Statistical Applications in Genetics and Molecular Biology

Volume 3, *Issue* 1 2004 *Article* 18

Deletion/Substitution/Addition Algorithm in Learning with Applications in Genomics

Sandra E. Sinisi, *Division of Biostatistics, School of Public Health, University of California, Berkeley*
Mark J. van der Laan, *Division of Biostatistics, School of Public Health, University of California, Berkeley*

Recommended Citation:
Sinisi, Sandra E. and van der Laan, Mark J. (2004) "Deletion/Substitution/Addition Algorithm in Learning with Applications in Genomics," *Statistical Applications in Genetics and Molecular Biology*: Vol. 3: Iss. 1, Article 18.
DOI: 10.2202/1544-6115.1069

Deletion/Substitution/Addition Algorithm in Learning with Applications in Genomics

Sandra E. Sinisi and Mark J. van der Laan

Abstract

van der Laan and Dudoit (2003) provide a road map for estimation and performance assessment where a parameter of interest is defined as the risk minimizer for a suitable loss function and candidate estimators are generated using a loss function. After briefly reviewing this approach, this article proposes 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. This algorithm provides us with a new class of loss-based cross-validated algorithms in prediction of univariate outcomes, which can be extended to handle multivariate outcomes, conditional density and hazard estimation, and 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 transcription factor binding sites in yeast gene expression experiments.

KEYWORDS: Cross-validation, estimation, loss function, measures of importance, model selection, polynomial regression, prediction, risk, variable selection

Author Notes: We thank Sunduz Keles for formulating the data analysis problem in a feature selection framework and providing us with the data. We are grateful to the reviewers for making helpful comments and offering useful suggestions to improve this paper. Sandra Sinisi was supported by an NIH Genomics training grant (5 T32 HG00047), and Mark van der Laan was supported in part by the NIH under grant number R01 GM67233.

1 Introduction.

This article introduces the Deletion/Substitution/Addition (D/S/A) algorithm by demonstrating how it works in linear regression with polynomial basis functions and illustrates the utility of this method in genomics by applying it to the detection of transcription factor binding sites in a publicly available dataset of the yeast *Saccharomyces cerevisiae*. The D/S/A algorithm can be used in a variety of settings including prediction of univariate outcomes (setting used in this article), prediction of multivariate outcomes, conditional density and hazard estimation, and can be generalized to censored outcomes such as survival. Our motivation stems from current statistical inference problems in the analysis of genomic data, such as the prediction of biological and clinical outcomes using microarray gene expression measures, the identification of regulatory motifs (i.e., transcription factor binding sites) in DNA sequences, and the genetic mapping of complex traits using single nucleotide polymorphisms (SNPs). One such motivating example is the identification of regulatory motifs in DNA sequences. Transcription factors (TF) are proteins that selectively bind to DNA to regulate gene expression. The transcription factor binding sites, or regulatory motifs, are short DNA sequences (5-25 base pairs) in the upstream control region (UCR) of genes, i.e., in regions roughly 600 to 1,000 base pairs from the gene start site (in lower eukaryotes, e.g., yeast). A possible statistical question is to utilize gene expression data to identify sequence motifs associated with genes that are activated under a specified experimental condition. Thus, the estimation problem can be framed as a prediction problem where one is predicting gene expression levels based on sequence features, such as pentamers (and their interactions). Making statistical inferences based on voluminous, genomic data involves exploring many different relationships (to account for high-order interactions) amongst a vast array of explanatory variables and a single outcome or multiple outcomes. As a result, dominating features of statistical inference problems in genomics include a high-dimensional parameter space and deciding upon a reasonable error measure which we wish to minimize. When faced with a high-dimensional parameter space, it becomes difficult to minimize a suitable error measure over the entire parameter space. A unified loss-based estimation framework to approach these problems has been offered by van der Laan and Dudoit (2003).

Given a large parameter space, we want to perform intensive searches over this space and form candidate estimators which address the desired

statistical question. Once we collect all these candidate estimators, we must then select a *best* estimate. The approach of van der Laan and Dudoit (2003) establishes that cross-validation can be used to select among many candidate estimators, even in finite sample situations. Furthermore, it is shown that aggressive searches are adaptive to the truth. van der Laan et al. (2004) propose a cross-validated adaptive ϵ-net estimation methodology where they consider collections of subspaces of the parameter space. For each choice of subspace and resolution, they generate candidate estimators as the empirical risk minimizers over ϵ-*nets*.

Consequently, it becomes necessary to construct an algorithm that is capable of adapting to the data completely to address inference questions satisfactorily. This means that we need an algorithm that is capable of minimizing a suitable error measure, say the empirical mean of a loss function, over an approximation of the complete parameter space, and we need cross-validation to select certain fine-tuning parameters. However, we do not want a nonparametric version of such an algorithm because it will try to fit the data perfectly resulting in estimators that are too variable due to the number of variables involved relative to a limited sample size. Consequently, we want to put certain stops on the algorithm and index it by a number of "brakes." These brakes correspond to specified subspaces of the complete parameter space. A natural collection of brakes can be obtained by parameterizing the complete parameter space in terms of linear combinations of basis functions where the choice of a basis, the number of basis functions, complexity measure(s) on the basis functions, and a constraint on the vector of coefficients (e.g., norm) provide natural choices for brakes. Based on the cross-validation results given by van der Laan and Dudoit (2003), 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. Hence, such notions as aggressive searches, using basis functions to parameterize large parameter spaces, applying certain brakes, and cross-validation selection led us to construct algorithms which incorporate these ideas to answer statistical questions in genomics.

In this article, we propose a D/S/A algorithm for minimizing empirical risk over a subspace. In the context of regression, we will study the performance of the D/S/A algorithm in simulation studies. Finally, we apply the methodology to a yeast data set to detect transcription factor binding sites.

DOI: 10.2202/1544-6115.1069

2 Review of Estimation Road Map.

This section briefly explains the concepts of loss-function based estimation and cross-validation selection which are important components of the D/S/A algorithm. Details on theoretical aspects behind the D/S/A algorithm are available in Sinisi and van der Laan (2004) and Dudoit et al. (2003). A more thorough description of loss-function based estimation is available in van der Laan and Dudoit (2003) and van der Laan et al. (2004).

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. The expectation of this loss function is minimized at ψ_0:

$$\begin{aligned} {}_0 &= \text{argmin}_{\psi \in \mathbf{\Psi}} \int L(o, \psi) dP_0(o) \\ &= \text{argmin}_{\psi \in \mathbf{\Psi}} E_0 L(O, \psi). \end{aligned} \tag{1}$$

In univariate outcome regression, we have $O = (Y, W) \sim P_0$, where Y is a scalar outcome and W is a vector of covariates. The parameter of interest is the conditional expected value, $\psi_0(W) \equiv E_{P_0}(Y \mid W)$, of the outcome Y given covariates W. We can use as loss function the *quadratic loss function*:

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

also known as the *squared error loss function* or the L^2 *loss function*.

Parameterization of the parameter space.

Having defined the parameter of interest as the risk minimizer for the squared error loss function, the next task is to generate a sequence of candidate estimators by minimizing the empirical risk over subspaces of increasing dimension approximating the complete parameter space $\mathbf{\Psi}$. We propose to parameterize $\mathbf{\Psi}$ in terms of tensor products of basis functions with polynomial basis functions. Given a d-vector $\vec{p} = (p_1, \ldots, p_d) \in \mathbb{N}^d$, we denote the polynomial basis functions by $\phi_{\vec{p}}(W) = W_1^{p_1} \ldots W_d^{p_d}$ where the collection $\{\phi_{\vec{p}} : \vec{p} \in \mathbb{N}^d\}$ provides a basis for the complete parameter space $\mathbf{\Psi}$. The index set $I \subset \mathcal{I}$ represents a set of elements in $\mathbb{N}^d$.

Next, we define a collection of subspaces $\mathbf{\Psi}_s \subset \mathbf{\Psi}$, indexed by s ranging over a set $\mathcal{A}_n$. Such subspaces can be obtained by restricting the subsets I

of basis functions to be contained in $\mathcal{I}_s \subset \mathcal{I}$:

$$\mathbf{\Psi}_s = \left\{ \sum_{\vec{p} \in I} \beta_{\vec{p}} \phi_{\vec{p}} \in \mathbf{\Psi} : m(I) \leq s \right\}, \tag{2}$$

where $m(I) = (m_1(I), \ldots, m_q(I))$ is defined as a q-valued function such that $m_1(I) \leq s_1, \ldots, m_q(I) \leq s_q$. In this article, $m_1(I) = |I|$ 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.

Construction of candidate estimators.

After having defined our subspaces, $\mathbf{\Psi}_s$, we would like to find the minimizer of the empirical risk over the subspace for each $s \in \mathcal{A}_n$. 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, which in regression corresponds to minimizing the sum of the squared residuals over the linear regression model in the basis functions $\phi_{\vec{p}}$ indexed by $\vec{p} \in I$. For each I this results in an estimator $\Psi_{I,s}(P_n) \equiv {}_{I,\beta(P_n|I,s)}$.

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

$$f_{E,s}(I) \equiv \int L\left(O, \Psi_{I,s}(P_n)\right) dP_n(O). \tag{3}$$

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 D/S/A algorithm that seeks to calculate $I_s(P_n)$.

Selection among candidate estimators: Cross-validation.

Now, we have the empirical risk minimizer, denoted by $\hat{\Psi}_s(P_n)$, for each choice of subspace s. The final task is to select s with cross-validation.

DOI: 10.2202/1544-6115.1069

To derive a general representation for cross-validation, let $B_n \in \{0,1\}^n$ be a random vector whose observed value defines a split of the observed data $O_1, \ldots, O_n$, the 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. 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 by $p = \sum_i B_n(i)/n$. 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 our parameter of interest is given by $\hat{\Psi}(P_n) \equiv \Psi_{s(P_n)}(P_n)$.

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 and its 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.

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

In this section, 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 approach are summarized below for the case where the index sets are subsets of $\mathbb{N}^d$, as is the case for tensor product polynomial basis functions $\phi_{\vec{p}}$. Note that the D/S/A algorithm has been adapted to histogram regression with partition-specific indicator basis functions (Molinaro and van der Laan, 2004) and to neural networks(Durbin and Dudoit, 2004).

To simplify notation, in this section we will suppress dependence of quantities on s; let $\mathcal{I}$ denote the collection of allowed index sets. Let s_0 denote the dimension d and assume that $s = (s_1, \ldots, s_q)$, where s_1 denotes the upper bound on the size of the index sets (i.e, the number of allowed basis functions), while $s_2, \ldots, s_q$ represent the remaining fine tuning parameters. The D/S/A algorithm described below aims to calculate $I_s(P_n)$ for each choice of s_1, given the remaining components of s. Thus, one has to carry out this algorithm for each value of $s_2, \ldots, s_q$ to obtain all optimal index sets $\{I_s(P_n) : s\}$ and thereby our collection of s-specific estimators $\hat{\Psi}_s(P_n)$. Throughout this section, we use k to represent s_1 where $s_1 = |I|$.

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 sub-

DOI: 10.2202/1544-6115.1069

stitution moves the *alternate-substitution* moves. The alternate-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 alternate-substitution moves) can 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 an obvious need for this is not seen presently.

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 our 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 $\hat{\Psi}_I(P_n)$

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

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, described in Box 1, returns for each k, an index set $I_k(P_n)$ that aims to approximate (or equal) $I_k^\star(P_n)$.

DOI: 10.2202/1544-6115.1069

Box 1. 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" by always carrying out the addition move.

Unlike many previously proposed forward/backward selection approaches, the 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. Similarly, this can be said about Logic Regression.

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 want also to 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 alternate moves: $W_2W_3W_4 + W_2W_4^5$, $W_1W_3W_4 + W_2W_4^5$, $W_1W_2W_4 + W_2W_4^5$.

DOI: 10.2202/1544-6115.1069

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

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

3.2 Available Options.

Dimension Reduction. Depending on the application at hand, it can be worthwhile to transform and/or reduce the number of given explanatory variables. To reduce the data, we compute d T-statistics corresponding to the main effects of $W_1, \ldots, W_d$ by fitting d univariate regressions. Next, we rank these statistics, possibly in absolute value, in decreasing order $\hat{R}(1), \ldots, \hat{R}(d) \subset \{1, \ldots, d\}$ yielding our ordered covariates $W_{\hat{R}(1)}, W_{\hat{R}(2)}, \ldots, W_{\hat{R}(d)}$. Then, one can input the set $(W_{\hat{R}(1)}, \ldots, W_{\hat{R}(s_0)})$, of length s_0, as the vector of covariates into the D/S/A algorithm and can choose whether or not to select s_0 via cross-validation. This reduction was done in Section 5.

Derivative-based importance measures. In prediction problems, a common and practical question is to assess the *importance* of a variable, or set of variables, in terms of its predictive ability for the outcome of interest. For instance, in microarray experiments, one is interested in determining how

important each gene (or set of genes) is for the prediction of a particular biological or clinical outcome. Measures of variable importance can assist in the identification of a subset of marker genes for the outcome.

Various measures of importance exist in the literature including loss-function based importance measures (Dudoit et al., 2003), and we will describe the measure that we used in Section 5.

As before, let the data be n observations of (Y, W), where Y is the outcome of interest and W is a d-dimensional vector of covariates for which we would like a measure of importance. Let $h(W) = E(Y|W)$.

Our proposal to measure variable importance is based on the idea of counterfactual variables in the causality literature (van der Laan and Robins, 2003). In causal inference, one is interested in the causal effect of assigning treatment A. If A is randomized in a study within strata of W, then $h(a) = E_W E(Y|A = a, W) = E(Y_a)$. Y_a is referred to as a *counterfactual variable*. The derivative of $h(a)$ represents a causal effect where we are trying to measure the remaining effect of A after having fully adjusted for potential confounders.

Without using the A notation, we are getting a sense of the importance of variable W_j for $j = 1, \ldots, d$ by seeing what happens when $W = w$ for a given variable of interest. Given a particular b-specific fit $\hat{h}_b(W)$ for $b = 1, \ldots, B$, let $\bar{h}_{jb}(w) = \frac{1}{n}\sum_i \hat{h}_b(W_{1,i}, \ldots, W_{j-1,i}, w, W_{j+1,i}, \ldots, W_{d,i})$. The importance measure which aims to measure the "causal" effect of W_j, for the following types of variables, can be estimated as follows:

- Continuous
  $\hat{\alpha}_b(j) = \frac{\int_{w \in \mathcal{W}_j} |\frac{d}{dw}\bar{h}_{jb}(w)| dw}{\int_{w \in \mathcal{W}_j} dw}$ where $\mathcal{W}_j$ represents the set of possible values of W_j.

- Binary
  $\hat{\alpha}_b(j) = \mid \bar{h}_{jb}(1) - \bar{h}_{jb}(0) \mid$

- General Discrete
  $\hat{\alpha}_b(j) = \frac{\sum_{l=1}^{K-1} |\bar{h}_{jb}(l+1) - \bar{h}_{jb}(l)|}{K-1}$ where $w \in \{1, \ldots, K\}$.

The final estimate of the importance measure is then a weighted average of $\hat{\alpha}_b(j)$ across many b-specific fits. Various approaches for obtaining b-specific

DOI: 10.2202/1544-6115.1069

fits $\hat{h}_b$ can be considered. One approach is to use the fits for all $s \in \mathcal{A}_n$ and estimate $\hat{\alpha}_s(j)$ for all s. Another approach is to reduce the data to $V \subset W$ reasonably important variables. Then form B random subsets of variables of a specified size from V. Let $S_b \in \{1, \ldots, d\}$ identify these subsets. Then for a given variable, its importance measure is estimated across fits as:

$$\hat{\alpha}(j) = \frac{\sum_{b=1}^{B} \hat{\alpha}_b(j) I(j \in S_b) \mathrm{wt}_b}{\sum_{b=1}^{B} I(j \in S_b) \mathrm{wt}_b} \tag{4}$$

In equation (4), wt represents a weight for a particular fit which can be the cross-validated risk of the given regression model for example. For instance, to measure variable importance in Section 5, we reduced the data to the 10 most important variables based on univariate regressions and formed all subsets of size 3 from these 10 variables.

3.3 Other Approaches.

The estimation road-map offered by van der Laan and Dudoit (2003) and van der Laan et al. (2004) inspired the D/S/A algorithm and lists as its three main steps:

1. Definition of the parameter of interest in terms of a loss function.

2. Construction of candidate estimators based on a loss function.
   Define a finite collection of candidate estimators for the parameter of interest based on a sieve of increasing dimension approximating the complete parameter space. For each element of the sieve, the candidate estimator is chosen as the minimizer of the empirical risk based on the observed data loss function.

3. Cross-validation for estimator selection and performance assessment.

The D/S/A algorithm is needed to carry out Step 2. It then uses cross-validation to select the optimal estimator among the candidates it formed.

There are many approaches to regression problems in the statistics and machine learning literature which can produce candidate estimators (Breiman et al. (1984), Friedman (1991), Ruczinski et al. (2003), Efron et al. (2004)). The results for cross-validation given by van der Laan and Dudoit (2003)

are very general, and hence their results can be applied to the candidate estimators generated by any type of algorithm, e.g., forward/backward type algorithms. The D/S/A algorithm does not differ in its use of cross-validation but in the way it produces candidate estimators.

Friedman (1991) introduced MARS, an adaptive procedure for regression, which uses linear splines as basis functions. Stone et al. (1997) developed a hybrid of MARS. Support Vector Machines (SVMs) (Vapnik, 1995) have been well-promoted for classification and have been adapted for regression, sometimes referred to as support vector regression (Shawe-Taylor and Cristianini, 2004). To use the SVM, one needs to define tuning parameters such as the regularization cost parameter and kernel parameters. Hastie et al. (2004) argue that the choice of the cost parameter can be critical and derived an algorithm that can fit the entire path of SVM solutions for every value of the cost parameter. For the two-class SVM model (i.e., classification), Hastie et al. (2004) offer the `SvmPath` software package for `R`. Logic Regression (Ruczinski et al., 2003) is an adaptive regression method and performs a very thorough search by allowing many different moves in its tree growing process. It works in a specialized context, to construct predictors as Boolean combinations of binary covariates.

However, many existing methods for constructing candidate estimators are not aggressive enough for the types of datasets encountered in genomics. They either only accommodate variable main effects or are too rigid to generate a good set of candidate estimators. These approaches do not aim to minimize the empirical mean of a loss function over specified subspaces of the complete parameter space. Instead, they rely on forward/backward-like local optimization steps. For example, while regression trees allow interactions among variables, the candidate tree estimators are generated according to a limited set of moves, amounting to forward selection (node splitting) followed by backward elimination (tree pruning).

The D/S/A algorithm keeps running because it can always improve the objective function by adding a term or in some cases by altering an existing term. Once it adds a term, it then has the ability to try to delete a term or alter a term and can avoid getting "stuck." It performs a very aggressive search. Logic Regression also performs an aggressive search. Yet, many other methods do not perform thorough enough searches.

We feel that the current literature on other approaches, with the exception of Logic Regression, do not focus on algorithms which minimize over specified subspaces and are not aggressive enough when applying cross-validation to

DOI: 10.2202/1544-6115.1069

select various fine-tuning parameters. The D/S/A algorithm provides a general class of algorithms filling this gap. In the next section, we compare our approach to Logic Regression, MARS, and SVMs. Two implementations of the D/S/A algorithm for histogram regression (Molinaro and van der Laan, 2004) and neural networks (Durbin and Dudoit, 2004) are being developed.

4 Simulations.

In this section, we will present the results of applying the D/S/A algorithm to simulated data sets. All simulations and data analyses were done using machines available in the labs of the Statistical Computing Facility (University of California, Berkeley). These are Sun workstations which have UltraSparc II processors ranging in speed from around 200 MHz to 440 MHz. They typically have 128 MB to 256 MB of RAM. On these machines, it takes about 0.06 seconds to fit one model. The algorithm loops through deletion, substitution, and addition moves until it reaches a maximal size model denoted here by K and let $k = |I_0|$. Deletion moves will take $0.06 \times k$ seconds; substitution moves will take $0.06 \times (k \times s_2 \times 2d)$ seconds; addition moves will take $0.06 \times (k \times s_2 \times 2d + d)$ seconds to run (if imposing the constraint involving s_2) for all $k \leq K$. Runtime depends strongly on the magnitude of K relative to d.

We conducted a variety of simulations to see how the algorithm performs in different settings. The first set of simulations use the D/S/A algorithm without cross-validation where we check to see if it can find the truth, then we impose a constraint on size and choose it with cross-validation. Later simulations impose additional constraints on the algorithm. Dimension reduction is done for the data analysis (Section 5).

The D/S/A algorithm first is implemented without constraints (brakes); we are not restricting the number of tensor products, k, and thus not using cross-validation to select k. We run the algorithm until we reach a k that gives a minimal residual sum of squared error (RSS). Next, the algorithm is implemented with a cross-validated constraint placed on the number of tensor products in the regression, i.e., selecting k via v-fold cross-validation. The cross-validated D/S/A algorithm (denoted by *DSA1-CV*) is then compared to: the R function `stepAIC()`, forward selection with cross-validation (*fscv*), Logic Regression, Multivariate Adaptive Regression Splines (MARS), and Support Vector Machines (SVMs). Finally, the algorithm which places brakes

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 *alternate-substitution* moves (denoted by *DSA2-CV*) is implemented and used in the Logic Regression and MARS comparisons.

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. 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 (see Sinisi and van der Laan (2004)).

Numerical results obtained from this simulation are available in (Sinisi and van der Laan, 2004). We found that the algorithm succeeded in minimizing $f_{E,s}(I)$; both sensitivity and specificity is 100% 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.

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 were run only once.

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

DOI: 10.2202/1544-6115.1069

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

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

Table 1 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 1, 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 $d = 5$.

$E[Y\|W]$	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%	100%	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 1: **DSA unconstrained.** Comparing $\varepsilon = 0$ and $\varepsilon \sim \mathcal{N}(0,1)^*$. *sens*: sensitivity, *spec*: specificity, RSS_n: $RSS/(n-p)$ represents the estimate of the variance of the error where p 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()`. We first generated 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. Then, we manually generated the following three true regression models:

$$E_1[Y|W] = W_1 + W_2^2$$

$$E_2[Y|W] = W_1 W_3$$

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

Using the models, we next generated the outcome Y with a Gaussian error with mean 0 and standard deviation 1. Then ran the procedure once with the cross-validated D/S/A algorithm (*DSA1-CV*) and `stepAIC` and reported 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 (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 2), both our method and `stepAIC` selected the exact true model. However, in the next two simulations (rows 2-3, table 2), our method fitted the truth exactly while `stepAIC` heavily overfitted the model (col 4, table 2). AIC's tendency to over-fit is well-known in the statistical literature, but 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.

4.3 Comparison to forward selection with cross-validation.

This simulation study (Table 3) compares the D/S/A algorithm (*DSA1-CV*) to a type of forward selection with cross-validation (fscv) algorithm. The forward selection algorithm makes all the same *addition* moves as our method

DOI: 10.2202/1544-6115.1069

$E[Y\|W]$	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 2: 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.

but does not carry out the deletion and substitution moves.

The true model, (Table 3), is $y = 4w + 3w^3 - 2w^5 + \varepsilon$, where $w \sim U(1, 5)$, ε is normal with mean 0 and standard deviation 1, and $y \in (-6000, 8)$. Both methods were run under 2, 5, and 10-fold cross-validation with the maximum number of terms in the model pre-set at 10.

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.

4.4 Logic Regression.

Logic Regression (Ruczinski et al., 2003) is a very useful regression method currently available, and it can handle a variety of problems including linear regression, logistic regression and classification and can be extended to other problems by defining an appropriate score function. Both the D/S/A algorithm and Logic Regression 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

Table 3: *Simulation study* **FSCV Comparison**. 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

DOI: 10.2202/1544-6115.1069

each tree, which corresponds to the complexity of our tensor products (implemented by *DSA2-CV*, Table 4). 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 Table 4.

DSA1-CV has been described previously; it uses cross-validation to select k, the number of tensor products. *DSA2-CV* uses cross-validation to select the number of tensor products and places a brake on the complexity of each tensor product thereby involving the alternate-substitution moves. Specifically, we are limiting the order of interactions to be no greater than a specified value, $m_2(I) = \max_{\vec{p} \in I} \sum_{j=1}^{d} I(p_j \neq 0) \leq s_2$. In the case of binary covariates, $m_2(I) \equiv m_3(I)$ and thus we did not need to select s_3 via cross-validation.

The true model was generated 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 $er \sim N(0, 1)$. It has been pointed out that the model can be reduced to:

$$\beta_1(w_3(1 - w_2)) + (\beta_2 - \beta_1)(w_3(1 - w_1)(1 - w_2)) + \beta_3(w_7 w_{10})$$

or

$$\beta_2(w_3(1 - w_2)) + (\beta_2 + \beta_1)(w_3 w_1(1 - w_2)) + \beta_3(w_7 w_{10}).$$

Thus, the true number of leaves is 7.

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 number of leaves or tensor product complexity measure. When $n = 1000$, Logic Regression picked the truth with 3 total trees and 7 total leaves for all ten repetitions. The D/S/A algorithm picked a final model with 5 terms. In terms of prediction, this simulation shows that the D/S/A algorithm is competitive with Logic Regression since the risk ratios are roughly one. When using binary covariates, both methods produce models that are easy to interpret. Logic Regression provides the user with enhanced interpretability by using intuitive operators. See (Sinisi and van der Laan, 2004) for two other comparisons to Logic Regression.

4.5 Multivariate Adaptive Regression Splines.

Multivariate Adaptive Regression Splines (MARS) (Friedman, 1991) is a method for flexible regression modeling of high-dimensional data. It can

Table 4: *Simulation study* **Logic Regression Comparison**. 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 of the 10 risks (with the L_2 loss function) for each method, col 5 is the standard deviation of the risks over the 10 reps, and col 6 is the ratio of averaged risks (col 4) − optimal risk, (dsa/logic).

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

DOI: 10.2202/1544-6115.1069

be viewed as a generalization of stepwise linear regression or a modification of the CART (Breiman et al., 1984) method. MARS uses expansions in the form of linear splines.

To compare the D/S/A algorithm to MARS, we used the `mars{mda}` `R` function to run MARS.

Data was simulated from two different models (Tables 5 and 6). The first was taken from Section 4.3 of the MARS paper (Friedman, 1991):

$$y = 10\sin(\pi w_1 w_2) + 20(w_3 - \frac{1}{2})^2 + 10w_4 + 5w_5 + er,$$

where $w_i \sim \mathcal{U}(0,1)$, $1 \leq i \leq 10$ and $er \sim N(0,1)$.

The second model is a variation of the first so that it does not include any main terms:

$$y = 10\sin(\pi w_1 w_2) + 20(w_3 - \frac{1}{2})^2 + 10w_4 w_5 w_6 + er,$$

where $w_i \sim \mathcal{U}(0,1)$, $1 \leq i \leq 10$ and $er \sim N(0,1)$.

Both methods allow the user to set an optional integer specifying the maximum interaction degree (`degree`) and number of model terms (`nk`). To compare both methods under the same level of constraint, we set `nk` to 10 or 15 and `degree` to two (for model one) and to three (for model two). In Tables 5 and 6, the number in parenthesis following the name of the method (dsa versus mars) refers to `nk`. We used 10-fold cross-validation to run the D/S/A algorithm. We also used Support Vector Machines (SVMs) on these two simulated examples where we ran the SVM using the `svm{e1071}` `R` function with the default arguments.

The results for 50 repetitions comparing the estimated risks of each method based on an independent test set of 10,000 are displayed in Table 5 for the first example and in Table 6 for the second example. The SVM results are not shown in the tables but the estimated risks (and the corresponding standard deviation) for the two sample sizes are:

- Example One
  $N = 250$: 5.171 (0.23)
  $N = 500$: 3.694 (0.19)
- Example Two
  $N = 250$: 4.659 (0.25)

Table 5: *Simulation study* **I. MARS Comparison**. Data simulated from $y = 10\sin(\pi w_1 w_2) + 20(w_3 - \frac{1}{2})^2 + 10w_4 + 5w_5 + er$, where $w_i \sim \mathcal{U}(0,1)$, $1 \leq i \leq 10$ and $er \sim N(0,1)$. Candidate estimator was chosen over 50 repetitions of two sample sizes (col 1) for our algorithm versus MARS (col 2). The results (cols 3-6) in the table are based on an independent test sample of $n = 10000$. col 3 is the average size (number of basis functions for ours and MARS), col 4 is the average of the 50 risks (with the L_2 loss function) for each method, col 5 is the standard deviation of the risks over the 50 reps, and col 6 is the ratio of averaged risks (col 5) − optimal risk, (dsa/mars).

Sample		50 Repetitions			
Size	Method	avg size	mean risk	std dev	ratio
250	dsa (10)	9.38	2.614	1.6	1
	mars (10)	6.44	5.633	.22	.348
	dsa (15)	13.72	1.904	1.1	1
	mars (15)	11.74	1.612	.14	1.477
500	dsa (10)	9.48	2.294	1.4	1
	mars (10)	6.20	5.425	.11	.292
	dsa (15)	13.60	1.505	.92	1
	mars (15)	12.28	1.454	.091	1.112

$N = 500$: 3.379 (0.15)

These results are given by the default options of the `svm()` function. Fine-tuning the svm with the appropriate kernel may produce *better* results. In comparison to the svm's default settings, the D/S/A algorithm produced better risk estimates. It is of interest for future work to optimize the regularization path in support vector regression, as has been done for the two-class SVM (Hastie et al., 2004).

The choice of `nk` has an effect on the results for example one (Table 5). When `nk` = 10, the D/S/A algorithm outperforms MARS. However, MARS produces more consistent results (smaller risk variance). The D/S/A algorithm has the ability to produce models with very low risk, but it is unable to do this for every repetition under the imposed constraints. Yet, when `nk` = 15, MARS tends to pick models with a lower risk estimate. The aggres-

DOI: 10.2202/1544-6115.1069

Table 6: *Simulation study* **II. MARS Comparison**. Data simulated from $y = 10\sin(\pi w_1 w_2) + 20(w_3 - \frac{1}{2})^2 + 10 w_4 w_5 w_6 + er$, where $w_i \sim \mathcal{U}(0,1)$, $1 \leq i \leq 10$ and $er \sim N(0,1)$. Candidate estimator was chosen over 50 repetitions of two sample sizes (col 1) for our algorithm versus MARS (col 2). The results (cols 3-6) in the table are based on an independent test sample of $n = 10000$. col 3 is the average size (number of basis functions for ours and MARS), col 4 is the average of the 50 risks (with the L_2 loss function) for each method, col 5 is the standard deviation of the risks over the 50 reps, and col 6 is the ratio of averaged risks (col 5) − optimal risk, (dsa/mars).

Sample		50 Repetitions			
Size	Method	avg size	mean risk	std dev	ratio
250	dsa (10)	9.42	4.434	1.2	1
	mars (10)	7.70	4.595	.27	.955
	dsa (15)	14.23	1.927	1.0	1
	mars (15)	11.16	2.871	.21	.495
500	dsa (10)	9.50	1.827	1.1	1
	mars (10)	7.96	4.451	.17	.240
	dsa (15)	14.5	1.333	.19	1
	mars (15)	11.00	2.744	.18	.191

siveness of the D/S/A algorithm leads to more variable estimators with the increase in model size.

The next example does not involve main terms, and we would expect MARS to be less efficient as a result (Table 6). For both sample sizes and `nk` choices, the D/S/A algorithm outperforms MARS in terms of risk. The gain by using the D/S/A algorithm increases with sample size.

5 Data Analysis.

An important problem in contemporary biology is transcription factor binding site identification. The activities of hundreds of sequence specific DNA binding proteins, transcription factors (TFs), play an important role in transcriptional regulation of eukaryotes. TFs are proteins, needed to initiate the

transcription of a gene, that bind to regions in the vicinity of genes and as a result regulate the activities of the genes. Each TF, or group of closely related factors, recognizes a unique grouping of short sequence elements, usually between five and fifteen basepairs in length. 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. In this section, we look at the identification of biologically significant transcription factor binding sites in the genome of the yeast *Saccharomyces cerevisiae*.

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 length five motifs, or 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.

5.1 Cell Cycle Data.

The eukaryotic cell cycle consists of four phases: M (mitosis), S (synthesis, DNA is replicated), G_1, and G_2. During the first gap phase, G_1, cells increase in size, produce RNA, and synthesize protein, and there is a checkpoint ensuring that everything is ready for DNA synthesis. During the gap between DNA synthesis and mitosis, G_2, the cell will continue to grow, produce new proteins, and determine if the cell can proceed to enter mitosis and divide.

Cho et al. (1998) gathered data by using Affymetrix oligonucleotide microarrays to query the abundances of 6,220 mRNA species in synchronized *Saccharomyces cerevisiae* batch cultures. Cells were collected at 17 time points taken at 10 minute intervals to cover nearly two full cell cycles. The time course was divided into early G_1, late G_1, S, G_2, and M phases (G_1-S for replication, S-G_2 for organization of centrosome, and M phase for budding and cell polarity).

For this data analysis, we used 15 of the 17 time points. Time points 90 and 100 minutes were excluded due to the less efficient labeling of their mRNA during the original chip hybridizations (Tavazoie et al., 1999). The outcome was the normalized expression profiles of the most variable 3,000 ORFs. With the 15 time points, we constructed a 3,000 by 15 outcome data matrix.

DOI: 10.2202/1544-6115.1069

In yeast, regulatory elements are found almost exclusively upstream from the promoter. There are several known upstream regulatory sequences involved in cell cycle-dependent transcription including the late G_1 elements MCB (MIuI cell cycle box) and SCB (Swi4/6 cell cycle box) and the early G_1 element ECB (early cell cycle box) (Cho et al., 1998). The SCB element has been identified as a regulatory sequence located upstream of genes transcribed in late G_1 and early S. SCB is bound by the SBF transcription factor, a complex of Swi4p and Swi6p (Wolfsberg et al., 1999). The Hap complex is formed by four proteins (Hap2p, Hap3p, Hap4p, and Hap5p); Hap2p and Hap3p have been shown to bind DNA, and Hap4p acts as activation domain for the complex. The Hap2p-Hap3p binding site contains the conserved motif $CCAAT/C$ (van Helden et al., 1998).

Upstream regions of all genes should be searched for other known yeast regulatory sequences, such as the ABF1 and RAP1 transcription factor binding sites, the stress response element STRE with consensus sequence AGGGG, and the SFF factor which acts with MCM1 to control cell cycle regulated genes. Other cell cycle period-specific transcription factors such as Swi5 and MCM1 do not have a highly conserved binding sequence, making it difficult to search genomic sequence for possible action sites accurately.

5.2 Applying the D/S/A algorithm.

Previous work on cell cycle regulation in yeast suggests that more than one sequence element may be responsible for transcription at the same phase (e.g., SCB and MCB both regulate late G_1 mRNA expression). Wolfsberg et al. (1999) predicts that a variety of elements can be responsible for transcription at each phase of the cell cycle. It is important to look at interactions between motifs, and the D/S/A algorithm is one adaptive regression approach that can easily search through two-way and multi-way interactions of explanatory variables.

As many transcription factors bind to short, highly conserved stretches of DNA, many analyses focuses on short oligomers of length five or six, pentamers or hexamers. We focus on pentamers for comparison to the results of Keleş et al. (2002). After having retrieved the set of upstream sequences from the regulatory family, the number of occurrences of all oligonucleotides of the selected size, five in our case, are counted. There are 512 distinct pairs of pentamers and reverse complements. For these 512 motifs, we form sequence motif scores which represent a weighted count, between zero and

one, of the number of occurrences of the given motif. We treat the sequence motif scores as explanatory variables, and model gene expression as a function of these pentamers present in presumptive transcription control regions. The D/S/A algorithm is used to extract the pentamers that are most relevant.

The D/S/A algorithm was run three different ways when analyzing the data: (1) select the number of tensor products, s_1, via cross-validation; (2) select the number of tensor products, s_1, and two complexity measures of the tensor products, s_2, s_3, via cross-validation where $m_2(I) = \max \sum_{j=1}^{d} \mathrm{I}(p_j \neq 0) \leq s_2$ and $m_3(I) = \max \sum_{j=1}^{d} p_j \leq s_3$; or (3) select s_0, s_1, s_2, and s_3 via cross-validation, where s_0 represents the dimension of the vector of covariates and can range between 1 and 512. Each implementation ran with a maximum size model of 5 under 2-fold cross-validation. These were run on the Statistical Computing Facility's machines (see Section 4) which share computing power with other users. Shared runtime for the three implementations are: (1) 2 hours from start to finish; and after selecting $\hat{s}$, (2) 3 hours; and (3) 1.4 hours. The runtimes can be reduced if run on faster machines or if given top priority.

The data reduction done in the third implementation ranks the main effects of the 512 given covariates based on the training set. An average of 29 (or fewer pentamers) have significant main effects for $p \leq 0.05$ across the 15 time points while an average of only 8 pentamers have significant main effects at the 0.01 significant p-level. In this particular data set, the reduction steps are essential for reducing the noise by narrowing the candidate covariates down to a significant set.

The variable importance measures calculated using equation (4), where $\hat{\alpha}_b(j)$ is estimated for continuous variables, for each time point are given by Table 8 (displayed for the first four time points). To calculate these measures we reduced the data to the top 10 variables (ranked by $\mid T \mid$) and formed all subsets of 3 variables from these 10 at each time point. Given that the main effects of a limited number of pentamers on average was small relative to the total number of pentamers (29 for $p \leq 0.05$, 8 for $p \leq 0.01$), we chose to form importance measures on the ten most significant pentamers for each time point.

Summary of Previous Results. Keleş et al. (2002) model gene expression on sequence motif scores using a forward and backward stepwise selection method embedded in Monte Carlo cross-validation allowing for main effects and two-way interactions. The scores they use as explanatory variables in-

DOI: 10.2202/1544-6115.1069

corporate the number of occurrences of the motifs and their positions with respect to the gene's translation start site. They begin with pentamers as sequence motifs.

Keleş et al. (2002) report *experiment specific importance measures*, $R_w(n)$, on selected pentamers for the first four time points (0, 10, 20, and 30 minutes). The final model given by their feature selection method is not shown. $R_w(n)$ represents a rank weighted proportion of the number of times that motif w is selected within the total number of splits, selected at random from their method. If a motif has $R_w(n) = 1$, then it entered the model first in all of the splits. A motif that is never selected will have an $R_w(n) = 0$. The three pentamers with the highest $R_w(n)$ are (Keleş et al., 2002):

- T = 0 minutes
  AGGGG/CCCCT [stre]
  ACGCG/CGCGT [mcb]
  GAAAA/TTTTC [ecb]

- T = 10 minutes
  AAACA/TGTTT [ste12]
  CTTAA/TTAAG
  GTTTA/TAAAC [sff]

- T = 20 minutes
  ACGCG/CGCGT [mcb]
  CGCGA/TCGCG [scb]
  AGGGG/CCCCT [stre]

- T = 30 minutes
  ACGCG/CGCGT [mcb]
  CCACA/TGTGG
  CGCGA/TCGCG [scb]

MCB has a measure of one at 20 and 30 minutes.

We restricted our analysis to pentamers for direct comparison to the results of (Keleş et al., 2002), however it is of interest to consider longer motifs. This could be done by incorporating the extension method (Keleş et al., 2002) into the D/S/A algorithm for example.

Table 7: *Yeast Data Analysis* **DSA1-CV, DSA2-CV, DSA3-CV**, 2-fold cross-validation, applied to yeast cell cycle data of (Cho et al., 1998), first time point.

T=0 min
$(s_0 = 512,\ \hat{s}_1 = 5)$ $(AGGGG[stre]) \quad + \quad (ACGCG[mcb])(CGAAA)$ $+ \quad (ATCCC)(CCTTA)(GCAAA) \quad + \quad (AAAAT)(CATCG)(GATGA)$ $+ \quad (ACCCG)(AGGGG[stre])(GAAAA[ecb])$
$(s_0 = 512,\ \hat{s}_1 = 5,\ \hat{s}_2 = 3,\ \hat{s}_3 = 3)$ $(AGGGG[stre]) \quad + \quad (ACGCG[mcb])(CGAAA)$ $+ \quad (ATCCC)(CCTTA)(GCAAA) \quad + \quad (AAAAT)(CATCG)(GATGA)$ $+ \quad (ACCCG)(AGGGG[stre])(GAAAA[ecb])$
$(\hat{s}_0 = 55,\ \hat{s}_1 = 5,\ \hat{s}_2 = 3,\ \hat{s}_3 = 3)$ $(AGGGG[stre]) \quad + \quad (ACGCG[mcb])$ $+ \quad (ATCCC)(CCTTA) \quad + \quad (CATCG)$ $+ \quad (ACCCG)(AGGGG[stre])(GAAAA[ecb])$

DOI: 10.2202/1544-6115.1069

5.3 Results.

Results given by the three implementations of the D/S/A algorithm for the first time point are given in Table 7. Looking at the results for $T = 0$ minutes, the first two reported models (DSA1: $\hat{s}_1 = 5$; DSA2: $\hat{s}_1 = 5, \hat{s}_2 = 3, \hat{s}_3 = 3$) are identical and composed of five terms: a main effect involving the pentamer $AGGGG$ and/or its reverse complement (N.B. a pentamer can refer to itself and/or its reverse complement), a two-way interaction of $ACGCG$ with $CGAAA$, and three three-way interactions. Pentamers have partial or exact matches to the regulatory elements shown in brackets after the pentamer. The third reported model reduced the data to 55 covariates and produced a similar, yet simplified, model. The two models contain STRE, MCB, and ECB which are the most highly ranked motifs at $T = 0$ based on the work of Keleş et al. (2002). The method was able to select biologically relevant pentamers, and perhaps, other identified pentamers play a role in predicting gene expression.

Importance measures for the ten pentamers with the highest ranked univariate T-statistic at the first four time points are displayed in Table 8. $ACGCG/CGCGT$ [MCB], $CGCGA/TCGCG$ [SCB], and $AGGGG/CCCCT$ [STRE] have the highest overall importance measure. This coincides with what Keleş et al. (2002) found. In late G_1, about 20 and 30 minutes, MCB has the highest importance measure and SCB has the second highest importance measure, as expected. ECB is known to be relevant at early G_1. It was selected at 0 minutes from the D/S/A algorithm, but it did not make the top ten variables.

Wolfsberg et al. (1999) identified pentamers and hexamers as potential regulatory motifs by analyzing upstream control regions (UCRs) of the genes that might have been involved in the cell-cycle dependent transcription regulation. Comparing their significant pentamers ($p \leq 0.05$) to ours, seven pentamers listed at 20 minutes and six pentamers listed at 30 minutes correspond to the seven significant pentamers they listed for late G_1. Keleş et al. (2002) found that most of the late G_1 pentamers identified by Wolfsberg et al. (1999) were picked up by their method as well. However, their results disagreed at other phases. The pentamers that we selected at different phases also differ from the results of Wolfsberg et al. (1999). In conclusion, it seems that the D/S/A algorithm worked reasonably well as a basis for this analysis.

Table 8: *Yeast Data Analysis* **Variable Importance Measures**

$T = 0$ min.		
Index	Pentamer	VIM
157	AGGGG CCCCT [stre]	11.54
42	AAGGG CCCTT	6.71
98	ACGCG CGCGT [mcb]	8.84
310	CCTTA TAAGG	5.23
82	ACCCC GGGGT	7.62
424	GCCCC GGGGC	7.41
192	ATCCC GGGAT	7.24
264	CATCG CGATG	8.11
455	GGGGA TCCCC	4.52
328	CGCGA TCGCG [scb]	7.43

$T = 10$ min.		
Index	Pentamer	VIM
4	AAACA TGTTT [ste12]	2.81
16	AACAA TTGTT [sff]	1.02
17	AACAC GTGTT	2.75
455	GGGGA TCCCC	3.86
157	AGGGG CCCCT [stre]	2.94
329	CGCGC GCGCG	1.95
318	CGAGA TCTCG	2.38
72	ACAGG CCTGT	3.57
254	CAGGA TCCTG	2.12
479	GTTTA TAAAC [sff]	1.16

DOI: 10.2202/1544-6115.1069

$T = 20$ min.		
Index	Pentamer	VIM
98	ACGCG CGCGT [mcb]	21.55
328	CGCGA TCGCG [scb]	17.49
397	GACGC GCGTC	9.20
25	AACGC GCGTT	5.24
428	GCGAA TTCGC [scb]	2.43
157	AGGGG CCCCT [stre]	7.19
42	AAGGG CCCTT	4.89
312	CGAAA TTTCG [scb]	1.30
242	CACGA TCGTG	0.97
433	GCGTA TACGC	2.33

$T = 30$ min.		
Index	Pentamer	VIM
98	ACGCG CGCGT [mcb]	10.51
328	CGCGA TCGCG [scb]	7.02
25	AACGC GCGTT	4.23
397	GACGC GCGTC	3.69
433	GCGTA TACGC	2.60
273	CCACA TGTGG	2.07
238	CACAG CTGTG	1.63
282	CCCAC GTGGG	0.02
428	GCGAA TTCGC [scb]	0.95
362	CTCCA TGGAG	2.15

6 Discussion.

The D/S/A algorithm has been developed as a general tool for loss-based estimation inspired by the theoretical results of van der Laan et al. (2004). It is completely defined by the following choices: the loss function; the basis functions defining the parameterization of the parameter space; and the sets of deletion, substitution, and addition moves. As a result, by choosing the appropriate loss function, it can deal with problems such as multivariate prediction and density/hazard estimation. The D/S/A algorithm, as presented in this article, has been implemented in the context of polynomial regression for the prediction of a univariate outcome. The current implementation of the D/S/A algorithm has the following options available: (1) to choose the number of basis functions by v-fold cross-validation, (2) the option to reduce the data based on univariate regressions to have no more than s_0 candidate covariates where s_0 can be chosen via cross-validation, (3) 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, (4) 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, and (5) the option to report variable importance measures.

Many regression procedures exist including Logic Regression (Ruczinski et al., 2003) and MARS (Friedman, 1991). Logic Regression is a method freely available in `R` to find interactions between binary inputs associated with an output. It introduced the idea of *substitutions* by defining a number of permissible moves in its tree growing process. It can be adapted to other statistical problems by using the appropriate score function. When faced with binary explanatory variables, Logic Regression is a nice tool to use both for its predictive capabilities and its ease of interpretation. 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. MARS first builds a model with its forward moves, and at the end of that process, a backward deletion procedure is applied. Barron and Xiao conclude that 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). Polynomials served as a practical way to represent

DOI: 10.2202/1544-6115.1069

one version of the D/S/A algorithm. However, it is of interest to implement the D/S/A algorithm using spline basis functions and see how it behaves. A final thought to address is that the D/S/A algorithm with polynomial basis functions keeps higher-order polynomial terms without necessarily the corresponding lower-order terms. This is not customary. Many methods, such as multivariate adaptive polynomial spline regression (available in `R` as `polymars{polspline}`) (Stone et al., 1997; Kooperberg et al., 1997) which is similar to MARS, require that a univariate basis function is in the model before a tensor-product basis function involving the univariate basis function can be in the model and that this hierarchy is maintained during stepwise deletion. Furthermore, it is common practice in fields such as epidemiology to select interactions only after selecting main terms. If we have *a priori* knowledge that a higher order implies a lower order, then we would have to augment the D/S/A algorithm so that the set of moves maintains a particular hierarchy of terms. But for an example where x^3 is the truth we like that we do not require x and x^2 to be in the model as well. An option that will be included is the ability to specify an initial model that is to remain untouched by the deletion and substitution moves. For example, if we wanted a treatment effect in the model, then we can specify an initial model which contains the treatment covariate. This will provide the user with some control over what variables are in the final model.

The D/S/A algorithm is currently implemented in C with subroutines from the NAG libraries, and it will be made into an R function. An overall description of the estimation methodology with examples of the D/S/A algorithms used here and in the context of histogram regression is available in (Dudoit et al., 2003).

References

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.

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/`.

B. Durbin and S. Dudoit. A Deletion/Substitution/Addition algorithm for optimization of neural network architecture. Technical report, Division of Biostatistics, UC Berkeley, 2004. (In preparation).

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2), 2004.

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, S. Rosset, R. Tibshirani, and J. Shu. The entire regularization path for the support vector machine. URL `www-stat.stanford.edu/ hastie/Papers/NIPS04/nips.pdf`. Submitted to NIPS, March 2004.

DOI: 10.2202/1544-6115.1069

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.

C. Kooperberg, S. Bose, and C. J. Stone. Polychotomous regression. *Journal of The American Statistical Association*, 92:117–127, 1997.

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).

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`.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

S. E. Sinisi and M. J. van der Laan. Loss-based cross-validated Deletion/Substitution/Addition algorithms in estimation. Technical Report 143, Division of Biostatistics, University of California, Berkeley, March 2004. URL `www.bepress.com/ucbbiostat/paper143/`.

C. J. Stone, M. Hansen, C. Kooperberg, and Y. K. Truong. Polynomial splines and their tensor products in extended linear modeling. *The Annals of Statistics*, 25(4):1371–1470, 1997.

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/`.

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, S. Dudoit, and A. W. van der Vaart. The cross-validated adaptive epsilon-net estimator. Technical Report 142, Division of Biostatistics, University of California, Berkeley, February 2004. URL `www.bepress.com/ucbbiostat/paper142/`.

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

J. van Helden, B. André, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *Journal of Molecular Biology*, 281:827–842, 1998.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

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.

DOI: 10.2202/1544-6115.1069