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

Year 2005 *Paper* 170

Optimization of the Architecture of Neural Networks Using a Deletion/Substitution/Addition Algorithm

Blythe Durbin* Sandrine Dudoit†
Mark J. van der Laan‡

*Postdoctoral Fellow, Division of Biostatistics, School of Public Health, University of California, Berkeley, bpdurbin@stat.berkeley.edu

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

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

http://biostats.bepress.com/ucbbiostat/paper170

Optimization of the Architecture of Neural Networks Using a Deletion/Substitution/Addition Algorithm

Blythe Durbin, Sandrine Dudoit, and Mark J. van der Laan

Abstract

Neural networks are a popular machine learning tool, particularly in applications such as the prediction of protein secondary structure. However, overfitting poses an obstacle to their effective use for this and other problems. Due to the large number of parameters in a typical neural network, one may obtain a network fit that perfectly predicts the learning data yet fails to generalize to other data sets. One way of reducing the size of the parameter space is to alter the network topology so that some edges are removed; however, it is often not immediately apparent which edges should be eliminated. We propose a data-adaptive method of selecting an optimal network architecture using the Deletion/Substitution/Addition algorithm introduced in Sinisi and van der Laan (2004) and Molinaro and van der Laan (2004). Results of this approach in the regression case are presented on two simulated data sets and the diabetes data of Efron et al. (2002).

1 Introduction

1.1 Motivation

Artificial neural networks are a popular tool for the prediction of both polychotomous and continuous outcomes, that is, for classification and regression. Because the sigmoidal basis functions upon which they rely can closely approximate a wide variety of functions, neural networks have the capacity to provide good estimates of parameters with complex functional forms (Barron, 1993). In recent years, neural networks have been used with some success for the prediction of the secondary structure of a protein from its amino acid sequence (Jones, 1999).

However, neural networks are not without problems. If all possible edges are included in the network graph, the size of the parameter space may become unwieldy even for a network with a relatively small number of nodes. For example, one of the neural networks used for structure prediction in Jones (1999) has 315 input units, 75 hidden units, and 3 output units, for a total of 24,240 parameters.

Eliminating some edges in the neural network can reduce the size of the parameter space and improve prediction accuracy; however, it is not immediately apparent how one might go about this. One approach has been to select the network architecture by hand based on subject matter knowledge, as in Riis and Krogh (1996). However, this method relies on such expert knowledge being available, which may not be the case for all types of data. A better approach would be to select the neural network data-adaptively, that is, allowing the data themselves to select a network giving the best predictions.

We present a method for data-adaptive optimization of the architecture of a neural network in order to reduce the number of parameters in the model and improve prediction. This method uses a
Deletion/Substitution/Addition algorithm of the type introduced by Sinisi and van der Laan (2004) and Molinaro and van der Laan (2004). By deleting, substituting, and adding edges in the network graph, a sequence of networks is produced from which an optimal model may be selected by cross-validation. This method is demonstrated on two simulated data sets and the diabetes data from Efron et al. (2002). In this paper we illustrate only the application to single-output neural networks for regression, in which the response consists of a single, continuous variable; however, future work will focus on the extension of the neural network Deletion/Substitution/Addition algorithm to classification, where the neural network has multiple output nodes.

1.2 The Neural-Network Model

Let $\mathbf{X} = (\mathbf{W}, Y)$, where $\mathbf{W} \in \mathbb{R}^{N_{input}}$ is a covariate vector (possibly including an intercept) with N_{input} components and Y is a univariate continuous response variable. For simplicity, we first present the neural-network model for a fully-connected network, that is, one in which every input unit (covariate) is

connected to every hidden unit. In the case of a fully-connected network with N_{hidden} hidden units and univariate continuous output (response), the conditional expectation of the response Y given the covariates $\mathbf{W}$ is modeled as

$$\psi(\mathbf{W}) = E(Y|\mathbf{W}) = \beta_{10} + \sum_{j=1}^{N_{hidden}} \beta_{1j}\sigma(\boldsymbol{\beta}_{2j}^{\top}\mathbf{W}), \tag{1}$$

where $\beta_{1j} \in \mathbb{R}^1$, $\boldsymbol{\beta}_{2j} \in \mathbb{R}^{N_{input}}$, and $\sigma(\cdot)$ is a sigmoidal function[1]. Let $\boldsymbol{\beta} = [\beta_{10}, \beta_{11}, \ldots, \beta_{1N_{hidden}}, \boldsymbol{\beta}_{21}, \ldots, \boldsymbol{\beta}_{2N_{hidden}}]^{\top}$ denote the complete parameter vector, of dimension $O(N_{input} \cdot N_{hidden})$.

The function in Equation (1) corresponds to a fully-connected neural network, of the sort represented by the graph in Figure 1. However, as mentioned

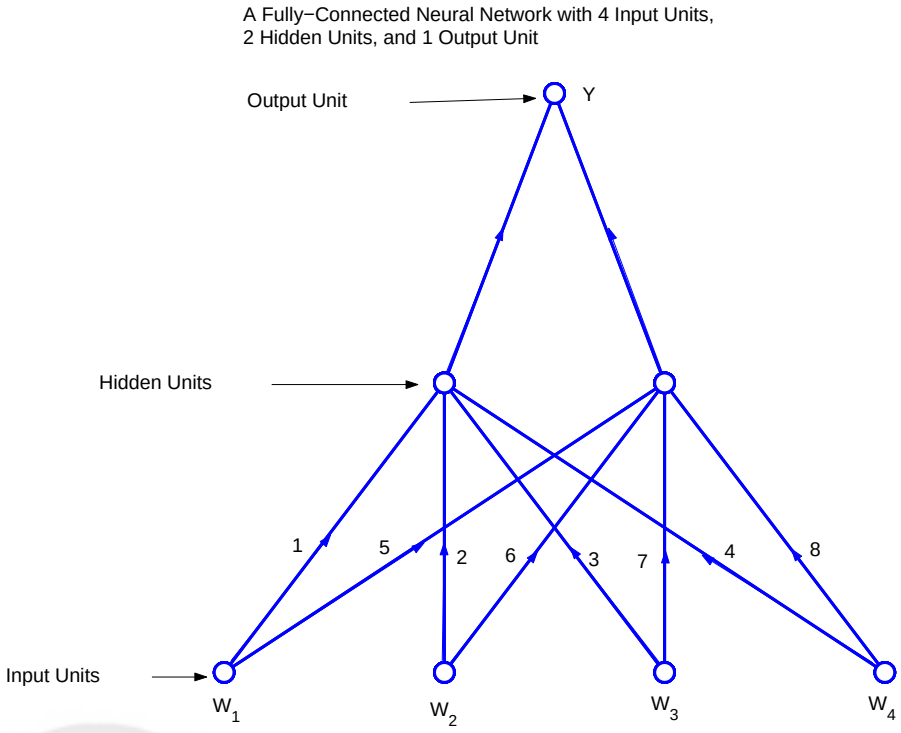

Figure 1: A fully-connected neural network, where every input unit is connected to every hidden unit.

previously, not every input unit need be connected to every hidden unit. Indeed, it is these sparser networks in which we will be most interested as we seek to

[1]A sigmoidal function $\sigma(\cdot)$ is "a bounded measurable function on the real line for which $\sigma(z) \to 1$ as $z \to \infty$ and $\sigma(z) \to 0$ as $z \to -\infty$" (Barron, 1993). A popular choice of sigmoidal function for neural networks is the logit function $\sigma(t) = 1/[1 + \exp(-t)]$.

reduce the size of the parameter space by eliminating edges in the network. We will use the following parameterization to describe these sparser networks.

With each hidden unit, associate a binary vector $\mathbf{p} \in \{0,1\}^{N_{input}}$ indicating which covariates feed into that hidden unit, with $p_l = 1$ if the l^{th} covariate W_l is present in the linear combination inside the sigmoidal function and $p_l = 0$ otherwise. Define an index set $I = \{\mathbf{p}_1, \ldots, \mathbf{p}_{N_{hidden}}\}$. Notice that this index set I completely describes the network architecture, assuming in the regression case that every hidden unit is connected to the single output unit. A null vector $\mathbf{p} = \mathbf{0}$ denotes a hidden unit with no covariates feeding into it.

Let

$$\phi_{\mathbf{p},\beta_{2\mathbf{p}}}(\mathbf{W}) = \sigma(\sum_{\{l:p_l=1\}} \beta_{2\mathbf{p}l} W_l). \tag{2}$$

The neural-network model may now be rewritten as

$$\psi_{I,\boldsymbol{\beta}}(\mathbf{W}) = \beta_{10} + \sum_{\mathbf{p}\in I} \beta_{1\mathbf{p}} \phi_{\mathbf{p},\beta_{2\mathbf{p}}}(\mathbf{W}). \tag{3}$$

Notice that $\psi_{I,\boldsymbol{\beta}}(\mathbf{W})$ is a linear combination of basis functions parameterized by $\mathbf{p}$ and $\boldsymbol{\beta}_{2\mathbf{p}}$. It is this parameterization with which we will work when applying the Deletion/Substitution/Addition algorithm to neural networks.

2 The Deletion/Substitution/Addition Algorithm for Selection of Network Architecture

2.1 Loss-Based Estimation with Cross-Validation

Deletion/Substitution/Addition (D/S/A) algorithms (Sinisi and van der Laan, 2004; Molinaro and van der Laan, 2004) provide an aggressive and flexible approach to selecting estimators from among possible linear combinations of basis functions such as that in Equation (3). In order to motivate this class of methods, a brief discussion of loss-based estimation with cross-validation appears to be in order at this point (as in van der Laan and Dudoit (2003)).

For a random variable $\mathbf{X}$ with data-generating distribution P, define the parameter of interest $\psi \in \boldsymbol{\Psi}$ as the minimizer of risk (or the expected value of the loss function) with respect to a given loss function $L(\cdot,\cdot)$ over a parameter space $\boldsymbol{\Psi}$:

$$\psi = \text{argmin}_{\psi' \in \boldsymbol{\Psi}} \int L(\mathbf{x}, \psi') dP(\mathbf{x}). \tag{4}$$

In the case of neural networks for regression, $\mathbf{X} = (\mathbf{W}, Y)$ and the parameter of interest is the conditional expected value of the response Y given the covariates $\mathbf{W}$,

$$\psi(\mathbf{W}) = E(Y|\mathbf{W}),$$

which minimizes the risk with respect to the squared-error loss function,

$$L(\mathbf{X}, \psi) = (Y - \psi(\mathbf{W}))^2.$$

Now, when presented with data $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$, with empirical distribution P_n, a naive approach might be to attempt to minimize the empirical risk $\int L(\mathbf{x}, \psi) dP_n(\mathbf{x})$; however, this would of course produce an overfit estimate. Sieve estimation, to which Deletion/Substitution/Addition algorithms provide one approach, confronts this problem by minimizing the empirical risk over subspaces of increasing size.

As in Section 1.2, define the parameter of interest in terms of a linear combination of basis functions, and let an index set I define which basis functions are used in a given linear combination. Then, collections $\mathcal{I}_k$ of index sets I, corresponding to neural networks of size k, may be used to define subspaces $\mathbf{\Psi}_k \subseteq \mathbf{\Psi}$ of the parameter space. The "size" of a neural network might refer to the number of parameters, the number of lower edges in the network graph, or the number of hidden units. The term "sieve" refers to a sequence of subspaces $\mathbf{\Psi}_k$, corresponding to collections $\mathcal{I}_k$ of index sets of size k, where, loosely speaking, the size of an index set I refers to a suitably defined size for the corresponding neural network.

For each subspace $\mathbf{\Psi}_k$ of the parameter space, one can generate a candidate estimator with minimum empirical risk in two steps:

1. For a fixed index set $I \in \mathcal{I}_k$, obtain $\psi_{I,\boldsymbol{\beta}_n,n}$ by finding $\boldsymbol{\beta}_n$ minimizing the empirical risk.

2. For each index set size k, find $I \in \mathcal{I}_k$ minimizing the empirical risk.

Finally, given a sequence of estimators, one for each subspace $\mathbf{\Psi}_k$, select among estimators corresponding to different values of k using cross-validation.

The D/S/A algorithm is concerned with Step 2 of the risk optimization problem and produces a sequence I_k, $k = 1, \ldots, K$ of index sets, one for each subspace.

In the implementation of the D/S/A algorithm for neural networks in Section 2.3, we define the size of the neural network as the number of lower edges in the network graph. Another implementation might define size to be the number of parameters in the network model; these two definitions are related but not identical. In either case, one only makes comparisons between models of the same size, based on whichever definition of size is given.

2.2 Deletion/Substitution/Addition Algorithms

Deletion/Substitution/Addition algorithms (Sinisi and van der Laan, 2004; Molinaro and van der Laan, 2004) attempt to solve the second aspect of the above problem: that of finding, for a given subspace $\mathbf{\Psi}_k$, an index set I of size k with minimum empirical risk. Sinisi and van der Laan (2004) apply a D/S/A algorithm to basis functions consisting of tensor products of polynomials, while Molinaro and van der Laan (2004) use indicator basis functions of the sort used in recursive partitioning.

For a given index set I of size k, a Deletion/Substitution/Addition algorithm is defined by three types of moves on I: deletion moves, substitution moves, and addition moves.

In a deletion move, an element is deleted from the index set I, producing an index set I^- of size $k-1$ corresponding to a smaller model:

$$\begin{aligned} I &\rightarrow I^- \\ k &\rightarrow k-1. \end{aligned}$$

In a substitution move, an element of the index set I is replaced with another element, producing a new index set $I^=$ of size k corresponding to a model of the same size:

$$\begin{aligned} I &\rightarrow I^= \\ k &\rightarrow k. \end{aligned}$$

Finally, in an addition move, an element is added to the index set I, producing an index set I^+ of size $k+1$:

$$\begin{aligned} I &\rightarrow I^+ \\ k &\rightarrow k+1. \end{aligned}$$

Let $f_E(I)$ denote the empirical risk associated with the index set I, with respect to a given loss function (e.g. the squared error loss function in the case of regression), and I_0 denote the current index set. Let I_k and $BEST(k)$ denote, respectively, the *current* best index set of size k and corresponding value of $f_E(I)$; that is, I_k is the current estimate of $\text{argmin}_{I \in \mathcal{I}_k} f_E(I)$, where $\mathcal{I}_k$ is the collection of all possible index sets of size k. Let $DEL(I)$ be the set of indices I^- that may be obtained from the index set I via a single deletion move, $SUB(I)$ be the set of indices $I^=$ that may be obtained from the index set I via a single substitution move, and $ADD(I)$ be the set of indices I^+ that may be obtained from the index set I via a single addition move. (Note that for a given index set I, the sets $DEL(I)$ or $SUB(I)$ may be empty, for example if $I = \emptyset$.)

The D/S/A algorithm proceeds as follows:

1. Initialization: Set $I_0 = \emptyset$, $BEST(k) = \infty$, $k = 1, 2, \ldots$
2. Algorithm: ($\star$) Let $k = |I_0|$.
   - Find best deletion move. Let $I^- \equiv \text{argmin}_{I \in DEL(I_0)} f_E(I)$. If $f_E(I^-) < BEST(k-1)$, set $I_0 = I^-$, $BEST(k-1) = f_E(I^-)$, $I_{k-1} = I^-$, go to ($\star$).
   - Else find best substitution move. Let $I^= \equiv \text{argmin}_{I \in SUB(I_0)} f_E(I)$. If $f_E(I^=) < f_E(I_0)$, set $I_0 = I^=$, go to ($\star$). If additionally, $f_E(I^=) < BEST(k)$, set $BEST(k) = f_E(I^=)$, $I_k = I^=$.
   - Else find best addition move. Let $I^+ \equiv \text{argmin}_{I \in ADD(I_0)} f_E(I)$. Set $I_0 = I^+$. If $f_E(I^+) < BEST(k+1)$, set $BEST(k+1) = f_E(I^+)$, $I_{k+1} = I^+$.

3. Stopping rule: Run algorithm until index sets exceed a predetermined size (other criteria are possible–see Sinisi and van der Laan (2004)).

Select among final index sets I_k, $k = 1, 2, \ldots$ using cross-validation.

By always attempting to update smaller models before increasing the model size, D/S/A algorithms allow an aggressive search of each subspace $\mathbf{\Psi}_k$ for the best model.

The D/S/A algorithm, as described above, applies to general estimation problems with arbitrary loss functions and parameterizations. However, the deletion, substitution, and addition moves are specific to the parameterization of the problem.

2.3 Deletion/Substitution/Addition Moves for Neural Networks

We will use a Deletion/Substitution/Addition algorithm to data-adaptively optimize the architecture of a neural network by deleting, substituting, and adding *edges* in the neural-network graph. The number of parameters in a neural-network model for regression is determined both by the number of edges in the lower part of the network graph and the number of hidden units. The number and placement of the edges in the lower part of the network graph, determining which covariates feed into which hidden units, will be the main focus of the algorithm. This leaves the number of hidden units as a tuning parameter, to be selected via cross-validation.

First, let us define the *edge set* $E \subseteq \{1, \ldots, N_{input} \cdot N_{hidden}\}$, derived from the previously defined index set $I = \{\mathbf{p}_1, \ldots, \mathbf{p}_{N_{hidden}}\}$ as follows: Let

$$\begin{aligned} \mathbf{e} &= \mathbf{e}(I) = [\mathbf{p}_1^\top, \ldots, \mathbf{p}_{N_{hidden}}^\top]^\top \\ E &= E(\mathbf{e}(I)) = E(I) = \{i \in \{1, \ldots, N_{input} \cdot N_{hidden}\} : e_i = 1\}. \end{aligned} \quad (5)$$

Loosely speaking, the edge set E is defined by numbering from 1 to $N_{input} \cdot N_{hidden}$ all of the possible lower edges that would be present in the fully-connected network and listing which ones are actually present.

The D/S/A moves described below will act on the edge set E rather than on the index set I. This means that the size k mentioned in Section 2.2 refers to the number of lower edges in the network graph; we have $|E| = k$ and $|I| = N_{hidden}$.

The moves of the D/S/A algorithm for neural networks are defined as follows:

Starting Value Initially, let $\mathbf{p}_j = \mathbf{0}$, $j = 1, \ldots, N_{hidden}$, and $I_0 = \{\mathbf{p}_1, \ldots, \mathbf{p}_{N_{hidden}}\}$. The number of allowed hidden units N_{hidden} is fixed in advance.

Deletion Move In a deletion move, an edge is deleted from the lower part of the network graph, decreasing the size of the edge set E by 1. Formally speaking, for a hidden unit j, this is equivalent to subtracting a unit vector $\mathbf{u}_l$

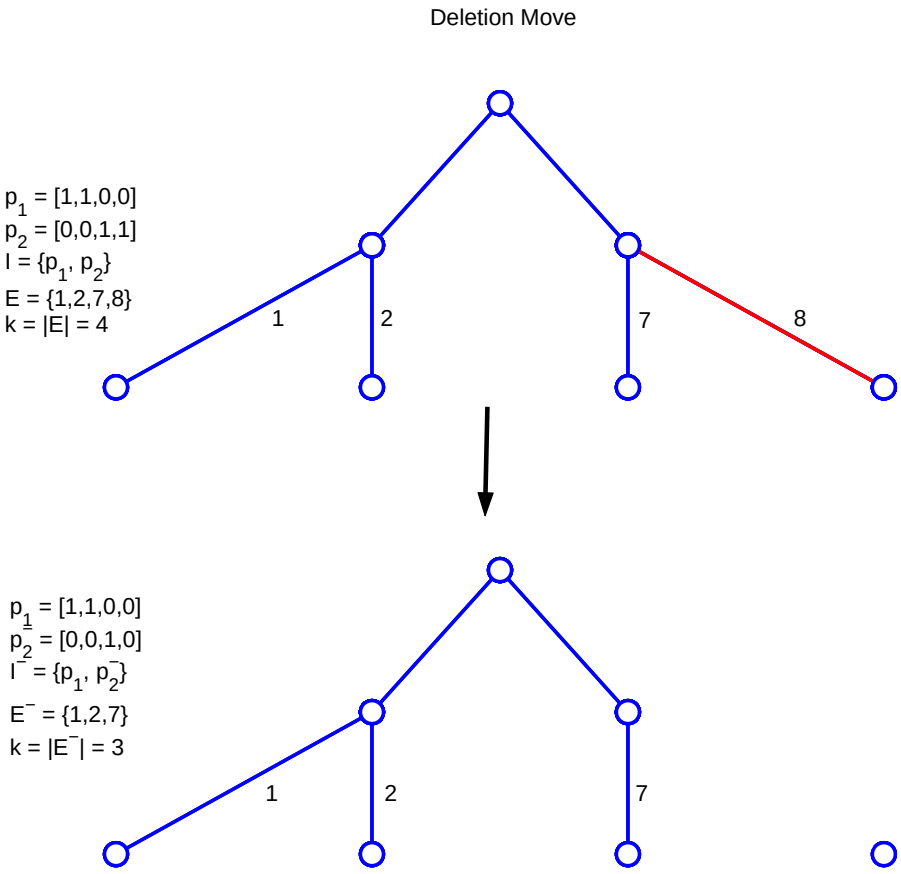

Figure 2: A deletion move.

(where $u_{lm} = 1$, $l = m$, $u_{lm} = 0$, $l \neq m$) from $\mathbf{p}_j$.

$$\begin{aligned} \mathbf{p}_j &\rightarrow \mathbf{p}_j^- = \mathbf{p}_j - \mathbf{u}_l,\ l \text{ s.t. } p_{jl} = 1 \\ I &\rightarrow I^- = \{\mathbf{p}_1, \ldots, \mathbf{p}_j^-, \ldots, \mathbf{p}_{N_{hidden}}\} \\ E &\rightarrow E^- = E(I^-), \\ &\quad |E^-| = |E| - 1 \end{aligned}$$

where the mapping from I^- to E^- is as defined in Equation (5). A deletion move is shown in Figure 2.

Note, that when a deletion move of the type described above is performed, the number of identifiable parameters in the network model may be reduced by as many as three if the last edge feeding into a hidden unit is deleted in the process.

Substitution move In a substitution move, an edge is removed from one hidden unit and a different edge is added either to that hidden unit or to a different existing hidden unit (meaning a hidden unit j for which $\mathbf{p}_j \neq \mathbf{0}$), producing a graph with the same number of lower edges. Formally speaking, for a hidden unit j, this is equivalent to subtracting a unit vector $\mathbf{u}_l$ from $\mathbf{p}_j$ and adding a unit vector $\mathbf{u}_{l'}$ to $\mathbf{p}_j$ or $\mathbf{p}_{j'}$.

If an edge is substituted within the same hidden unit,

$$\begin{aligned} \mathbf{p} &\rightarrow \mathbf{p}_j^{=} = \mathbf{p}_j - \mathbf{u}_l + \mathbf{u}_{l'},\ l \text{ s.t. } p_{jl} = 1, l' \text{ s.t. } p_{jl'} = 0 \\ I &\rightarrow I^{=} = \{\mathbf{p}_1, \ldots, \mathbf{p}_j^{=}, \ldots, \mathbf{p}_{N_{hidden}}\} \\ E &\rightarrow E^{=} = E(I^{=}), \\ &\quad\ |E^{=}| = |E|. \end{aligned}$$

If an edge is substituted between two different hidden units,

$$\begin{aligned} \mathbf{p}_j &\rightarrow \mathbf{p}_j^{=} = \mathbf{p}_j - \mathbf{u}_l,\ l \text{ s.t. } p_{jl} = 1 \\ \mathbf{p}_{j'} &\rightarrow \mathbf{p}_{j'}^{=} = \mathbf{p}_{j'} + \mathbf{u}_{l'},\ l' \text{ s.t. } p_{j'l'} = 0 \\ I &\rightarrow I^{=} = \{\mathbf{p}_1, \ldots, \mathbf{p}_j^{=}, \ldots, \mathbf{p}_{j'}^{=}, \ldots, \mathbf{p}_{N_{hidden}}\} \\ E &\rightarrow E^{=} = E(I^{=}), \\ &\quad\ |E^{=}| = |E|. \end{aligned}$$

A substitution move is shown in Figure 3.

When a substitution move of the type described above is performed, the size of the edge set E remains constant, but the number of parameters in the model may be reduced by up to two if the last edge is removed from a hidden unit in the process.

Addition Move In an addition move, either an edge is added to an existing hidden unit or a new hidden unit with a single covariate feeding into it is added to the neural-network model, increasing the number of lower edges in the model by 1. Formally, this is equivalent to adding a unit vector $\mathbf{u}_l$ to some $\mathbf{p}_j \in I$.

We have

$$\begin{aligned} \mathbf{p}_j &\rightarrow \mathbf{p}_j^{+} = \mathbf{p}_j + \mathbf{u}_l, l \text{ s.t. } p_{jl} = 0 \\ I &\rightarrow I^{+} = \{\mathbf{p_1}, \ldots, \mathbf{p}_j^{+}, \ldots, \mathbf{p}_{N_{hidden}}\} \\ E &\rightarrow E^{+} = E(I^{+}), \\ &\quad\ |E^{+}| = |E| + 1. \end{aligned}$$

An addition move is shown in Figure 4.

When an addition move is performed, the number of identifiable parameters in the model may increase by as many as three if the first edge is added to a previously unused hidden unit.

These deletion, substitution, and addition moves are used in the D/S/A algorithm outlined in Section 2.2 to produce a sequence of edge sets E_k corresponding to neural networks with k lower edges, each of which is an approximation to $\text{argmin}_{I:|E(I)|=k} f_E(I)$.

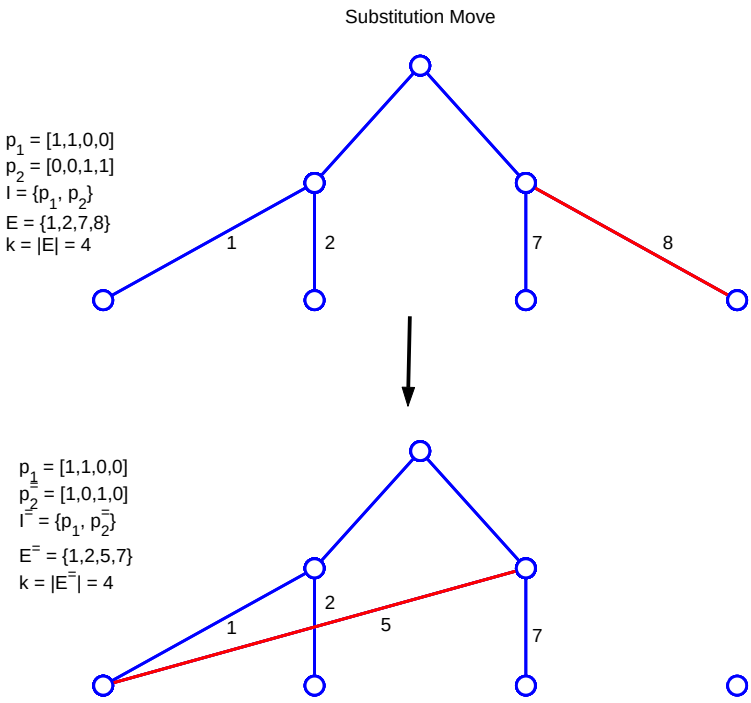

Figure 3: A substitution move.

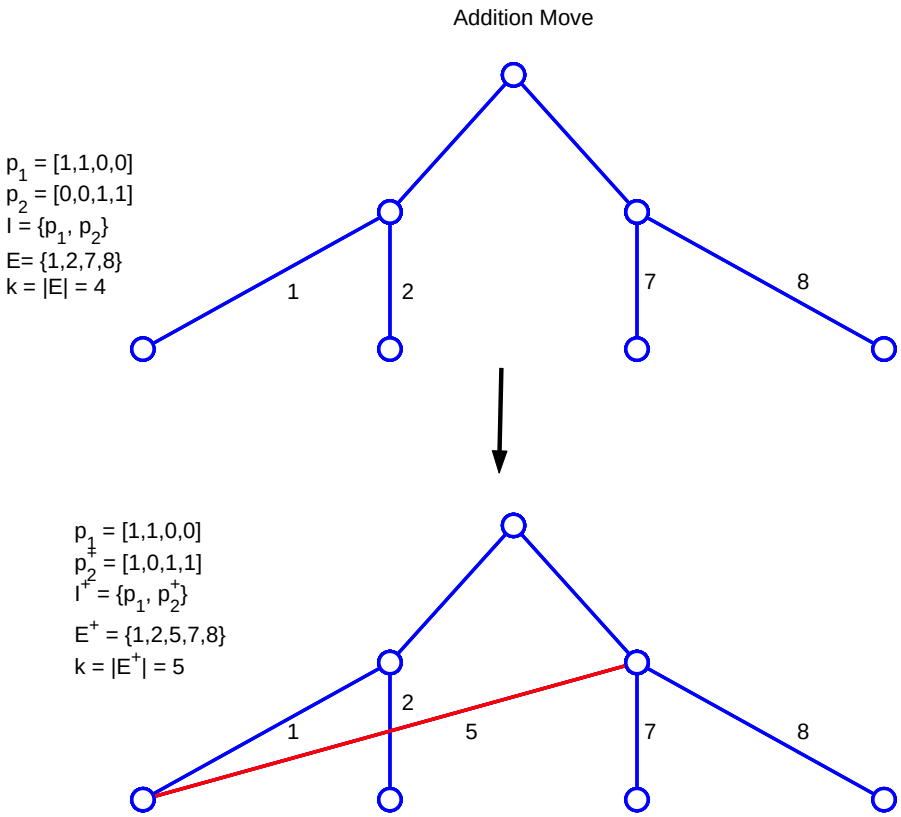

Figure 4: An addition move.

Note that in the D/S/A algorithm for neural networks outlined above, the index k (from the basic D/S/A algorithm in Section 2.2) refers to the number of lower edges in the network graph; that is, we use $k = |E|$ where $E = E(I)$ is the set of lower edges. An alternate version of D/S/A for neural networks might add and delete hidden units rather than edges. In this alternate version, we would then have $k = |I|$, where I is the index set defining which covariates feed into each hidden unit. We have chosen to use $k = |E|$ because the number of lower edges in the network graph more closely reflects the number of parameters in the model than does the number of hidden units, although adding or deleting a single lower edge does sometimes result in more than one parameter being added to or deleted from the network model. In the version we have used, $|I|$ remains fixed at N_{hidden}.

2.4 Cross-Validation

The D/S/A algorithm produces a sequence of K candidate estimators $\psi_{1,n}, \ldots, \psi_{K,n}$ of the parameter ψ. We use V-fold cross-validation to select a single estimator ψ_n from this sequence. In V-fold cross-validation, the learning sample $\mathcal{L}_n = \{X_1, \ldots, X_n\}$, $X_i = (\mathbf{W}_i, Y_i) \sim P$, is partitioned into V mutually exclusive and exhaustive sets. Each of these sets is used in turn as the validation sample, while the remaining $(V-1)$ parts are combined to make up the training sample. As in Dudoit and van der Laan (2003), let $\mathbf{B}_n$ be a binary random vector of length n, where $B_n(i) = 0$ if the i^{th} observation is in the training set and $B_n(i) = 1$ if the i^{th} observation is in the validation set. In V-fold cross validation, $\mathbf{B}_n$ comes from a distribution placing mass $1/V$ on each of V binary vectors $\mathbf{b}_n^v$ of length n, such that $\sum_i b_n^v(i) \approx \frac{n}{V} \, \forall \, v$ and $\sum_v b_n^v(i) = 1 \, \forall \, i$. Let $P^0_{n,\mathbf{B}_n}$ denote the empirical distribution of the training sample and let $P^1_{n,\mathbf{B}_n}$ denote the empirical distribution of the validation sample.

The cross-validated risk estimator for $\psi_{k,n}$, $\hat{\theta}_{p_n,n}(k)$, is defined as

$$\hat{\theta}_{p_n,n}(k) \equiv E_{\mathbf{B}_n} \int L(\mathbf{x}, \hat{\mathbf{\Psi}}_k(P^0_{n,\mathbf{B}_n})) dP^1_{n,\mathbf{B}_n}(\mathbf{x}), \tag{6}$$

where $\hat{\mathbf{\Psi}}_k(P^0_{n,\mathbf{B}_n})$ denotes the estimator obtained using only the training data and p_n is the proportion of observations in the validation sample. Essentially, the cross-validated risk estimator is obtained by evaluating the predictive power on the validation data of an estimate $\hat{\mathbf{\Psi}}_k(P^0_{n,\mathbf{B}_n})$ trained on the training data, then averaging over all possible splits of the learning sample into training and validation sets.

The cross-validated selector $\hat{k}_{p_n,n}$ is the value of k with lowest cross-validated risk, or

$$\hat{k}_{p_n,n} \equiv \text{argmin}_{k \in \{1,\ldots,K\}} \hat{\theta}_{p_n,n}(k). \tag{7}$$

Dudoit and van der Laan (2003) establish finite sample and asymptotic optimality results for the cross-validation selector, for general data generating distributions, loss functions, estimators, and cross-validation procedures. The asymptotic optimality results state that the cross-validation selector performs

(in terms of risk) asymptotically as well as an optimal benchmark or oracle selector based on the true unknown data generating distribution. For this reason, we use cross-validation to select among elements of the sequences produced by the D/S/A algorithm and to select tuning parameters, such as the number of hidden units.

3 Simulation Studies

3.1 Implementation

Two simulation studies were conducted to evaluate the performance of the cross-validated D/S/A algorithm for selection of neural-network architecture and to compare it to other predictors. This proceeded as follows: First, a learning set was simulated according to the specified simulation model. For 5-fold cross-validation, the learning set was then divided into 5 parts of equal size, each of which served in turn as the validation data set, with the remaining 4/5 of the learning set making up the training data. For each split of the learning set into training and validation data, the maximum number of hidden units allowed was fixed and the D/S/A algorithm was run on the training data. This produced, for each training/validation split and each maximum number of hidden units, a sequence of networks with increasing numbers of lower edges, ranging from a network with 1 lower edge to a fully-connected network. The empirical risk associated with each of these networks was then evaluated on the validation data. Averaging over each of the possible splits into training and validation data and repeating this procedure for different maximum allowed numbers of hidden units, we obtained the 5-fold cross-validated risk for each maximum number of hidden units and number of lower edges. The hidden unit-edge combination with the lowest cross-validated risk was selected as the optimal network size.

Note that this cross-validation procedure had not yet yielded a specific network architecture, as each training/validation split might well have produced a different sequence of networks. In order to obtain our final choice of network and estimates of the parameters $\boldsymbol{\beta}$, the D/S/A algorithm was run again on the full learning set, with the maximum number of hidden units fixed at the previously-selected optimum. The network architecture in this final sequence with the previously-selected number of edges was chosen as the final, optimal network. The performance of this final model was then evaluated on independently-simulated test sets from the same simulation model. In order to obtain 95% confidence intervals for the test-set risk, 100 test sets were simulated and the average and standard deviation of these test-set risks computed.

For analysis of the data sets described below, a variant of the D/S/A algorithm presented in Section 2.2 was used in which a substitution move was accepted only if it improved on the best network of size k (i.e. if $f_E(I^=) < BEST(k)$), rather than if it improved on the current network, (i.e. if $f_E(I^=) < f_E(I)$). This has the effect of speeding up the algorithm slightly.

The neural-network D/S/A method was compared with several other pre-

dictors on the same simulated learning and test sets. These predictors were:

- Least angle regression (LARS) (fit using the `lars` package in R, Version 0.9-5, by T. Hastie and B. Efron, see also Efron et al. (2002)), using both the C_p criterion and 5-fold cross-validation to select the LARS model;
- A multiple linear regression model including all of the predictor variables but no interactions (fit using the `lm()` function in R, Version 1.9.1);
- Recursive partitioning (fit using the `rpart()` function in R, from the `rpart` package, Version 3.1-2.1, by T. M. Therneau and B. Atkinson, see also Breiman et al. (1984));
- Multivariate adaptive regression splines (fit using the `polymars()` function in R, from the `polspline` package, Version 1.0.8, by C. Kooperberg, see also Friedman (1991));
- A full neural network (fit using the R function `nnet()`, Version 7.2-11, by B. D. Ripley.).

All of these computational methods were applied using default settings except where otherwise indicated. Code in R implementing the neural network-D/S/A algorithm used the `nnet()` function to fit each neural network model. 95% confidence intervals for test-set risks for each of these predictors were obtained using the same 100 simulated data sets described above.

It should be noted that the computation time for our prototype cross-validated D/S/A algorithm for neural networks exceeds that of the other predictors described above. Various options are being considered to increase computational efficiency.

3.2 Friedman 1 Simulation

The first simulation study used the Friedman 1 simulation (Friedman, 1991), implemented in the `mlbench` package in R, Version 1.0-0. This model consists of 10 independent covariates, $W_1, \ldots, W_{10}$, generated from a $U(0,1)$ distribution, and a univariate continuous outcome Y, where

$$Y = 10\sin(\pi W_1 W_2) + 20(W_3 - 0.5)^2 + 10W_4 + 5W_5 + \varepsilon, \tag{8}$$

and $\varepsilon \sim N(0,1)$. Notice that the last 5 covariates, $W_6, \ldots, W_{10}$, are not actually used to generate the response Y. The learning set and each of 100 independent test sets consisted, respectively, of 500 and 10,000 observations simulated from this model.

The cross-validated D/S/A procedure described above was applied to the learning set, examining models with up to 4 hidden units. Five-fold cross-validation selected 4 hidden units and 9 lower edges as the optimal network size. Notice that this model has only 18 unknown parameters (including intercept terms), compared to 49 ($= 4 \times (10+1)+4+1$) in a fully-connected network with

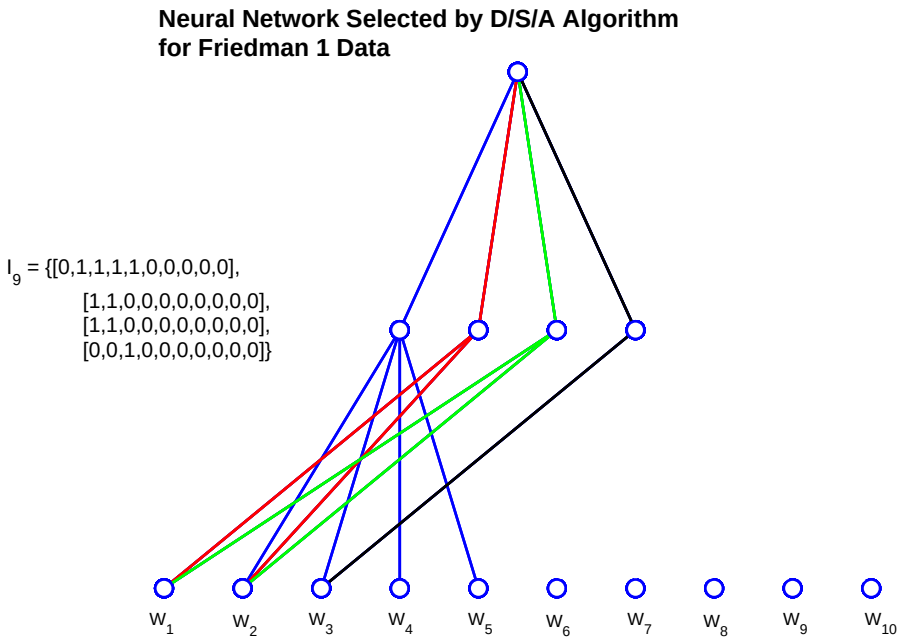

Figure 5: *Friedman 1 simulation*: Neural-network model selected by 5-fold cross-validated D/S/A algorithm.

4 hidden units. The D/S/A algorithm was run again on the full learning set in order to select the final network architecture, which is shown in Figure 5. Notice that this model does not include any of the "irrelevant" covariates, $W_6, \ldots, W_{10}$. Part of the final D/S/A sequence on the full learning set, consisting of models with 1 through 9 edges, is shown in Figures 14–22 in Appendix A.

Test-set risks for the neural-network D/S/A method and 6 other predictors are shown in Table 1, ranked in order of performance. D/S/A on neural networks was compared to a full neural network with 4 hidden units, to LARS, using both the C_p criterion and 5-fold cross-validation to select the LARS model, to recursive partitioning fit using `rpart()`, to multivariate adaptive regression splines fit using `polymars()`, and to a multiple linear regression model including all of the predictor variables but no interactions fit using `lm()`. The table shows the mean test-set risk and, in parentheses, 95% confidence intervals based on 100 simulated test sets (using the *same* simulated test sets for each predictor). The covariates selected by each predictor are shown in the fourth column, recalling that only covariates 1–5 were used to simulate the response. The "ratio" column shows the ratio of the average test-set risk for the predictor in question to that of the neural-network D/S/A method. Boxplots of the test-set risks are shown in Figures 6 and 7, listed in order of performance as in Table 1.

For this simulation model, the neural-network D/S/A had the lowest test-

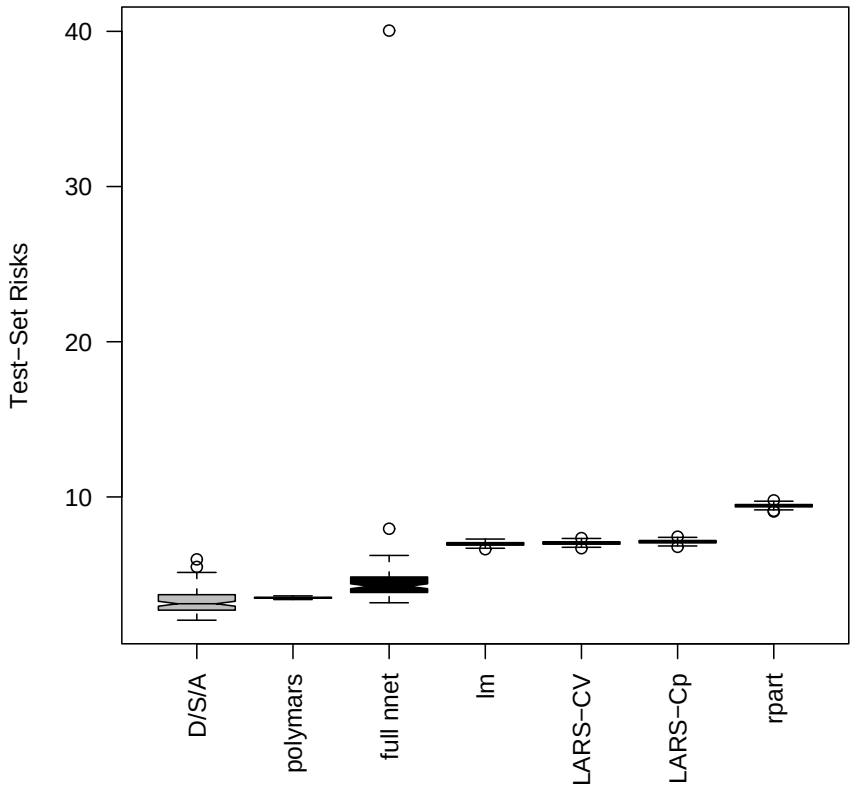

Figure 6: *Friedman 1 simulation*: Boxplots of test-set risks. If notches in boxplots do not overlap, there is evidence that the medians of two groups differ (Chambers et al., 1983).

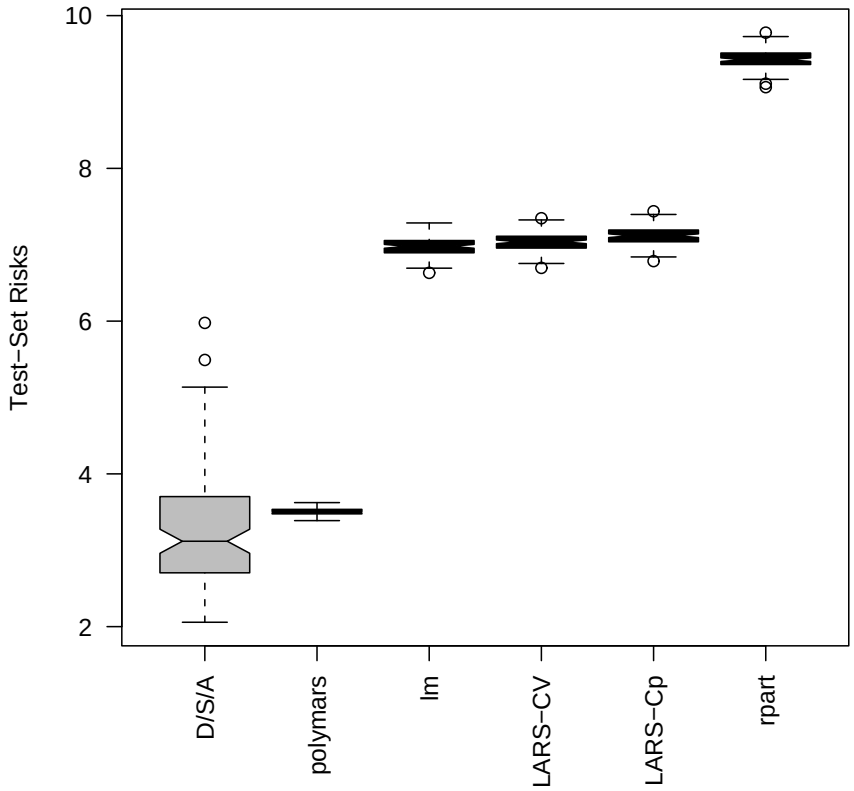

Figure 7: *Friedman 1 simulation*: Boxplots of test-set risks. The risks for the full neural network have been omitted for easier viewing. If notches in boxplots do not overlap, there is evidence that the medians of two groups differ (Chambers et al., 1983).

Predictor	Test-set risk	Ratio	Covariates
nnet-D/S/A	3.285 (1.732, 4.837)	1	1,2,3,4,5
polymars	3.504 (3.409, 3.598)	1.07	1,2,3,4,5
full nnet (4)	4.750 (0, 11.895)	1.45	(all)
lm	6.976 (6.745, 7.206)	2.12	(all)
LARS (CV)	7.033 (6.203, 7.864)	2.14	1,2,3,4,5,7
LARS (Cp)	7.117 (6.889, 7.345)	2.17	1,2,4,5
rpart	9.430 (9.186, 9.673)	2.87	1,2,3,4,5

Table 1: *Friedman 1 simulation*: Comparison of test-set risks. Table shows mean test-set risk and 95% confidence intervals from 100 independently simulated test sets, the ratio of the mean test-set risk for each predictor to that of the neural-network D/S/A algorithm, and covariates included in the model.

set risk, followed by multivariate adaptive regression splines, the full neural network, the linear regression model, LARS using cross-validation, LARS using the C_p criterion, and recursive partioning. Selection of a good model for these data requires both the ability to select the most informative covariates and to approximate the complicated functional form of the response. Neural-network D/S/A and multivariate adaptive regression splines appear to accomplish both of these tasks, resulting in a low test-set risk. The full neural network might approximate the functional form well, but the performance of this method is hampered by inclusion of irrelevant covariates. Recursive partitioning selects the correct covariates, but the piecewise-constant basis function cannot fit the complicated functional form of the response. The two methods, neural-network D/S/A and multivariate adaptive regression splines, that are able fit the complicated functional form and select covariates data-adaptively perform best in this situation.

As Figures 6 and 7 indicate, the test-set risks for the neural-network models appear to be much more variable than for the other predictors, particularly those for the full neural network. However, as will be seen in Sections 3.3 and 4, the Friedman 1 data are the only example data set on which this occurs, and the increased variability may be an anomaly specific to these data.

Figure 8 shows the correlation matrix for the test-set risks of the 7 different predictors on 100 simulated data sets. From this figure, the performance of the linear model and that of the two LARS models appear to be highly correlated (likely because the models themselves are very similar), with little correlation between any of the other predictors.

3.3 Polynomial Simulation

The second simulation study uses a simple polynomial model with second-order interaction terms. Three independent covariates, $W_1, \ldots, W_3$, were simulated

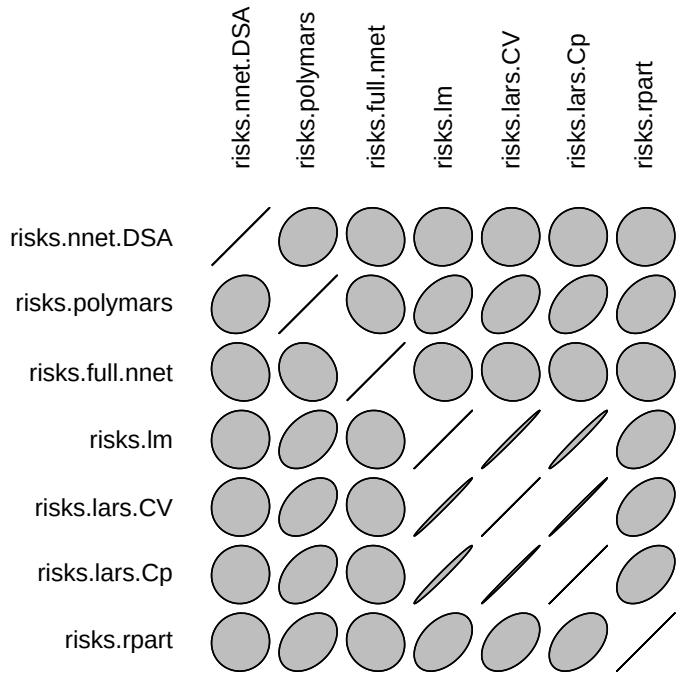

Figure 8: *Friedman 1 simulation*: Correlation matrix for test-set risks. Each ellipse represents a level curve of a bivariate normal density with the matching correlation.

Predictor	Test-set risk	Ratio	Covariates
Truth	1.008 (0.9790, 1.036)	0.995	1,2,3
nnet-D/S/A	1.012 (0.9822, 1.042)	1	1,2,3
polymars	1.019 (0.9898, 1.048)	1.01	1,2,3
LARS (CV), lm	1.025 (0.9955, 1.054)	1.01	1,2,3
rpart	1.273 (1.237, 1.310)	1.26	1,2,3
LARS (Cp)	1.814 (1.760, 1.868)	1.79	1,2

Table 2: *Polynomial simulation*: Comparison of test-set risks. Table shows mean test-set risk and 95% confidence intervals from 100 independently simulated test sets, the ratio of the mean test-set risk for each predictor to that of the neural-network D/S/A algorithm, and covariates included in the model.

from a $U(0,1)$ distribution, and the response Y was generated as

$$Y = W_1 + W_2 + W_3 + W_1 * W_2 + W_1 * W_3 + W_2 * W_3 + \varepsilon, \tag{9}$$

where $\varepsilon \sim N(0,1)$.

A learning set of size 500 was simulated from this model, as were 100 independent test sets of size 10,000. The neural-network D/S/A procedure with cross-validation was implemented as described in Section 3.1, examining models with up to 3 hidden units. This method selected a fully-connected neural network with 2 hidden units as the optimal network architecture for these data.

The neural-network D/S/A procedure was compared to the true model with parameters estimated using `lm()`, to multivariate adaptive regression splines fit using `polymars()`, to LARS, using both the C_p criterion and 5-fold cross-validation to select the LARS model, to a multiple linear regression model including all of the predictor variables but no interactions fit using `lm()`, and to recursive partitioning fit using `rpart()`. Table 2 shows, for each predictor, the mean test-set risks and 95% confidence intervals obtained from 100 independently simulated test sets of size 10,000, as well as the ratio of the mean test-set risk for each predictor to that of the neural-network D/S/A method. Boxplots of the test-set risks are shown in Figure 9.

As one would expect, the true model has the lowest test-set risk, followed by the neural-network D/S/A procedure, multivariate adaptive regression splines, LARS using cross-validation (which selected the multiple linear regression model including all of the covariates but no interactions), recursive partitioning, and LARS using the C_p criterion. The neural-network basis appears to be able to approximate the functional form well, and the D/S/A procedure selected a network architecture (a fully-connected network) that, like the true data-generating model, is symmetric in all of the covariates.

Figure 10 shows the correlation matrix for the test-set risks of the 6 different predictors on 100 simulated data sets (results are only shown for 6 predictors because CV-LARS selected the linear model in this case). From this figure, the performances of the true model, nnet-D/S/A, polymars, and the LARS model

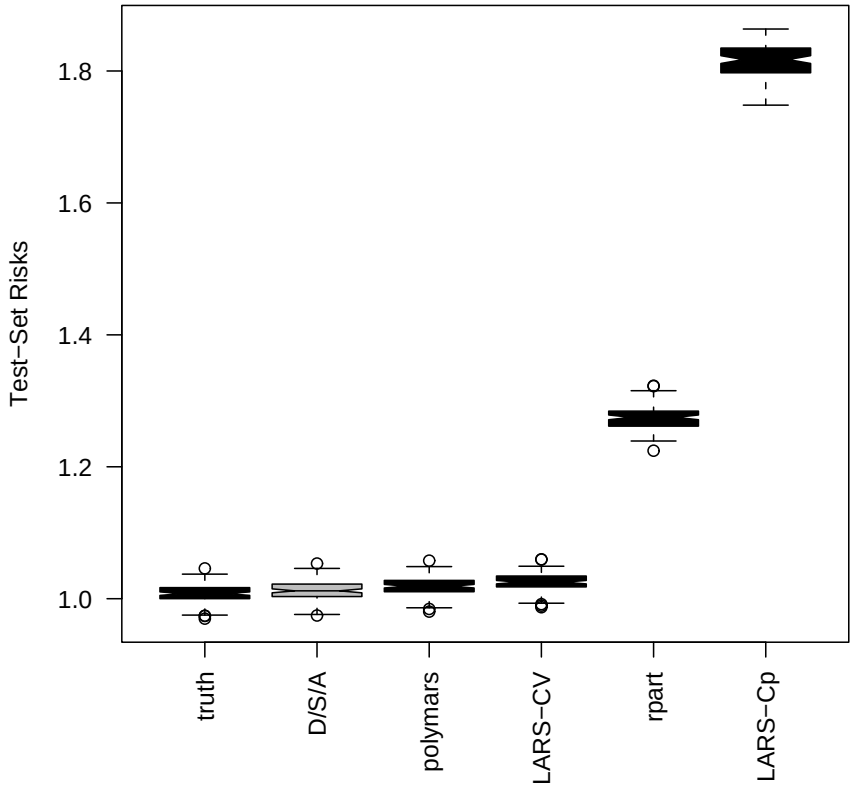

Figure 9: *Polynomial simulation*: Boxplots of test-set risks. If notches in boxplots do not overlap, there is evidence that the medians of two groups differ (Chambers et al., 1983). For these data, LARS using cross-validation (LARS-CV) selected the multiple regression model, so separate results for that model are not shown.

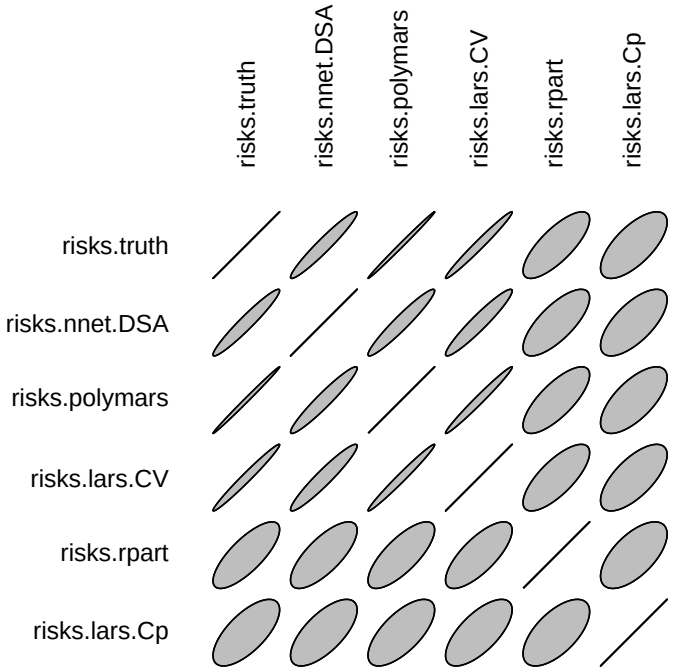

Figure 10: *Polynomial simulation*: Correlation matrix for test-set risks. Each ellipse represents a level curve of a bivariate normal density with the matching correlation.

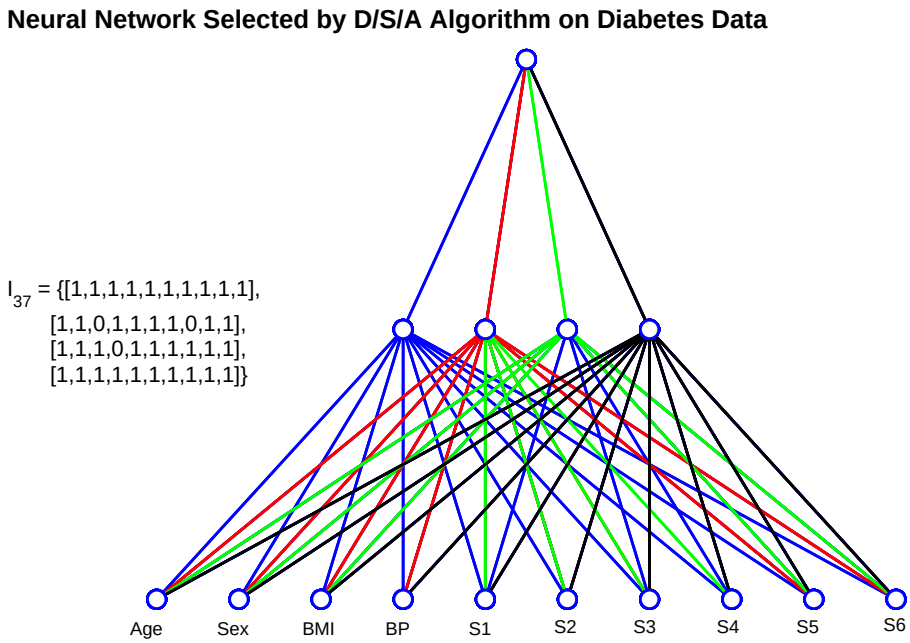

Figure 11: *Diabetes data*: Neural-network model selected by the 5-fold cross-validated D/S/A procedure.

selected by cross-validation appear to be highly correlated with one another, with a fair amount of correlation exisiting between the other predictors as well.

4 Diabetes Data Analysis

The methods above were next applied to the diabetes data from Efron et al. (2002). These data consist of a univariate continuous response measuring disease progression and 10 covariates: age, sex, blood pressure, body mass index, and 6 continuous blood serum measurements referred to as S1–S6.

We randomly divided the 442 observations into a learning set with 392 observations and a test set with 50 observations (thus allocating 11% of the data to the test set). The covariates were scaled to lie between 0 and 1, which is recommended by Ripley (1996) for use with neural networks and does not change the test-set risk of models fit using LARS, recursive partitioning, or linear regression.

The neural-network D/S/A procedure selected a model with 4 hidden units and 37 lower edges from models with up to 4 hidden units. Figure 11 shows this network, which is nearly indistinguishable from a fully-connected network.

The performance of the neural-network D/S/A procedure on the diabetes data was compared to that of a multiple linear regression model including all

Predictor	Test-set risk	Ratio	Covariates
lm	3182.0 (1966.2, 4397.7)	0.892	(all)
LARS (CV)	3270.9 (2118.6, 4423.2)	0.917	Sex, BMI, BP, S1–S3, S5, S6
polymars	3301.8 (2123.8, 4479.9)	0.926	Sex, BMI, BP, S3, S5, S6
LARS (Cp)	3336.7 (2206.2, 4467.3)	0.936	Sex, BMI, BP, S2, S3, S5, S6
full nnet (4)	3552.4 (2297.2, 4807.6)	0.996	(all)
nnet-D/S/A	3565.2 (2368.4, 4175.8)	1	(all)
rpart	3692.0 (2498.9, 4885.0)	1.04	BMI, BP, S2, S3, S5, S6

Table 3: *Diabetes data*: Comparison of test-set risks. Table shows the mean test-set risk and 95% confidence intervals over 100 bootstrap samples from the test set, the ratio of the mean test-set risk for each predictor to that of the neural-network D/S/A algorithm, and covariates included in the model.

of the predictor variables but no interactions fit using `lm()`, to LARS, using both the C_p criterion and 5-fold cross-validation to select the LARS model, to multivariate adaptive regression splines fit using `polymars()`, to recursive partitioning fit using `rpart()`, and to a full neural network with 4 hidden units fit using `nnet()`. 95% confidence intervals for the test-set risks were calculated from 100 bootstrap samples taken with replacement from the test set of size 50. Table 3 shows the mean test-set risk over the bootstrap samples, 95% confidence intervals for the test-set risks, and the covariates each predictor selected for inclusion in the model. Boxplots of the test-set risks are shown in Figure 12.

On these data, curiously, the multiple linear regression model has the lowest test-set risk and the neural-network D/S/A procedure does not perform particularly well. However, the good performance of a linear regression model suggests that the relationship between covariates and response might in fact be linear in nature. Thus, it is not particularly surprising that the sigmoidal basis functions used by neural networks might not suit data that appear to follow a simpler functional form. Neural networks appear to be, so to speak, too big of a hammer for this problem. However, as Figure 12 shows, the performance of all of these predictors is quite similar.

Figure 13 shows the correlation matrix for the 100 bootstrap test-set risks of the 7 different predictors. A fair amount of correlation appears to exist between the performances of most of the predictors.

5 Conclusions

The neural-network D/S/A algorithm performed well compared to other predictors on two simulated data sets. The use of neural networks, which allows approximation of complicated functional forms, and of data-adaptive selection of the network architecture, which allows selection of the most relevant covariates, combine to provide good prediction of new observations. The neural-network D/S/A method performed less well on the diabetes data of Efron et al. (2002),

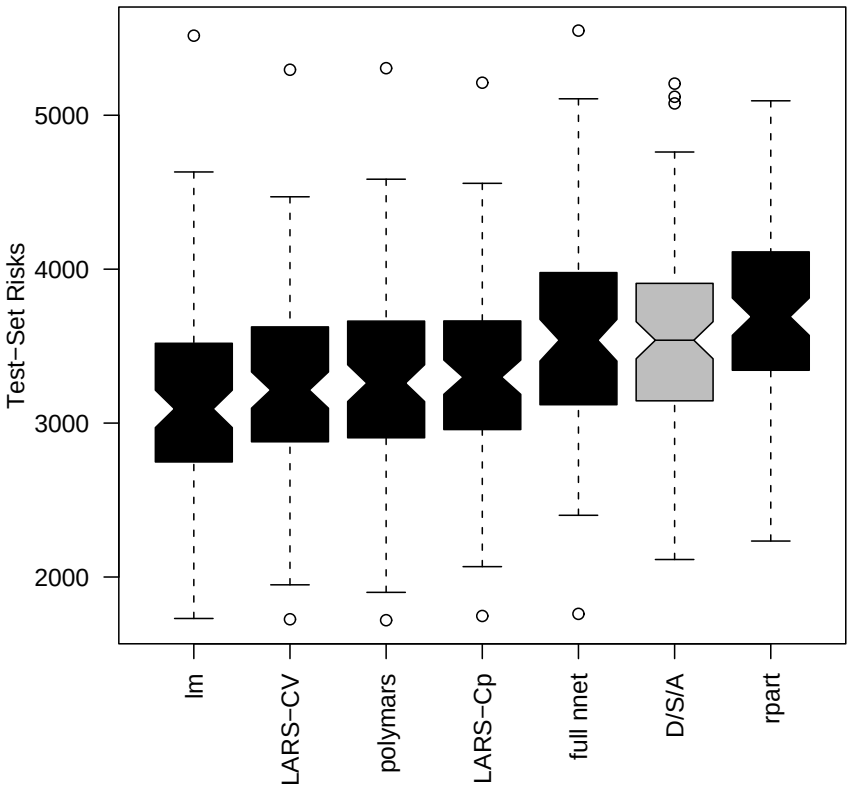

Figure 12: *Diabetes data*: Boxplots of bootstrap test-set risks. If notches in boxplots do not overlap, there is evidence that the medians of two groups differ (Chambers et al., 1983).

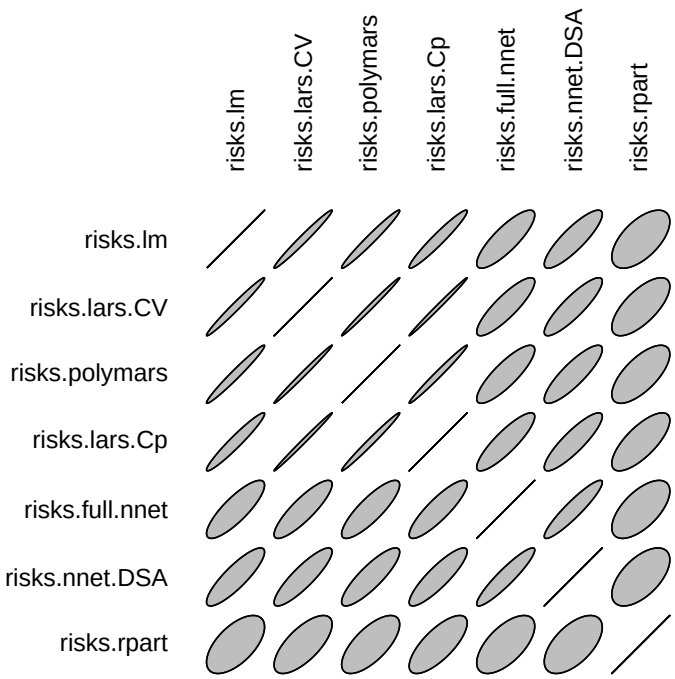

Figure 13: *Diabetes data*: Correlation matrix for test-set risks. Each ellipse represents a level curve of a bivariate normal density with the matching correlation.

but this is likely because the data have a rather simple functional form (to which neural networks might be less well-suited), as evidenced by the strong performance of a linear regression model.

Future work will center on the extension of the D/S/A approach to polychotomous outcomes, where one wishes to classify new observations into one of K classes. This differs from the regression case in that there are multiple output units in the network, and every hidden unit must no longer be connected to every output unit. In this setting, one may delete, substitute, and add edges in the upper part of the neural-network graph as well as the lower part, which increases the complexity of the problem. This D/S/A algorithm for classification neural networks will be applied to the prediction of the secondary structure of proteins from their amino acid sequences. Additionally, one may use neural networks to assess the importance of covariates in predicting an outcome, with measures such as the number and coefficients of edges feeding from a given covariate. Another approach is to derive loss-based variable importance measures from the neural-network model, as in Dudoit et al. (2003).

6 Acknowledgements

Funding for the work presented in this paper was provided in part by the National Science Foundation. Discussions with Kasper Hansen and Sandra Sinisi of the Division of Biostatistics at UC Berkeley and with Theresa Head-Gordon of the Department of Bioengineering at UC Berkeley and the Lawrence Berkeley National Laboratory are also gratefully acknowledged.

References

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

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

J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, Belmont, CA, 1983.

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, 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. In G. Piatetsky-Shapiro and P. Tamayo, editors, *Microarray Data Mining*, vol-

ume 5 of *SIGKDD Explorations*, pages 56–68. ACM, 2003. URL www.bepress.com/ucbbiostat/paper137,www.acm.org/sigs/sigkdd/explorations/issue5-2.htm.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. Technical Report 220, Department of Statistics, Stanford University, 2002.

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–67, 1991.

D. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292:195–202, 1999.

A. Molinaro and M. J. van der Laan. A Deletion/Substitution/Addition algorithm for partitioning the covariate space in prediction. Technical Report 162, Division of Biostatistics, University of California, Berkeley, 2004. URL www.bepress.com/ucbbiostat/paper162.

R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2004. URL http://www.R-project.org. ISBN 3-900051-00-3.

S. Riis and A. Krogh. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of Computational Biology*, 3:163–183, 1996.

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

S. E. Sinisi and M. J. van der Laan. Deletion/Substitution/Addition algorithms in learning with applications in genomics. *SAGMB*, 3(1), 2004.

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

A D/S/A Sequence on Friedman 1 Data

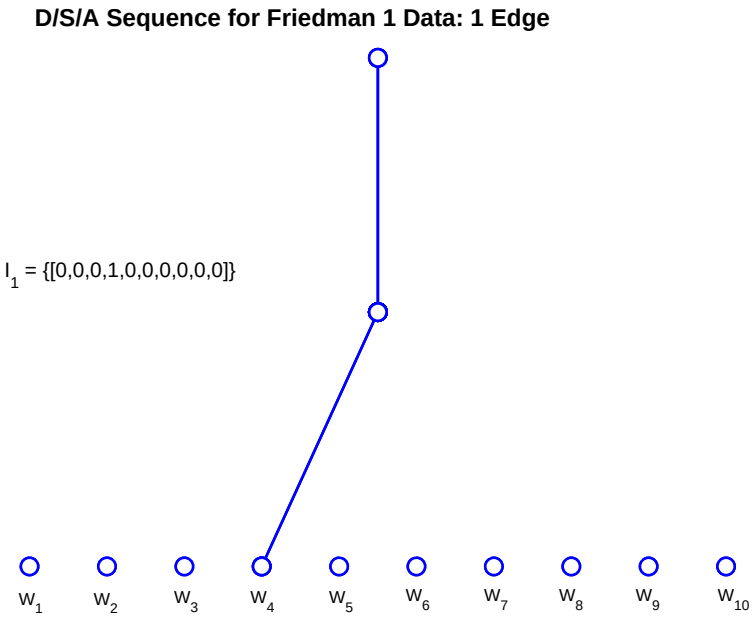

Figure 14: *Friedman 1 simulation*: D/S/A sequence–model with 1 lower edge.

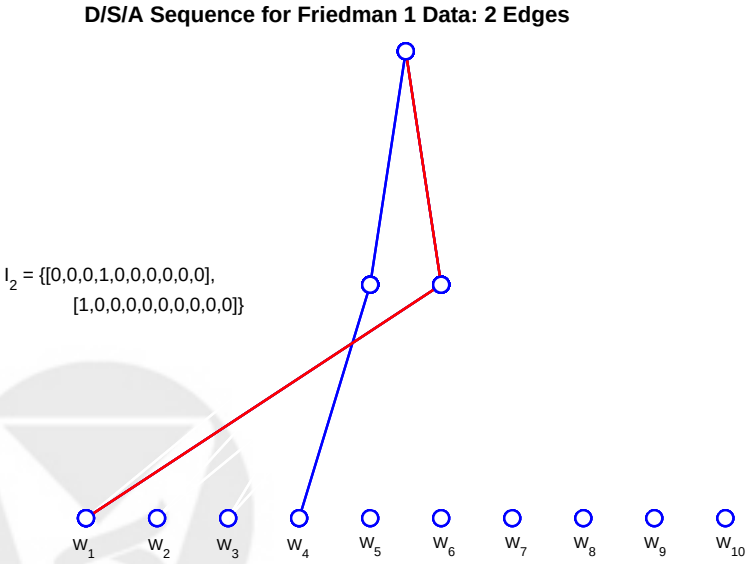

Figure 15: *Friedman 1 simulation*: D/S/A sequence–model with 2 lower edges.

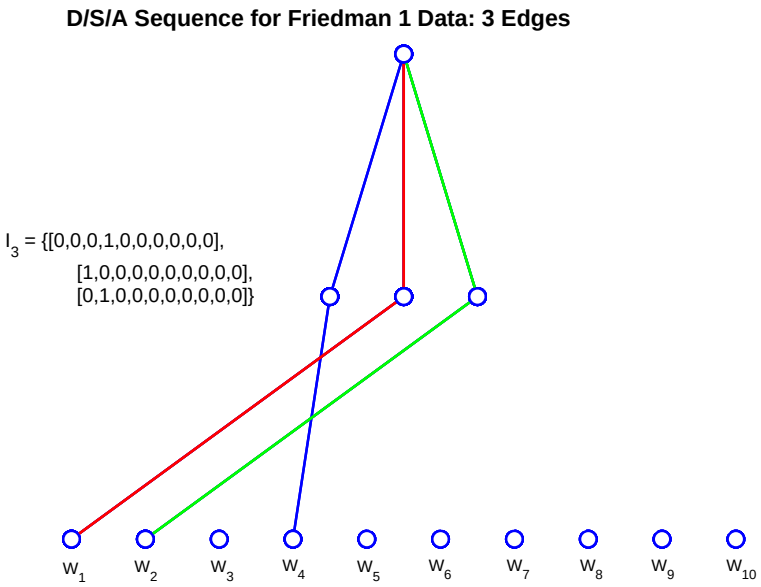

Figure 16: *Friedman 1 simulation*: D/S/A sequence–model with 3 lower edges.

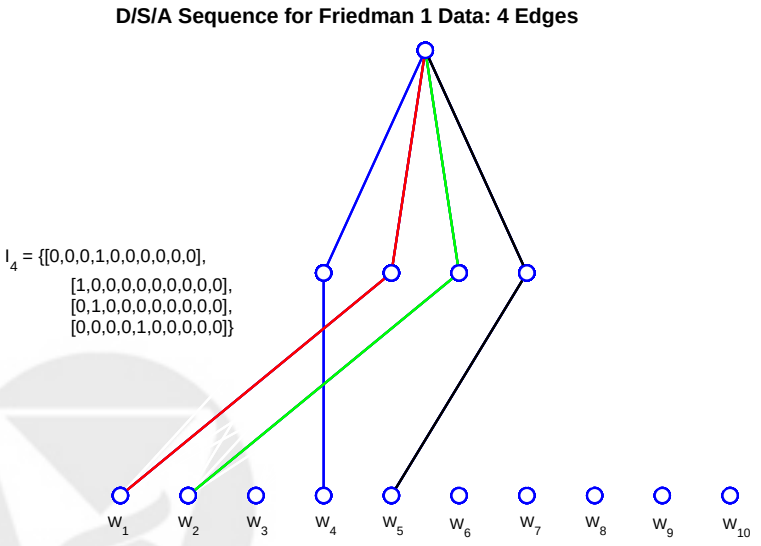

Figure 17: *Friedman 1 simulation*: D/S/A sequence–model with 4 lower edges.

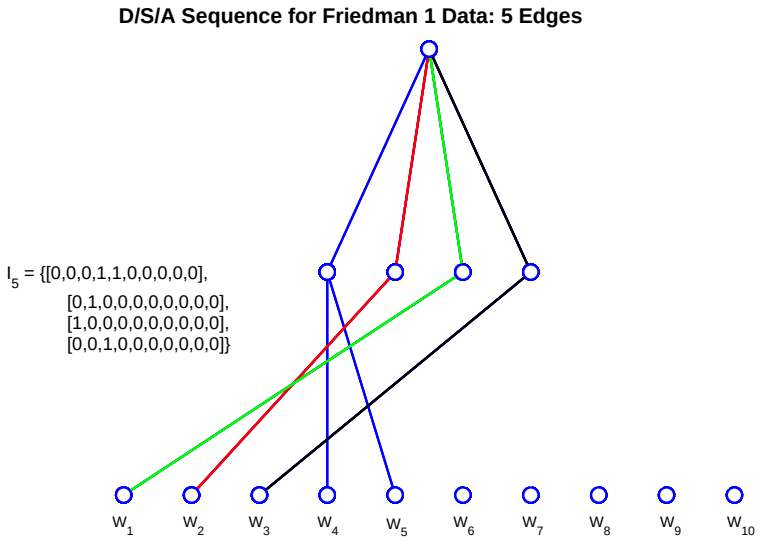

Figure 18: *Friedman 1 simulation*: D/S/A sequence–model with 5 lower edges.

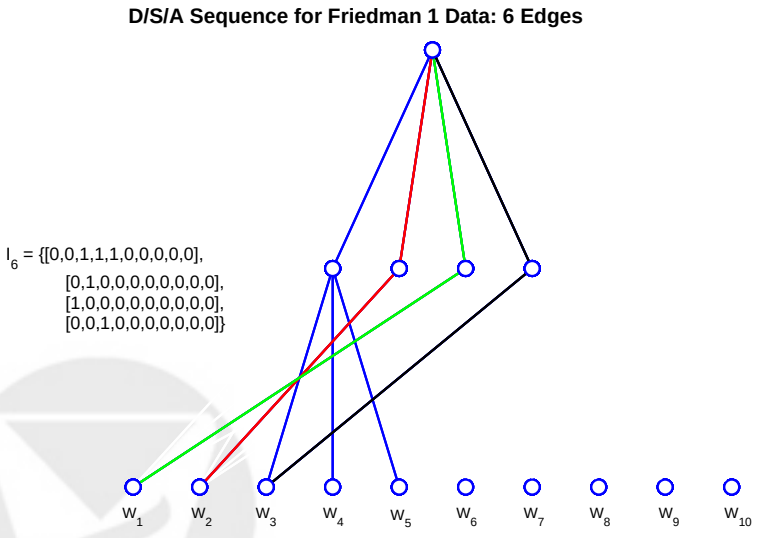

Figure 19: *Friedman 1 simulation*: D/S/A sequence–model with 6 lower edges.

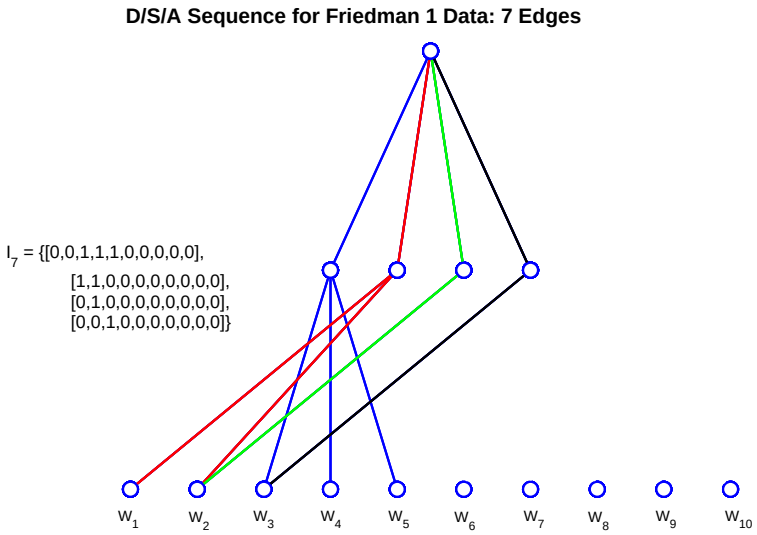

Figure 20: *Friedman 1 simulation*: D/S/A sequence–model with 7 lower edges.

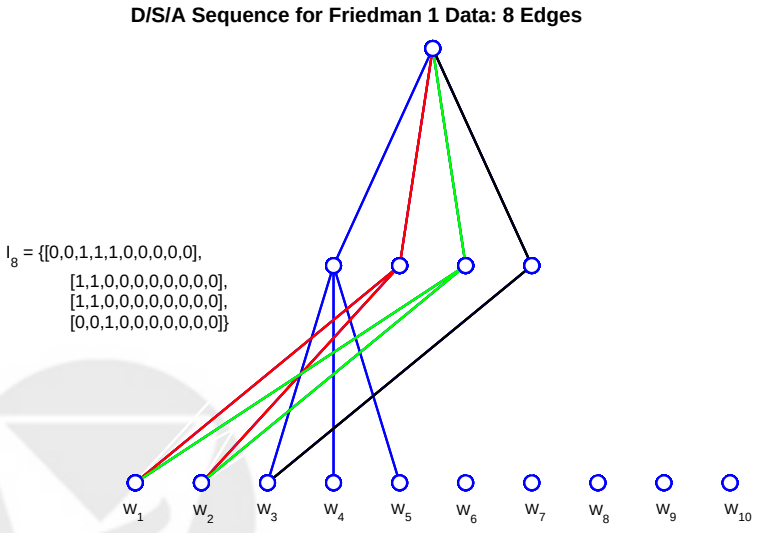

Figure 21: *Friedman 1 simulation*: D/S/A sequence–model with 8 lower edges.

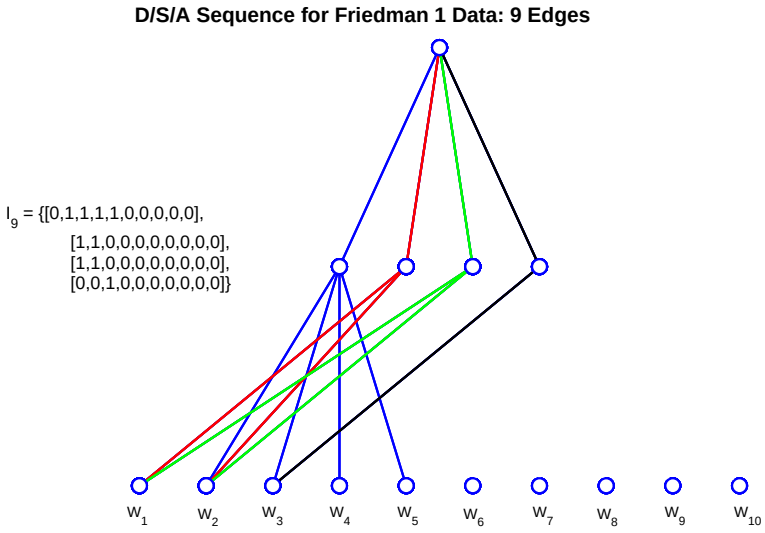

Figure 22: *Friedman 1 simulation*: D/S/A sequence–model with 9 lower edges, selected as best model by 5-fold cross-validation.