATAGA)I(ATGAA)I(CAAAG)I(CACAA)I(CCCGC)$ $I(CTAAA)I(TAAGA)$ $+ \quad I(AAAAA)I(AAATG)I(AATAA)I(ATACA)I(ATATA)I(ATTGA)I(CGCGA[scb])$ $I(GCGAA[scb])$ $+ \quad I(AAAAT)I(AAAGC)I(AAATT)I(AACAC)I(ACGCG[mcb])I(ACTTA)I(ATAAG)$ $I(ATATG)I(GGAAA)I(TATAA)I(TGTAA)$
$(\hat{s}_1 = 4,\ \hat{s}_2 = 1)$ $I(ACGCG[mcb]) \quad + \quad I(AGGGG[stre]) \quad + \quad I(CGCGA[scb]) \quad + \quad I(GATTA)$

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

T=30 min
$(\hat{s}_1 = 3,\ s_2 = \infty)$
$I(ACGCG[mcb]) \quad + \quad I(AAAGA)I(AACAC)I(AACTT)I(AATAG)I(ACAAC)$ $I(ACGCG[mcb])I(AGCCG)I(ATTTA)\ I(CAAGC)I(CCATA)I(CTTTA)I(GAAAA[ecb])$ $I(GAAGA)I(GATAC)I(GCAAA)I(TAATA) \quad + \quad I(AAAAC)I(AAATA)$ $I(AAATT)I(AATTC)I(AATTG)I(ACGCG[mcb])I(ATCAA)\ I(CTAAA)I(TCAAA)$
$(\hat{s}_1 = 5,\ \hat{s}_2 = 3\ )$
$I(ACGCG[mcb]) \quad + \quad I(ACGCG[mcb])I(AGCCG)I(GATAC)$ $\quad + \quad I(AACGC)I(ATAAT)I(CGCGA[scb]) \quad + \quad I(AAAAG)I(AATAT)I(TTAAA)$ $\quad + \quad I(AACGT)I(AATTC)I(ACGCG[mcb])$

T=40 min
$(\hat{s}_1 = 2,\ s_2 = \infty)$
$I(AAAAG)I(AAACT)I(AAAGT)I(AACAA)I(AACGC)I(AAGAA)I(AATAT)$ $I(ACAAC)\ I(AGACA)I(AGTAA)I(ATGCC)I(ATTGA)I(CAACA)I(CGCCA)$ $I(GAATA)\ I(GCGAA[scb])I(GGAAA) \quad + \quad I(AAAAG)I(AAAAT)$ $I(AAATA)I(AACGA)I(AATAA)I(ACAAA)I(ACGAA)I(ATAAT)$ $I(ATCGC)I(CAAAA)I(CACGA)I(GAAGA)$
$(\hat{s}_1 = 1,\ \hat{s}_2 = 2)$
$I(AGACA)I(CGCCA)$

T=50 min
$(\hat{s}_1 = 2,\ s_2 = \infty)$
$I(AAAAA)I(AAACA[ste12])I(AAATT)I(AAGAT)I(AAGTA)I(AATGT)$ $I(ACATA)I(ACTTC)I(AGAAG)I(AGGGA)I(ATGAG)\ I(ATTTA)I(CTTGA)$ $I(GAAGA)I(GCAAA)I(GCGGA)\ I(TAATA)I(TAGAA)I(TCTCA)I(TGAAA)I(TGTAA)$ $\quad + \quad I(AAAAC)I(AAACT)I(AAGAA)I(AAGTG)I(ACAAA)\ I(ACTTG)I(AGCAT)$ $I(ATAAA)\ I(ATCAA)I(ATGAA)I(ATTCA)\ I(CGAAA)I(CTTGC)I(GCATA)I(GTATA)$ $I(GTTCA)\ I(TAACA)\ I(TGCAA)I(TTAAA)$
$(\hat{s}_1 = 2,\ \hat{s}_2 = 1)$
$I(GCGGA) \quad + \quad I(TCCCA)$

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

T=60 min
$(\hat{s}_1 = 4,\, s_2 = \infty)$ $I(AAACA[ste12])I(AAACG)I(AAACT)I(AAAGA)I(AAATT)I(AAGAT)I(ACAAG)$ $I(AGAAT)I(AGTAA)I(ATAGA)I(ATATG)I(ATCAC)I(ATCTG)I(CCATA)I(CGCTC)$ $I(GACAA)I(GCAAA)I(TCAAA)I(TCCAA)I(TGAAA)I(TGTAA)$ $+ \quad I(AAAAG)I(AAAGA)I(AACAC)I(AGAAG)I(AGGGG[stre])I(ATAAT)I(ATATC)$ $I(ATGTA)I(ATTAA)I(CAAAA)I(CATTA)$ $+ \quad I(AAAAT)I(AAACA)I(AAAGG)I(AGAAA)I(ATATA)I(CAGAA)I(CATAA)$ $I(CTCCA)I(GACGC)$ $+ \quad I(AAATT)I(AGATG)I(TCAAA)I(TGAAA)$
$(\hat{s}_1 = 5,\, \hat{s}_2 = 2)$ $I(CTCCA)I(GACGC) \quad + \quad I(AGATG)I(TCAAA)$ $+ \quad I(AACAC)I(AGGGG[stre]) \quad + \quad I(AAAAC)I(CGAGA)$ $+ \quad I(ATATC)I(CTTAC)$

T=70 min
$(\hat{s}_1 = 4,\, s_2 = \infty)$ $I(ACGCG[mcb]) \quad +$ $I(AAAAC)I(AAAAG)I(AAACC)I(AAACT)I(AAAGC)I(AAATA)I(AAGGA)$ $I(ACAAA)I(ACGAT)I(ACGCG[mcb])I(ACTTA)I(AGCAA)I(CACCA)I(CTTTA)$ $I(GAAGA)I(GGATA)I(GGGAA)I(TAAGA)I(TACAA)I(TACGA)I(TTAAA)$ $+ \quad I(AAAGC)I(AAATC)I(AAATT)I(AAGAG)I(AAGTA)I(ACAAA)I(ACAAC)$ $I(ACATA)I(ATTCG)I(CATCG)I(CATTA)I(CCGTC)I(CGAAG)I(GGATA)$ $I(GTCAA)I(TAATA)I(TACAA)I(TAGAA) \quad + \quad I(AAAAT)I(AAATA)$ $I(AATAA)I(ATAAT)I(CGCGA[scb])I(GCGAA[scb])I(TACAA)$
$(\hat{s}_1 = 4,\, \hat{s}_2 = 1)$ $I(ACGCG[mcb]) \quad + \quad I(CGCGA[scb]) \quad + \quad I(CATCG) \quad + \quad I(GACGC)$

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

T=80 min
$(\hat{s}_1 = 5,\ s_2 = \infty)$ $I(AAAAA)I(AACAA)I(ACGCG[mcb])I(CAAAA) + I(AAAAT)$ $I(AAACA[ste12])I(AAATG)I(AACAT)I(AACTT)I(AAGAT)I(AATAT)I(AATGA)$ $I(AATTG)I(ACAAA)I(ACTAA)I(AGAAT)I(AGATG)I(AGTAA)I(AGTTA)I(ATAAG)$ $I(ATAAT)I(ATAGA)I(ATGAA)I(ATTAA)I(ATTAG)I(ATTGA)I(CAATA)I(CACAA)$ $I(CAGAA)I(CTAAA)I(GAAAA[ecb])I(GAACA)I(GATGA)I(GATTA)I(GTTAA)$ $I(GTTTA[sff])I(TATAA)I(TCCAA)I(TGAAA)I(TGGAA)$ $+ I(AAAAA)I(AACAA)I(AACAT)I(AACTT)I(AAGCA)I(AAGGG)$ $I(AAGTA)I(AATAT)I(ACAAT)I(ACCCC)I(ACGCG[mcb])I(AGATA)I(AGGAA)$ $I(CAAAA)\ I(CTAAA)I(CTTTC)I(GAAAC)I(GATAA)I(GTAAA)I(TAATA)$ $I(TCCAA)I(TGACA) + I(AAAGG)I(AATAA)I(ACAAA)I(ACACG)$ $I(ATACA)I(ATTTG)\ I(CGAAA)\ I(CGCGA[scb])I(GCGAA[scb])I(TAAAA)I(TGAAA)$ $I(TTAAA) + I(AAAGG)\ I(AATTC)I(CATAA)I(CTAAA)I(CTTTA)$ $I(GAATA)\ I(GATTA)I(GGAAA)I(TACTA)$
$(\hat{s}_1 = 5,\ \hat{s}_2 = 2)$ $I(AACAA)I(ACGCG[mcb]) + I(CGCGA[scb])I(GCGAA[scb])$ $+ I(CTAAA)I(GATTA) + I(CACCC)I(CTAAG) + I(AACTT)I(ACCCC)$

T=110 min
$(\hat{s}_1 = 1,\ s_2 = \infty)$ $I(AAAAA)I(AAAAT)I(AAACA)I(AAAGA)I(AAAGT)I(ACGCG[mcb])I(GACGC)$ $I(TAGCA)$
$(\hat{s}_1 = 5,\ \hat{s}_2 = 1)$ $I(CGCGA[scb]) + I(AGTCA) + I(GACGC) + I(ATGAG)$ $+ I(ACGCG[mcb])$

T=120 min
$(\hat{s}_1 = 1,\ s_2 = \infty)$ $I(AACTT)I(CTTTA)$
$(\hat{s}_1 = 1,\ \hat{s}_2 = 2)$ $I(AACTT)I(CTTTA)$

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

T=130 min
$(\hat{s}_1 = 1,\, s_2 = \infty)$ $I(AAACG)I(AAATC)I(AAGCG)I(AAGGT)I(AATGA)I(ACAAA)I(ACAAT)$ $I(ACGAA)\ I(AGATC)I(ATAGC)I(CTTCA)I(CTTGA)I(CTTTC)I(GAGAA)$ $I(GCACA)I(GCCAA)\ I(GCTAA)I(GTAAC)I(TCGAA)I(TTCAA)$
$(\hat{s}_1 = 2,\, \hat{s}_2 = 1)$ $I(TCGAA) \quad + \quad I(ACCCA)$

T=140 min
$(\hat{s}_1 = 1,\, s_2 = \infty)$ $I(AACGT)I(ACGTA)I(ATAAC)I(CACAC)I(CGCGA[scb])I(CTAGC)I(CTGTA)$ $I(CTTCA)I(GGAAC)I(GGCAA)I(GGCTA)I(TAGAA)$
$(\hat{s}_1 = 1,\, \hat{s}_2 = 2)$ $I(CACAC)I(CGCGA[scb])$

T=150 min
$(\hat{s}_1 = 1,\, s_2 = \infty)$ $I(AAAAT)I(AAACT)I(AAAGC)I(AACAA)I(AATTA)I(ACATA)I(ACGCG[mcb])$ $I(ACTTG)I(AGAAT)I(AGATA)I(ATAAT)I(ATCAT)I(ATTAG)I(ATTTC)$ $I(CAAAG)I(CTTCA)I(GAATA)I(TAATA)I(TACTA)$
$(\hat{s}_1 = 1,\, \hat{s}_2 = 2)$ $I(ACGCG[mcb])I(AGATA)$

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

Table 12: *Yeast Data Analysis* **DSA1-CV vs. DSA2-CV**, 2-fold cross-validation, applied to yeast cell cycle data of (Cho et al., 1998), time point 15.

T=160 min
$(\hat{s}_1 = 3,\, s_2 = \infty)$ $I(AAATC)I(AACTA)I(AACTT)I(AAGGA)I(AATAA)I(AATAT)I(AATCA)$ $I(ACTAA)I(AGAAA)I(AGCCT)I(AGGTA)I(ATACA)I(ATGTA)I(CAATC)$ $I(CATAA)I(CTGAA)I(GAATC)I(GTTTA[sff])I(TCTCA)$ $+ \quad I(AAAAA)I(AAGAA)I(AATAA)I(AATAT)I(ACGCG[mcb])I(ATAAA)$ $I(ATACA)\ I(ATATC)I(GAAAA[ecb])I(GTTTA[sff])$ $+ \quad I(AAAAC)I(AAGAA)I(AAGGT)I(AATAA)I(AATAC)I(AATAT)$ $I(ACATC)\ I(ACCCA)I(ACGCG[mcb])I(AGTAA)I(ATAAA)I(ATACA)I(ATATC)$ $I(CATGC)\ I(GAAAA[ecb])I(GAAGA)I(GTACA)I(TAATA)I(TACTA)I(TTAAA)$
$(\hat{s}_1 = 4,\, \hat{s}_2 = 2)$ $I(AACGC)I(ACGCG[mcb]) \quad + \quad I(ACGGA)I(ACTAA)$ $+ \quad I(CAAGA)I(CGGGA) \quad + \quad I(CGCCA)I(TTAAA)$

of) the minimizer(s) of the expected loss, or risk,

$$\int L(x, \psi_0) dF_{X,0}(x) \equiv \min_{\psi \in \Psi} \int L(x, \psi) dF_{X,0}(x).$$

IPCW loss function: We can map the full data loss function into the IPCW observed data loss function (van der Laan and Robins (2003)):

$$L(O, \psi \mid G) = L(X, \psi) \frac{\Delta}{\bar{G}(T|W)}.$$

Note that indeed, if $\bar{G}_0(T \mid X) > 0$, F_{X0}-a.e, then $E_0 L(O, \psi \mid G_0) = E_0 L(X, \psi)$. For finite sample and asymptotic results regarding the cross-validation selector based on this loss function, we refer to van der Laan and Dudoit (2003) and (Keleş et al., 2003).

The optimal doubly robust inverse probability of censoring weighted (DR-IPCW) loss function is more involved and given by (van der Laan and Robins (2003))

$$L(O, \psi \mid Q, G) \equiv \frac{L(X, \psi)\Delta}{\bar{G}(T|X)}$$

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

$$+ \int E_{G,Q}\left(\frac{L(X,\psi)\Delta}{\bar{G}(T|X)} \mid \bar{X}(u), \tilde{T} \geq u\right) dM_G(u),$$

where $dM_G(u) = I(\tilde{T} \in du, \Delta = 0) - I(\tilde{T} \geq u)\lambda_c(u|X)du$, and $Q = Q(F_X)$ refers to the F_X-part of the density for the observed data, $O = (\bar{X}(C), C)$, under the CAR assumption.

If G_n, Q_n are consistent estimators of G_0, Q_0, respectively, then under regularity conditions (van der Laan and Robins (2003)) $1/n \sum_i L(O_i, \psi \mid Q_n, G_n)$ is an asymptotically efficient estimator (in the non-parametric observed data model) of the full data risk $E_0 L(X, \psi)$. In addition, if either G_n or Q_n is inconsistent, but not both, then this risk estimate remains consistent, due to the double robustness property:

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

if either $G = G_0$ (and $\bar{G}_0(T \mid X) > 0$, F_X-a.e) or $Q = Q_0$.

The ability to map from a full data loss function into an observed data loss function with the same risk offers several important practical advantages. Firstly, this allows us to directly extend full data estimation methodology to censored data situations: for example, if there is no censoring, then our methodology reduces to the preferred full data methodology. This in contrast to common censored data estimation approaches, such as survival trees, which bypass the risk estimation problem for censored outcomes by altering the node splitting, tree pruning, and performance assessment criteria in manners that are specific to censored survival data (Molinaro et al., 2003). The splitting and pruning criteria seem to be chosen based on convenience for handling censored data and do not reduce to the preferred choice for uncensored data. For example, most tree-based regression and density estimation procedures rely on the negative log-likelihood loss function assuming certain models (such as exponential distributions within each node, or Cox-proportional hazards model within each node), with the explicit or implicit goal of estimating the conditional survival function given explanatory variables, and differ mainly in their choice of model for the observed data likelihood within nodes. This general difficulty in evaluating risk for censored observations results in a discontinuity between the full and observed data worlds. Secondly, as shown in Molinaro et al. (2003), gains in accuracy can be achieved by employing a loss function that is specific to the parameter of interest (e.g., by using the squared error loss function for regression

Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

rather than the negative log-likelihood loss function typically used in survival trees which are designed to estimate the density itself). Finally, the IPCW loss function allows us to assess performance on censored data for arbitrary full-data loss functions. Current methods typically rely on the negative log-likelihood loss function or lead to biased risk estimators by ignoring censored observations altogether.

7 Discussion.

The D/S/A algorithm, as presented in this article, has been implemented in the context of polynomial regression where the number of basis functions is chosen by v-fold cross-validation. Also available is (1) the option to restrict the order of interaction of candidate tensor products to be no higher than a specified limit s_2 and to choose s_2 also with cross-validation and (2) the option to restrict the sum of polynomial powers of candidate tensor products to be no higher than a specified limit s_3 and to choose s_3 again with cross-validation. The restriction on s_3 of course is not needed in settings involving binary inputs and hence was not demonstrated in the Logic Regression simulations or in the data analysis. Future work will allow the constraint(s) on the vector of coefficients to be an available option as well.

The simulations, displayed in Tables 5 - 7, which compare the D/S/A algorithm to a forward selection cross-validation algorithm involve unusual situations to see how a very aggressive algorithm compares to a less aggressive algorithm. The forward selection algorithm used in those simulations does involve aggressive forward moves because it uses the *addition* moves proposed in this article, namely adding a main term or increasing or decreasing the coordinates of a term by one (described in Section 3). These simulations illustrate the fact mentioned in Section 2.1 that the cross-validation selector performs well for uniformly bounded loss functions. If the variance of the loss function is large relative to the number of observations, this results in poor estimates of the empirical risk. Therefore, for theoretical reasons, we would not expect good performance (e.g., Table 6). It is interesting to observe that forward selection with cross-validation is able to fit a large model with good risk performance, though it is expected to fit a larger model where the truth is a subset of the fitted model given the nature of the simulated data. In these simulations, a choice must be made between high specificity (CV-DSA) and low risk (fscv).

Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

Logic Regression is freely available in R and is a nice tool to find interactions between binary inputs associated with an output. Based on the limited number of comparisons made between Logic Regression and CV-DSA, it appears that the D/S/A algorithm is competitive with Logic Regression. One advantage of the D/S/A algorithm is its ability to handle continuous and binary inputs.

Barron and Xiao argue in favor of their multivariate adaptive polynomial synthesis (MAPS) method over MARS (Friedman, 1991, pg. 67-82). At the time of writing their discussion, Barron had only implemented the forward stepwise synthesis in the MAPS program, implying the utility of allowing backward passes. Perhaps, this is similar to the fscv method that we have implemented in Section 4.3. MARS first builds a model with its forward moves, and at the end of that process, a backward deletion procedure is applied. The D/S/A algorithm always is attempting to make backward and substitution selection moves throughout its search, thereby eliminating the luggage of undesirables.

Polynomials are a reasonable choice for basis functions for reasons including its known approximation capabilities, interpretability, and a model dimension which tends to be smaller than the sample size (Cox, 1988; Friedman, 1991, pg. 67-82). In the rejoinder to (Friedman, 1991, pg. 123-141), Friedman argues that polynomials do not possess a good locality property: *"the function estimate at a point can be strongly influenced by data points very far away from it in the predictor space."* The idea here is not to argue for or against polynomials (or splines for that matter) as a choice of basis function but to introduce a new class of algorithms, the D/S/A algorithms. Polynomials served as a practical way to represent one version of the D/S/A algorithm. The D/S/A algorithm generates candidate estimators and is defined by choices: the loss function; the basis functions; and the set of deletion, substitution, and addition moves.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

References

A. R. Barron. Statistical properties of artificial neural networks. In *Proceedings of the 28th Conference on Decision and Control*, pages 280–285, Tampa, Florida, 1989.

A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

A. R. Barron, L. Birgé, and P. Massart. Risk bounds for model selection via penalization. *Probability Theory and Related Fields*, 113:301–413, 1999.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. The Wadsworth Statistics/Probability series. Wadsworth International Group, 1984.

C. Canuto and A. Quarteroni. Approximation results for orthogonal polynomials in Sobolev spaces. *Mathematics of Computation*, 38(157):67–86, 1982.

R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell*, 2:65–73, 1998.

D. D. Cox. Approximation of least squares regression on nested subspaces. *The Annals of Statistics*, 16(2):713–732, 1988.

S. Dudoit and M. J. van der Laan. Asymptotics of cross-validated risk estimation in model selection and performance assessment. Technical Report 126, Division of Biostatistics, University of California, Berkeley, Feb. 2003. URL `www.bepress.com/ucbbiostat/paper126/`.

S. Dudoit, M. J. van der Laan, S. Keleş, A. M. Molinaro, S. E. Sinisi, and S. L. Teng. Loss-based estimation with cross-validation: Applications to microarray data analysis and motif finding. Technical Report 137, Division of Biostatistics, University of California, Berkeley, Dec. 2003. URL `www.bepress.com/ucbbiostat/paper137/`.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991. Discussion by A. R. Barron and X. Xiao.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, 2001.

S. Keleş, M. J. van der Laan, and S. Dudoit. Asymptotically optimal model selection method with right censored outcomes. Technical Report 124, Division of Biostatistics, University of California, Berkeley, Sept. 2003. URL `www.bepress.com/ucbbiostat/paper124/`.

S. Keleş, M. J. van der Laan, and M. B. Eisen. Identification of regulatory elements using a feature selection method. *Bioinformatics*, 18:1167–1175, 2002.

A. M. Molinaro, S. Dudoit, and M. J. van der Laan. Tree-based multivariate regression and density estimation with right-censored data. Technical Report 135, Division of Biostatistics, University of California, Berkeley, Sept. 2003. URL `www.bepress.com/ucbbiostat/paper135/`.

A. M. Molinaro and M. J. van der Laan. A Deletion/Substitution/Addition algorithm for partitioning the covariate space in prediction. Technical report, Division of Biostatistics, UC Berkeley, 2004. (In preparation).

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

I. Ruczinski, C. Kooperberg, and M. LeBlanc. Logic regression. *Journal of Computational and Graphical Statistics*, 12(3):475–511, 2003. URL `www.biostat.jhsph.edu/~iruczins/publications/publications.html`.

S. Tavazoie, J. D. Hughes, M. J. Campbell, R. J. Cho, and G. M. Church. Systematic determination of genetic network architecture. *Nature Genet.*, 22:281–285, 1999.

M. J. van der Laan and S. Dudoit. Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples. Technical Report 130, Division of Biostatistics, University of California, Berkeley, Nov. 2003. URL `www.bepress.com/ucbbiostat/paper130/`.

Collection of Biostatistics
Research Archive

Hosted by The Berkeley Electronic Press

M. J. van der Laan, S. Dudoit, and S. Keleş. Asymptotic optimality of likelihood based cross-validation. Technical Report 125, Division of Biostatistics, University of California, Berkeley, Feb. 2003. URL `www.bepress.com/ucbbiostat/paper125/`.

M. J. van der Laan and J. Robins. *Unified Methods for Censored Longitudinal Data and Causality*. Springer Series in Statistics. Springer, 2003.

T. G. Wolfsberg, A. E. Gabrielian, M. J. Campbell, R. J. Cho, J. L. Spouge, and D. Landsman. Candidate regulatory sequence elements for cell cycle-dependent transcription in *Saccharomyces cerevisiae*. *Genome Research*, 9:775–792, 1999.

COBRA
A BEPRESS REPOSITORY
Collection of Biostatistics
Research Archive

http://biostats.bepress.com/ucbbiostat/paper143