Journal of Machine Learning Research 7 (2006) 1001–1024 Submitted 08/05; Revised 3/06; Published 6/06

Sparse Boosting

Peter Bühlmann BUHLMANN@STAT.MATH.ETHZ.CH
Seminar für Statistik
ETH Zürich
Zürich, CH-8092, Switzerland

Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics
University of California
Berkeley, CA 94720-3860, USA

Editors: Yoram Singer and Larry Wasserman

Abstract

We propose Sparse Boosting (the SparseL_2Boost algorithm), a variant on boosting with the squared error loss. SparseL_2Boost yields sparser solutions than the previously proposed L_2Boosting by minimizing some penalized L_2-loss functions, the FPE model selection criteria, through small-step gradient descent. Although boosting may give already relatively sparse solutions, for example corresponding to the soft-thresholding estimator in orthogonal linear models, there is sometimes a desire for more sparseness to increase prediction accuracy and ability for better variable selection: such goals can be achieved with SparseL_2Boost.

We prove an equivalence of SparseL_2Boost to Breiman's nonnegative garrote estimator for orthogonal linear models and demonstrate the generic nature of SparseL_2Boost for nonparametric interaction modeling. For an automatic selection of the tuning parameter in SparseL_2Boost we propose to employ the gMDL model selection criterion which can also be used for early stopping of L_2Boosting. Consequently, we can select between SparseL_2Boost and L_2Boosting by comparing their gMDL scores.

Keywords: lasso, minimum description length (MDL), model selection, nonnegative garrote, regression

1. Introduction

Since its inception in a practical form in Freund and Schapire (1996), boosting has obtained and maintained its outstanding performance in numerous empirical studies both in the machine learning and statistics literatures. The gradient descent view of boosting as articulated in Breiman (1998, 1999), Friedman et al. (2000) and Rätsch et al. (2001) provides a springboard for the understanding of boosting to leap forward and at the same time serves as the base for new variants of boosting to be generated. In particular, the L_2Boosting (Friedman, 2001) takes the simple form of refitting a base learner to residuals of the previous iteration. It coincides with Tukey's (1977) twicing at its second iteration and reproduces matching pursuit of Mallat and Zhang (1993) when applied to a dictionary or collection of fixed basis functions. A somewhat different approach has been suggested by Rätsch et al. (2002). Bühlmann and Yu (2003) investigated L_2Boosting for linear base procedures (weak learners) and showed that in such cases, the variance or complexity of the boosted procedure is bounded and increases at an increment which is exponentially diminishing as iterations run – this special case calculation implies that the resistance to the over-fitting behavior of boosting could be due to the fact that the complexity of boosting increases at an extremely slow pace.

Recently Efron et al. (2004) made an intriguing connection for linear models between L_2Boosting and Lasso (Tibshirani, 1996) which is an ℓ^1-penalized least squares method. They consider a modification of L_2Boosting, called forward stagewise least squares (FSLR) and they show that for some special cases, FSLR with infinitesimally small step-sizes produces a set of solutions which coincides with the set of Lasso solutions when varying the regularization parameter in Lasso. Furthermore, Efron et al. (2004) proposed the least angle regression (LARS) algorithm whose variants give a clever computational short-cut for FSLR and Lasso.

For high-dimensional linear regression (or classification) problems with many ineffective predictor variables, the Lasso estimate can be very poor in terms of prediction accuracy and as a variable selection method, see Meinshausen (2005). There is a need for more sparse solutions than produced by the Lasso. Our new SparseL_2Boost algorithm achieves a higher degree of sparsity while still being computationally feasible, in contrast to all subset selection in linear regression whose computational complexity would generally be exponential in the number of predictor variables. For the special case of orthogonal linear models, we prove here an equivalence of SparseL_2Boost to Breiman's (1995) nonnegative garrote estimator. This demonstrates the increased sparsity of SparseL_2Boost over L_2Boosting which is equivalent to soft-thresholding (due to Efron et al. (2004) and Theorem 2 in this article).

Unlike Lasso or the nonnegative garrote estimator, which are restricted to a (generalized) linear model or basis expansion using a fixed dictionary, SparseL_2Boost enjoys much more generic applicability while still being computationally feasible in high-dimensional problems and yielding more sparse solutions than boosting or ℓ^1-regularized versions thereof (see Rätsch et al., 2002; Lugosi and Vayatis, 2004). In particular, we demonstrate its use in the context of nonparametric second-order interaction modeling with a base procedure (weak learner) using thin plate splines, improving upon Friedman's (1991) MARS.

Since our SparseL_2Boost is based on the final prediction error criterion, it opens up the possibility of bypassing the computationally intensive cross-validation by stopping early based on the model selection score. The gMDL model selection criterion (Hansen and Yu, 2001) uses a data-driven penalty to the L_2-loss and as a consequence bridges between the two well-known AIC and BIC criteria. We use it in the SparseL_2Boost algorithm and for early stopping of L_2Boosting. Furthermore, we can select between SparseL_2Boost and L_2Boosting by comparing their gMDL scores.

2. Boosting with the Squared Error Loss

We assume that the data are realizations from

$$(X_1, Y_1), \ldots, (X_n, Y_n),$$

where $X_i \in \mathbb{R}^p$ denotes a p-dimensional predictor variable and $Y_i \in \mathbb{R}$ a univariate response. In the sequel, we denote by $x^{(j)}$ the jth component of a vector $x \in \mathbb{R}^p$. We usually assume that the pairs (X_i, Y_i) are i.i.d. or from a stationary process. The goal is to estimate the regression function $F(x) = \mathbb{E}[Y|X = x]$ which is well known to be the (population) minimizer of the expected squared error loss $\mathbb{E}[(Y - F(X))^2]$.

The boosting methodology in general builds on a user-determined base procedure or weak learner and uses it repeatedly on modified data which are typically outputs from the previous iterations. The final boosted procedure takes the form of linear combinations of the base procedures. For L_2Boosting, based on the squared error loss, one simply fits the base procedure to the original data to start with, then uses the residuals from the previous iteration as the new response vector and refits the base procedure, and so on. As we will see in section 2.2, L_2Boosting is a "constrained" minimization of the empirical squared error risk $n^{-1}\sum_{i=1}^{n}(Y_i - F(X_i))^2$ (with respect to $F(\cdot)$) which yields an estimator $\hat{F}(\cdot)$. The regularization of the empirical risk minimization comes in implicitly by the choice of a base procedure and by algorithmical constraints such as early stopping or penalty barriers.

2.1 Base Procedures Which Do Variable Selection

To be more precise, a base procedure is in our setting a function estimator based on the data $\{(X_i, U_i);\ i = 1, \ldots, n\}$, where $U_1, \ldots, U_n$ denote some (pseudo-) response variables which are not necessarily the original $Y_1, \ldots, Y_n$. We denote the base procedure function estimator by

$$\hat{g}(\cdot) = \hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot), \tag{1}$$

where $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{U} = (U_1, \ldots, U_n)$.

Many base procedures involve some variable selection. That is, only some of the components of the p-dimensional predictor variables X_i are actually contributing in (1). In fact, almost all of the successful boosting algorithms in practice involve base procedures which do variable selection: examples include decision trees (see Freund and Schapire, 1996; Breiman, 1998; Friedman et al., 2000; Friedman, 2001), componentwise smoothing splines which involve selection of the best single predictor variable (see Bühlmann and Yu, 2003), or componentwise linear least squares in linear models with selection of the best single predictor variable (see Mallat and Zhang, 1993; Bühlmann, 2006).

It will be useful to represent the base procedure estimator (at the observed predictors X_i) as a hat-operator, mapping the (pseudo-) response to the fitted values:

$$\mathcal{H} : \mathbf{U} \mapsto (\hat{g}_{(\mathbf{X},\mathbf{U})}(X_1), \ldots, \hat{g}_{(\mathbf{X},\mathbf{U})}(X_n)),\ \mathbf{U} = (U_1, \ldots, U_n).$$

If the base procedure selects from a set of predictor variables, we denote the selected predictor variable index by $\hat{\mathcal{S}} \subset \{1, \ldots, p\}$, where $\hat{\mathcal{S}}$ has been estimated from a specified set Γ of subsets of variables. To emphasize this, we write for the hat operator of a base procedure

$$\mathcal{H}_{\hat{\mathcal{S}}} : \mathbf{U} \mapsto (\hat{g}_{(\mathbf{X}^{(\hat{\mathcal{S}})},\mathbf{U})}(X_1), \ldots, \hat{g}_{(\mathbf{X}^{(\hat{\mathcal{S}})},\mathbf{U})}(X_n)),\ \mathbf{U} = (U_1, \ldots, U_n), \tag{2}$$

where the base procedure $\hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot) = \hat{g}_{(\mathbf{X}^{(\hat{\mathcal{S}})},\mathbf{U})}(\cdot)$ depends only on the components $\mathbf{X}^{(\hat{\mathcal{S}})}$ from $\mathbf{X}$. The examples below illustrate this formalism.

Componentwise linear least squares in linear model (see Mallat and Zhang, 1993; Bühlmann, 2006)
We select only single variables at a time from $\Gamma = \{1, 2, \ldots, p\}$. The selector $\hat{\mathcal{S}}$ chooses the predictor variable which reduces the residual sum of squares most when using least squares fitting:

$$\hat{\mathcal{S}} = \text{argmin}_{1 \le j \le p} \sum_{i=1}^{n} (U_i - \hat{\gamma}_j X_i^{(j)})^2, \; \hat{\gamma}_j = \frac{\sum_{i=1}^{n} U_i X_i^{(j)}}{\sum_{i=1}^{n} (X_i^{(j)})^2} \; (j = 1, \ldots, p).$$

The base procedure is then

$$\hat{g}_{(\mathbf{X},\mathbf{U})}(x) = \hat{\gamma}_{\hat{\mathcal{S}}} x^{(\hat{\mathcal{S}})},$$

and its hat operator is given by the matrix

$$\mathcal{H}_{\hat{\mathcal{S}}} = \mathbf{X}^{(\hat{\mathcal{S}})} (\mathbf{X}^{(\hat{\mathcal{S}})})^T, \; \mathbf{X}^{(j)} = (X_1^{(j)}, \ldots, X_n^{(j)})^T.$$

L_2Boosting with this base procedure yields a linear model with model selection and parameter estimates which are shrunken towards zero. More details are given in sections 2.2 and 2.4.

Componentwise smoothing spline (see Bühlmann and Yu, 2003)
Similarly to a componentwise linear least squares fit, we select only one single variable at a time from $\Gamma = \{1, 2, \ldots, p\}$. The selector $\hat{\mathcal{S}}$ chooses the predictor variable which reduces residual sum of squares most when using a smoothing spline fit. That is, for a given smoothing spline operator with fixed degrees of freedom `d.f.` (which is the trace of the corresponding hat matrix)

$$\hat{\mathcal{S}} = \text{argmin}_{1 \le j \le p} \sum_{i=1}^{n} (U_i - \hat{g}_j(X_i^{(j)}))^2,$$

$\hat{g}_j(\cdot)$ is the fit from the smoothing spline to $\mathbf{U}$ versus $\mathbf{X}^{(j)}$ with `d.f.`

Note that we use the same degrees of freedom `d.f.` for all components j's. The hat-matrix corresponding to $\hat{g}_j(\cdot)$ is denoted by $\mathcal{H}_j$ which is symmetric; the exact from is not of particular interest here but is well known, see Green and Silverman (1994). The base procedure is

$$\hat{g}_{(\mathbf{X},\mathbf{U})}(x) = \hat{g}_{\hat{\mathcal{S}}}(x^{(\hat{\mathcal{S}})}),$$

and its hat operator is then given by a matrix $\mathcal{H}_{\hat{\mathcal{S}}}$. Boosting with this base procedure yields an additive model fit based on selected variables (see Bühlmann and Yu, 2003).

Pairwise thin plate splines
Generalizing the componentwise smoothing spline, we select pairs of variables from $\Gamma = \{(j,k);\ 1 \le j < k \le p\}$. The selector $\hat{\mathcal{S}}$ chooses the two predictor variables which reduce residual sum of squares most when using thin plate splines with two arguments:

$$\hat{\mathcal{S}} = \text{argmin}_{1 \le j < k \le p} \sum_{i=1}^{n} (U_i - \hat{g}_{j,k}(X_i^{(j)}, X_i^{(k)}))^2,$$

$\hat{g}_{j,k}(\cdot,\cdot)$ is an estimated thin plate spline based on $\mathbf{U}$ and $\mathbf{X}^{(j)}, \mathbf{X}^{(k)}$ with `d.f.`, where the degrees of freedom `d.f.` is the same for all components $j<k$. The hat-matrix corresponding to $\hat{g}_{j,k}$ is denoted by $\mathcal{H}_{j,k}$ which is symmetric; again the exact from is not of particular interest but can be found in Green and Silverman (1994). The base procedure is

$$\hat{g}_{(\mathbf{X},\mathbf{U})}(x) = \hat{g}_{\hat{\mathcal{S}}}(x^{(\hat{\mathcal{S}})}),$$

where $x^{(\hat{\mathcal{S}})}$ denotes the 2-dimensional vector corresponding to the selected pair in $\hat{\mathcal{S}}$, and the hat operator is then given by a matrix $\mathcal{H}_{\hat{\mathcal{S}}}$. Boosting with this base procedure yields a nonparametric fit with second order interactions based on selected pairs of variables; an illustration is given in section 3.4.

In all the examples above, the selector is given by

$$\hat{\mathcal{S}} = \mathrm{argmin}_{\mathcal{S}\in\Gamma} \sum_{i=1}^{n} (U_i - (\mathcal{H}_{\mathcal{S}}\mathbf{U})_i)^2 \qquad (3)$$

Also (small) regression trees can be cast into this framework. For example for stumps, $\Gamma = \{(j, c_{j,k});\ j=1,\ldots,p,\ k=1,\ldots,n-1\}$, where $c_{j,1} < \ldots < c_{j,n-1}$ are the mid-points between (non-tied) observed values $X_i^{(j)}$ $(i=1,\ldots,n)$. That is, Γ denotes here the set of selected single predictor variables and corresponding split-points. The parameter values for the two terminal nodes in the stump are then given by ordinary least squares which implies a linear hat matrix $\mathcal{H}_{(j,c_{j,k})}$. Note however, that for mid-size or large regression trees, the optimization over the set Γ is usually not done exhaustively.

2.2 L_2Boosting

Before introducing our new SparseL_2Boost algorithm, we describe first its less sparse counterpart L_2Boosting, a boosting procedure based on the squared error loss which amounts to repeated fitting of residuals with the base procedure $\hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot)$. Its derivation from a more general functional gradient descent algorithm using the squared error loss has been described by many authors, see Friedman (2001).

L_2Boosting

Step 1 (initialization). $\hat{F}_0(\cdot) \equiv 0$ and set $m=0$.

Step 2. Increase m by 1.
Compute residuals $U_i = Y_i - \hat{F}_{m-1}(X_i)$ $(i=1,\ldots,n)$ and fit the base procedure to the current residuals. The fit is denoted by $\hat{f}_m(\cdot) = \hat{g}_{(\mathbf{X},\mathbf{U})}(\cdot)$.
Update

$$\hat{F}_m(\cdot) = \hat{F}_{m-1}(\cdot) + \nu \hat{f}_m(\cdot),$$

where $0 < \nu \leq 1$ is a pre-specified step-size parameter.

Step 3 (iteration). Repeat Steps 2 and 3 until some stopping value for the number of iterations is reached.

With $m = 2$ and $\nu = 1$, L_2Boosting has already been proposed by Tukey (1977) under the name "twicing". The number of iterations is the main tuning parameter for L_2Boosting. Empirical evidence suggests that the choice for the step-size ν is much less crucial as long as ν is small; we usually use $\nu = 0.1$. The number of boosting iterations may be estimated by cross-validation. As an alternative, we will develop in section 2.5 an approach which allows to use some model selection criteria to bypass cross-validation.

2.3 SparseL_2Boost

As described above, L_2Boosting proceeds in a greedy way: if in Step2 the base procedure is fitted by least squares and when using $\nu = 1$, L_2Boosting pursues the best reduction of residual sum of squares in every iteration.

Alternatively, we may want to proceed such that the out-of-sample prediction error would be most reduced, that is we would like to fit a function $\hat{g}_{\mathbf{X},\mathbf{U}}$ (from the class of weak learner estimates) such that the out-of-sample prediction error becomes minimal. This is not exactly achievable since the out-sample prediction error is unknown. However, we can estimate it via a model selection criterion. To do so, we need a measure of complexity of boosting. Using the notation as in (2), the L_2Boosting operator in iteration m is easily shown to be (see Bühlmann and Yu, 2003)

$$\mathcal{B}_m = I - (I - \nu\mathcal{H}_{\hat{\mathcal{S}}_m}) \cdot \;\cdots\; \cdot (I - \nu\mathcal{H}_{\hat{\mathcal{S}}_1}), \tag{4}$$

where $\hat{\mathcal{S}}_m$ denotes the selector in iteration m. Moreover, if all the $\mathcal{H}_{\mathcal{S}}$ are linear (that is the hat matrix), as in all the examples given in section 2.1, L_2Boosting has an approximately linear representation, where only the data-driven selector $\hat{\mathcal{S}}$ brings in some additional nonlinearity. Thus, in many situations (for example the examples in the previous section 2.1 and decision tree base procedures), the boosting operator has a corresponding matrix-form when using in (4) the hat-matrices for $\mathcal{H}_{\mathcal{S}}$. The degrees of freedom for boosting are then defined as

$$\text{trace}(\mathcal{B}_m) = \text{trace}(I - (I - \nu\mathcal{H}_{\hat{\mathcal{S}}_m}) \cdots (I - \nu\mathcal{H}_{\hat{\mathcal{S}}_1})).$$

This is a standard definition for degrees of freedom (see Green and Silverman, 1994) and it has been used in the context of boosting in Bühlmann (2006). An estimate for the prediction error of L_2Boosting in iteration m can then be given in terms of the final prediction error criterion FPE_γ (Akaike, 1970):

$$\sum_{i=1}^{n} (Y_i - \hat{F}_m(X_i))^2 + \gamma \cdot \text{trace}(\mathcal{B}_m). \tag{5}$$

2.3.1 The SparseL_2Boost Algorithm

For SparseL_2Boost, the penalized residual sum of squares in (5) becomes the criterion to move from iteration $m-1$ to iteration m. More precisely, for $\mathcal{B}$ a (boosting) operator, mapping the response vector $\mathbf{Y}$ to the fitted variables, and a criterion $C(RSS, k)$, we use the following objective function to boost:

$$T(\mathbf{Y}, \mathcal{B}) = C\left(\sum_{i=1}^{n} (Y_i - (\mathcal{B}\mathbf{Y})_i)^2, \text{trace}(\mathcal{B})\right). \tag{6}$$

For example, the criterion could be FPE_γ for some $\gamma > 0$ which corresponds to

$$C_\gamma(RSS, k) = RSS + \gamma \cdot k. \tag{7}$$

An alternative which does not require the specification of a parameter γ as in (7) is advocated in section 2.5.

The algorithm is then as follows.

SparseL_2Boost

Step 1 (initialization). $\hat{F}_0(\cdot) \equiv 0$ and set $m = 0$.

Step 2. Increase m by 1.
Search for the best selector

$$\begin{aligned}
&\tilde{\mathcal{S}}_m = \operatorname{argmin}_{\mathcal{S} \in \Gamma} T(\mathbf{Y}, \operatorname{trace}(\mathcal{B}_m(\mathcal{S}))),\\
&\mathcal{B}_m(\mathcal{S}) = I - (I - \mathcal{H}_{\mathcal{S}})(I - \nu \mathcal{H}_{\tilde{\mathcal{S}}_{m-1}}) \cdots (I - \nu \mathcal{H}_{\tilde{\mathcal{S}}_1}),\\
&(\text{for } m = 1\text{: } \mathcal{B}_1(\mathcal{S}) = \mathcal{H}_{\mathcal{S}}).
\end{aligned}$$

Fit the residuals $U_i = Y_i - \hat{F}_{m-1}(X_i)$ with the base procedure using the selected $\tilde{\mathcal{S}}_m$ which yields a function estimate

$$\hat{f}_m(\cdot) = \hat{g}_{\tilde{\mathcal{S}}_m;(\mathbf{X},\mathbf{U})}(\cdot),$$

where $\hat{g}_{\mathcal{S};(\mathbf{X},\mathbf{U})}(\cdot)$ corresponds to the hat operator $\mathcal{H}_{\mathcal{S}}$ from the base procedure.

Step 3 (update). Update,

$$\hat{F}_m(\cdot) = \hat{F}_{m-1}(\cdot) + \nu \hat{f}_m(\cdot).$$

Step 4 (iteration). Repeat Steps 2 and 3 for a large number of iterations M.

Step 5 (stopping). Estimate the stopping iteration by

$$\hat{m} = \operatorname{argmin}_{1 \le m \le M} T(\mathbf{Y}, \operatorname{trace}(\mathcal{B}_m)), \quad \mathcal{B}_m = I - (I - \nu \mathcal{H}_{\tilde{\mathcal{S}}_m}) \cdots (I - \nu \mathcal{H}_{\tilde{\mathcal{S}}_1}).$$

The final estimate is $\hat{F}_{\hat{m}}(\cdot)$.

The only difference to L_2Boosting is that the selection in Step 2 yields a different $\tilde{\mathcal{S}}_m$ than in (3). While $\hat{\mathcal{S}}_m$ in (3) minimizes the residual sum of squares, the selected $\tilde{\mathcal{S}}_m$ in SparseL_2Boost minimizes a model selection criterion over all possible selectors. Since the selector $\tilde{\mathcal{S}}_m$ depends not only on the current residuals $\mathbf{U}$ but also explicitly on all previous boosting iterations through $\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2, \ldots, \tilde{\mathcal{S}}_{m-1}$ via the trace of $\mathcal{B}_m(\mathcal{S})$, the estimate $\hat{f}_m(\cdot)$ in SparseL_2Boost is not a function of the current residuals $\mathbf{U}$ only. This implies that we cannot represent SparseL_2Boost as a linear combination of base procedures, each of them acting on residuals only.

2.4 Connections to the Nonnegative Garrote Estimator

SparseL_2Boost based on C_γ as in (7) enjoys a surprising equivalence to the nonnegative garrote estimator (Breiman, 1995) in an orthogonal linear model. This special case allows explicit expressions to reveal clearly that SparseL_2Boost (aka nonnegative-garrote) is sparser than L_2Boosting (aka soft-thresholding).

Consider a linear model with n orthonormal predictor variables,

$$Y_i = \sum_{j=1}^{n} \beta_j x_i^{(j)} + \varepsilon_i,\ i = 1, \ldots, n,$$

$$\sum_{i=1}^{n} x_i^{(j)} x_i^{(k)} = \delta_{jk}, \tag{8}$$

where δ_{jk} denotes the Kronecker symbol, and $\varepsilon_1, \ldots, \varepsilon_n$ are i.i.d. random variables with $\mathbb{E}[\varepsilon_i] = 0$ and $\mathrm{Var}(\varepsilon_i) = \sigma_\varepsilon^2 < \infty$. We assume here the predictor variables as fixed and non-random. Using the standard regression notation, we can re-write model (8) as

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon,\ \ \mathbf{X}^T\mathbf{X} = \mathbf{X}\mathbf{X}^T = I, \tag{9}$$

with the $n \times n$ design matrix $\mathbf{X} = (x_i^{(j)})_{i,j=1,\ldots,n}$, the parameter vector $\beta = (\beta_1, \ldots, \beta_n)^T$, the response vector $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$ and the error vector $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)^T$. The predictors could also be basis functions $g_j(t_i)$ at observed values t_i with the property that they build an orthonormal system.

The nonnegative garrote estimator has been proposed by Breiman (1995) for a linear regression model to improve over subset selection. It shrinks each ordinary least squares (OLS) estimated coefficient by a nonnegative amount whose sum is subject to an upper bound constraint (the garrote). For a given response vector $\mathbf{Y}$ and a design matrix $\mathbf{X}$ (see (9)), the nonnegative garrote estimator takes the form

$$\hat{\beta}_{Nngar,j} = c_j \hat{\beta}_{OLS,j}$$

such that

$$\sum_{i=1}^{n} (Y_i - (\mathbf{X}\hat{\beta}_{Nngar})_i)^2 \text{ is minimized, subject to } \ c_j \geq 0,\ \sum_{j=1}^{p} c_j \leq s, \tag{10}$$

for some $s > 0$. In the orthonormal case from (8), since the ordinary least squares estimator is simply $\hat{\beta}_{OLS,j} = (\mathbf{X}^T\mathbf{Y})_j = Z_j$, the nonnegative garrote minimization problem becomes finding c_j's such that

$$\sum_{j=1}^{n} (Z_j - c_j Z_j)^2 \text{ is minimized, subject to } \ c_j \geq 0,\ \sum_{j=1}^{n} c_j \leq s.$$

Introducing a Lagrange multiplier $\tau > 0$ for the sum constraint gives the dual optimization problem: minimizing

$$\sum_{j=1}^{n} (Z_j - c_j Z_j)^2 + \tau \sum_{j=1}^{n} c_j, \quad c_j \geq 0\ (j = 1, \ldots, n). \tag{11}$$

This minimization problem has an explicit solution (Breiman, 1995):

$$c_j = (1 - \lambda/|Z_j|^2)^+,\ \lambda = \tau/2,$$

where $u^+ = \max(0, u)$. Hence $\hat{\beta}_{Nngar,j} = (1 - \lambda/|Z_j|^2)^+ Z_j$ or equivalently,

$$\hat{\beta}_{Nngar,j} = \begin{cases} Z_j - \lambda/|Z_j|, & \text{if } \mathrm{sign}(Z_j)Z_j^2 \geq \lambda, \\ 0, & \text{if } Z_j^2 < \lambda, \\ Z_j + \lambda/|Z_j|, & \text{if } \mathrm{sign}(Z_i)Z_j^2 \leq -\lambda. \end{cases}, \qquad \text{where } Z_j = (\mathbf{X}^T\mathbf{Y})_j. \tag{12}$$

We show in Figure 1 the nonnegative garrote threshold function in comparison to hard- and soft-thresholding, the former corresponding to subset variable selection and the latter to the Lasso (Tibshirani, 1996). Hard-thresholding either yields the value zero or the ordinary least squares estimator; the nonnegative garrote and soft-thresholding either yield the value zero or a shrunken ordinary least squares estimate, where the shrinkage towards zero is stronger for the soft-threshold than for the nonnegative garrote estimator. Therefore, for the same amount of "complexity" or "degrees of freedom" (which is in case of hard-thresholding the number of ordinary least squares estimated variables), hard-thresholding (corresponding to subset selection) will typically select the fewest number of variables (non-zero coefficient estimates) while the nonnegative garrote will include more variables and the soft-thresholding will be the least sparse in terms of the number of selected variables; the reason is that for the non-zero coefficient estimates, the shrinkage effect, which is slight in the nonnegative garotte and stronger for soft-thresholding, causes fewer degrees of freedom for every

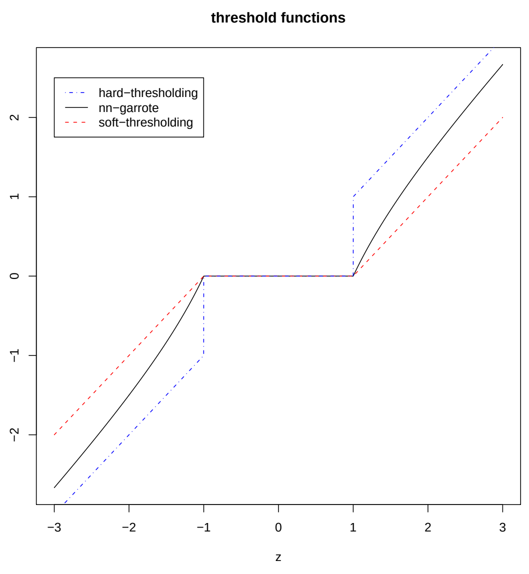

Figure 1: Threshold functions for subset selection or hard-thresholding (dashed-dotted line), non-negative garrote (solid line) and lasso or soft-thresholding (dashed line).

selected variable. This observation can also be compared with some numerical results in section 3.

The following result shows the equivalence of the nonnegative garrote estimator and SparseL_2Boost with componentwise linear least squares (using $\hat{m}$ iterations) yielding coefficient estimates $\hat{\beta}^{(\hat{m})}_{SparseBoost,j}$.

Theorem 1 *Consider the model in (8) and any sequence $(\gamma_n)_{n \in \mathbb{N}}$. For Sparse$L_2$Boost with componentwise linear least squares, based on C_{γ_n} as in (7) and using a step-size ν, as described in section 2.3, we have*

$$\hat{\beta}^{(\hat{m})}_{SparseBoost,j} = \hat{\beta}_{Nngar,j} \text{ in (12) with parameter } \lambda_n = \frac{1}{2}\gamma_n(1+e_j(\nu)),$$
$$\max_{1 \le i \le n} |e_j(\nu)| \le \nu/(1-\nu) \to 0 \ (\nu \to 0).$$

A proof is given in section 5. Note that the sequence $(\gamma_n)_{n \in \mathbb{N}}$ can be arbitrary and does not need to depend on n (and likewise for the corresponding λ_n). For the orthogonal case, Theorem 1 yields the interesting interpretation of SparseL_2Boost as the nonnegative garrote estimator.

We also describe here for the orthogonal case the equivalence of L_2Boosting with componentwise linear least squares (yielding coefficient estimates $\hat{\beta}^{(m)}_{Boost,j}$) to soft-thresholding. A closely related result has been given in Efron et al. (2004) for the forward stagewise linear regression method which is similar to L_2Boosting. However, our result is for (non-modified) L_2Boosting and brings out more explicitly the role of the step-size.

The soft-threshold estimator for the unknown parameter vector β, is

$$\hat{\beta}_{soft,j} = \begin{cases} Z_j - \lambda, & \text{if } Z_j \ge \lambda, \\ 0, & \text{if } |Z_j| < \lambda, \\ Z_j + \lambda, & \text{if } Z_j \le -\lambda. \end{cases} \qquad \text{where } Z_j = (\mathbf{X}^T \mathbf{Y})_j. \tag{13}$$

Theorem 2 *Consider the model in (8) and a threshold λ_n in (13) for any sequence $(\lambda_n)_{n \in \mathbb{N}}$. For L_2Boosting with componentwise linear least squares and using a step-size ν, as described in section 2.2, there exists a boosting iteration m, typically depending on λ_n, ν and the data, such that*

$$\hat{\beta}^{(m)}_{Boost,j} = \hat{\beta}_{soft,j} \text{ in (13) with threshold of the form } \lambda_n(1+e_j(\nu)), \text{ where}$$
$$\max_{1 \le j \le n} |e_j(\nu)| \le \nu/(1-\nu) \to 0 \ (\nu \to 0).$$

A proof is given in section 5. We emphasize that the sequence $(\lambda_n)_{n \in \mathbb{N}}$ can be arbitrary: in particular, λ_n does not need to depend on sample size n.

2.5 The gMDL choice for the criterion function

The FPE criterion function $C(\cdot,\cdot)$ in (7) requires in practice the choice of a parameter γ. In principle, we could tune this parameter using some cross-validation scheme. Alternatively, one could use a parameter value corresponding to well-known model selection criteria such as AIC ($\gamma = 2$) or BIC ($\gamma = \log n$). However, in general, the answer to whether to use AIC or BIC depends on the true underlying model being finite or not (see Speed and Yu, 1993, and the references therein). In practice, it is difficult to know which situation one is in and thus hard to choose between AIC and BIC. We employ here instead a relatively new minimum description length criterion, gMDL (see Hansen and Yu, 2001), developed for linear models. For each model class, roughly speaking, gMDL is derived as a mixture code length based on a linear model with an inverse Gamma prior (with a shape hyperparameter) for the variance and conditioning on the variance, the linear model parameter β follows an independent multivariate normal prior with the given variance multiplied by a scale hyperparameter. The two hyperparameters are then optimized based on the MDL principle and their coding costs are included in the code length. Because of the adaptive choices of the hyperparameters, the resulted gMDL criterion has a data-dependent penalty for each dimension, instead of the fixed penalty 2 or $\log n$ for AIC or BIC, respectively. In other words, gMDL bridges the AIC and BIC criteria by having a data-dependent penalty $\log(F)$ as given below in (14). The F in the gMDL penalty is related to the signal to noise ratio (SNR), as shown in Hansen and Yu (1999). Moreover, the gMDL criterion has an explicit analytical expression which depends only on the residual sum of squares and the model dimension or complexity. It is worth noting that we will not need to tune the criterion function as it will be explicitly given as a function of the data only. The gMDL criterion function takes the form

$$C_{gMDL}(RSS,k) = \log(S) + \frac{k}{n}\log(F),$$

$$S = \frac{RSS}{n-k},\ F = \frac{\sum_{i=1}^{n} Y_i^2 - RSS}{kS}. \tag{14}$$

Here, RSS denotes again the residual sum of squares as in formula (6) (first argument of the function $C(\cdot,\cdot)$).

In the SparseL_2Boost algorithm in section 2.3.1, if we take

$$T(\mathbf{Y},\mathcal{B}) = C_{gMDL}(RSS, \text{trace}(\mathcal{B})),$$

then we arrive at the **gMDL-SparseL_2Boost** algorithm. Often though, we simply refer to it as SparseL_2Boost.

The gMDL criterion in (14) can also be used to give a new stopping rule for L_2Boosting. That is, we propose

$$\hat{m} = \text{argmin}_{1\le m\le M} C_{gMDL}(RSS_m, \text{trace}(\mathcal{B}_m)), \tag{15}$$

where M is a large number, RSS_m the residual sum of squares after m boosting iterations and $\mathcal{B}_m$ is the boosting operator described in (4). If the minimizer is not unique, we use the minimal m which minimizes the criterion. Boosting can now be run without tuning any parameter (we typically do not tune over the step-size ν but rather take a value such as $\nu = 0.1$), and we call such an automatically stopped boosting method **gMDL-L_2Boosting**. In the sequel, it is simply referred to as L_2Boosting.

There will be no overall superiority of either SparseL_2Boost or L_2Boosting as shown in Section 3.1. But it is straightforward to do a data-driven selection: we choose the fitted model which has the smaller gMDL-score between gMDL-SparseL_2Boost and the gMDL stopped L_2Boosting. We term this method **gMDL-sel-L_2Boost** which does not rely on cross-validation and thus could bring much computational savings.

3. Numerical Results

In this section, we investigate and compare SparseL_2Boost with L_2Boosting (both with their data-driven gMDL-criterion), and evaluate gMDL-sel-L_2Boost. The step-size in both boosting methods is fixed at $\nu = 0.1$. The simulation models are based on two high-dimensional linear models and one nonparametric model. Except for two real data sets, all our comparisons and results are based on 50 independent model simulations.

3.1 High-Dimensional Linear Models

3.1.1 ℓ^0-SPARSE MODELS

Consider the model

$$Y = 1 + 5X_1 + 2X_2 + X_9 + \varepsilon,$$
$$X = (X_1, \ldots, X_{p-1}) \sim \mathcal{N}_{p-1}(0, \Sigma),\ \varepsilon \sim \mathcal{N}(0,1), \tag{16}$$

where ε is independent from X. The sample size is chosen as $n = 50$ and the predictor-dimension is $p \in \{50, 100, 1000\}$. For the covariance structure of the predictor X, we consider two cases:

$$\Sigma = I_{p-1}, \tag{17}$$
$$[\Sigma]_{ij} = 0.8^{|i-j|}. \tag{18}$$

The models are ℓ^0-sparse, since the ℓ^0-norm of the true regression coefficients (the number of effective variables including an intercept) is 4.

The predictive performance is summarized in Table 1. For the ℓ^0-sparse model (16), SparseL_2Boost outperforms L_2Boosting. Furthermore, in comparison to the oracle performance (denoted by an asterisk $*$ in Table 1), the gMDL rule for the stopping iteration $\hat{m}$ works very well for the lower-dimensional cases with $p \in \{50, 100\}$ and it is still reasonably accurate for the very high-dimensional case with $p = 1000$. Finally, both boosting methods are essentially insensitive when increasing the

Σ , dim.	SparseL_2Boost	L_2Boosting	SparseL_2Boost*	L_2Boosting*
(17), $p = 50$	0.16 (0.018)	0.46 (0.041)	0.16 (0.018)	0.46 (0.036)
(17), $p = 100$	0.14 (0.015)	0.52 (0.043)	0.14 (0.015)	0.48 (0.045)
(17), $p = 1000$	0.77 (0.070)	1.39 (0.102)	0.55 (0.064)	1.27 (0.105)
(18), $p = 50$	0.21 (0.024)	0.31 (0.027)	0.21 (0.024)	0.30 (0.026)
(18), $p = 100$	0.22 (0.024)	0.39 (0.028)	0.22 (0.024)	0.39 (0.028)
(18), $p = 1000$	0.45 (0.035)	0.97 (0.052)	0.38 (0.030)	0.72 (0.049)

Table 1: Mean squared error (MSE), $\mathbb{E}[(\hat{f}(X) - f(X))^2]$ ($f(x) = \mathbb{E}[Y|X = x]$), in model (16) for gMDL-SparseL_2Boost and gMDL early stopped L_2Boosting using the estimated stopping iteration $\hat{m}$. The performance using the oracle m which minimizes MSE is denoted by an asterisk *. Estimated standard errors are given in parentheses. Sample size is $n = 50$.

number of ineffective variables from 46 ($p = 50$) to 96 ($p = 100$). However, with very many, that is 996 ($p = 1000$), ineffective variables, a significant loss in accuracy shows up in the orthogonal design (17) and there is an indication that the relative differences between SparseL_2Boost and L_2Boosting become smaller. For the positive dependent design in (18), the loss in accuracy in the $p = 1000$ case is not as significant as in the orthogonal design case in (17), and the relative differences between SparseL_2Boost and L_2Boosting actually become larger.

It is also worth pointing out that the resulting mean squared errors (MSEs) in design (17) and (18) are not really comparable even for the same number p of predictors. This is because, even though the noise level is $\mathbb{E}|\varepsilon|^2 = 1$ for both designs, the signal levels $\mathbb{E}|f(X)|^2$ are different, that is

31 for the uncorrelated design in (17) and 49.5 for the correlated design in (18). If we would like to compare the performances among the two designs, we should rather look at the signal-adjusted mean squared error

$$\frac{\mathbb{E}|\hat{f}(X) - f(X)|^2}{\mathbb{E}|f(X)|^2}$$

which is the test-set analogue of $1 - R^2$ in linear models. This signal adjusted error measure can be computed from the results in Table 1 and the signal levels given above. We then obtain for the lower dimensional cases with $p \in \{50, 100\}$ that the prediction accuracies are about the same for the correlated and the uncorrelated design (for SparseL_2Boost and for L_2Boosting). However, for the high-dimensional case with $p = 1000$, the performance (of SparseL_2Boost and of L_2Boosting) is significantly better in the correlated than the uncorrelated design.

Next, we consider the ability of selecting the correct variables: the results are given in Table 2.

Σ , dim.		SparseL_2Boost	L_2Boosting
(17), $p = 50$:	ℓ^0-norm	5.00 (0.125)	13.68 (0.438)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	1.00 (0.125)	9.68 (0.438)
(17), $p = 100$:	ℓ^0-norm	5.78 (0.211)	21.20 (0.811)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	1.78 (0.211)	17.20 (0.811)
(17), $p = 1000$:	ℓ^0-norm	23.70 (0.704)	78.80 (0.628)
	non-selected T	0.02 (0.020)	0.02 (0.020)
	selected F	19.72 (0.706)	74.82 (0.630)
(18), $p = 50$:	ℓ^0-norm	4.98 (0.129)	9.12 (0.356)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	0.98 (0.129)	5.12 (0.356)
(18), $p = 100$:	ℓ^0-norm	5.50 (0.170)	12.44 (0.398)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	1.50 (0.170)	8.44 (0.398)
(18), $p = 1000$:	ℓ^0-norm	13.08 (0.517)	71.68 (1.018)
	non-selected T	0.00 (0.000)	0.00 (0.000)
	selected F	9.08 (0.517)	67.68 (1.018)

Table 2: Model (16): expected number of selected variables (ℓ^0-norm), expected number of non-selected true effective variables (non-selected T) which is in the range of $[0, 4]$, and expected number of selected non-effective (false) variables (selected F) which is in the range of $[0, p-4]$. Methods: SparseL_2Boost and L_2Boosting using the estimated stopping iteration $\hat{m}$ (Step 5 in the SparseL_2Boost algorithm and (15) respectively). Estimated standard errors are given in parentheses. Sample size is $n = 50$.

In the orthogonal case, we have argued that SparseL_2Boost has a tendency for sparser results than L_2Boosting; see the discussion of different threshold functions in section 2.4. This is confirmed in all our numerical experiments. In particular, for our ℓ^0-sparse model (16), the detailed results are reported in Table 2. SparseL_2Boost selects much fewer predictors than L_2Boosting. Moreover, for this model, SparseL_2Boost is a good model selector as long as the dimensionality is not very large, that is for $p \in \{50, 100\}$, while L_2Boosting is much worse selecting too many false predictors (that is too many false positives). For the very high-dimensional case with $p = 1000$, the selected models are clearly too large when compared with the true model size, even when using SparseL_2Boost. However, the results are pretty good considering the fact that we are dealing with a much harder problem of getting rid of 996 irrelevant predictors based on only 50 sample points. To summarize, for this synthetic example, SparseL_2Boost works significantly better than L_2Boosting both in terms of MSE, model selection and sparsity, due to the sparsity of the true model.

3.1.2 A NON-SPARSE MODEL WITH RESPECT TO THE ℓ^0-NORM

We provide here an example where L_2Boosting will be better than SparseL_2Boost. Consider the model

$$Y = \sum_{j=1}^{p} \frac{1}{5} \beta_j X_j + \varepsilon,$$
$$X_1, \ldots, X_p \sim \mathcal{N}_p(0, I_p),\ \varepsilon \sim \mathcal{N}(0,1), \tag{19}$$

where $\beta_1, \ldots, \beta_p$ are fixed values from i.i.d. realizations of the double-exponential density $p(x) = \exp(-|x|)/2$. The magnitude of the coefficients $|\beta_j|/5$ is chosen to vary the signal to noise ratio from model (16), making it about 5 times smaller than for (19). Since Lasso (coinciding with L_2Boosting in the orthogonal case) is the maximum a-posteriori (MAP) method when the coefficients are from a double-exponential distribution and the observations from a Gaussian distribution, as in (19), we expect L_2Boosting to be better than SparseL_2Boost for this example (even though we understand that MAP is not the Bayesian estimator under the L^2 loss). The squared error performance is given in Table 3, supporting our expectations. SparseL_2Boost nevertheless still has the virtue of sparsity with only about 1/3 of the number of selected predictors but with an MSE which is larger by a factor 1.7 when compared with L_2Boosting.

	SparseL_2Boost	L_2Boosting	SparseL_2Boost*	L_2Boosting*
MSE	3.64 (0.188)	2.19 (0.083)	3.61 (0.189)	2.08 (0.078)
ℓ^0-norm	11.78 (0.524)	29.16 (0.676)	11.14 (0.434)	35.76 (0.382)

Table 3: Mean squared error (MSE) and expected number of selected variables (ℓ^0-norm) in model (19) with $p = 50$. Estimated standard errors are given in parentheses. All other specifications are described in the caption of Table 1.

3.1.3 DATA-DRIVEN CHOICE BETWEEN SPARSEL_2BOOST AND L_2BOOSTING: GMDL-SEL-L_2BOOST

We illustrate here the gMDL-sel-L_2Boost proposal from section 2.5 that uses the gMDL model selection score to choose in a data-driven way between SparseL_2Boost and L_2Boosting. As an illustration, we consider again the models in (16)-(17) and (19) with $p = 50$ and $n = 50$. Figure 2 displays the results in the form of boxplots across 50 rounds of simulations.

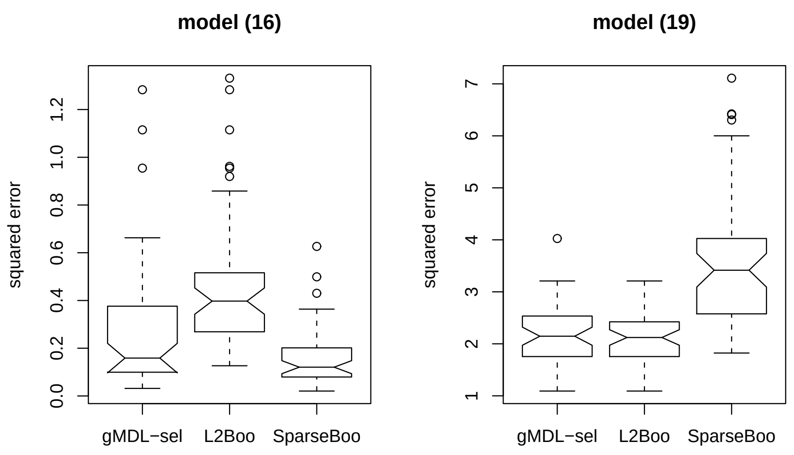

Figure 2: Out-of-sample squared error losses, $\text{ave}_X[(\hat{f}(X) - f(X))^2]$ ($f(x) = \mathbb{E}[Y|X = x]$), from the 50 simulations for the models in (16)-(17) and (19) with $p = 50$. gMDL-sel-L_2Boost (gMDL-sel), L_2Boosting (L2Boo) and SparseL_2Boost (SparseBoo). Sample size is $n = 50$.

The gMDL-sel-L_2Boost method performs between the better and the worse of the two boosting algorithms, but closer to the better performer in each situation (the latter is only known for simulated data sets). For model (19), there is essentially no degraded performance when doing a data-driven selection between the two boosting algorithms (in comparison to the best performer).

3.2 Ozone Data with Interactions Terms

We consider a real data set about ozone concentration in the Los Angeles basin. There are $p = 8$ meteorological predictors and a real-valued response about daily ozone concentration; see Breiman (1996). We constructed second-order interaction and quadratic terms after having centered the original predictors. We then obtain a model with $p = 45$ predictors (including an intercept) and a response. We used 10-fold cross-validation to estimate the out-of-sample squared prediction error and the average number of selected predictor variables. When scaling the predictor variables (and their interactions) to zero mean and variance one, the performances were very similar. Our results are comparable to the analysis of bagging in Breiman (1996) which yielded a cross-validated squared error of 18.8 for bagging trees based on the original eight predictors.

We also run SparseL_2Boost and L_2Boosting on the whole data set and choose the method according to the better gMDL-score, that is gMDL-sel-L_2Boost (see section 2.5). Some results are given in Table 5. Based on SparseL_2Boost, an estimate for the error variance is $n^{-1}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = 15.56$

	SparseL_2Boost	L_2Boosting
10-fold CV squared error	16.52	16.57
10-fold CV ℓ^0-norm	10.20	16.10

Table 4: Boosting with componentwise linear least squares for ozone data with first order-interactions ($n = 330,\ p = 45$). Squared prediction error and average number of selected predictor variables using 10-fold cross-validation.

and the goodness of fit equals $R^2 = \sum_{i=1}^n (\hat{Y}_i - \overline{Y})^2 / \sum_{i=1}^n (Y_i - \overline{Y})^2 = 0.71$, where $\hat{Y}_i = \hat{F}(X_i)$ and $\overline{Y} = n^{-1} \sum_{i=1}^n Y_i$.

	SparseL_2Boost (#)	L_2Boosting
gMDL-score	2.853	2.862
RSS	15.56	15.24
ℓ^0-norm	10	18

Table 5: Boosting with componentwise linear least squares for ozone data with first order-interactions ($n = 330,\ p = 45$). gMDL-score, $n^{-1}\times$ residual sum of squares (RSS) and number of selected terms (ℓ^0-norm). (#) gMDL-sel-L_2Boost selects SparseL_2Boost as the better method.

In summary, while SparseL_2Boost is about as good as L_2Boosting in terms of predictive accuracy, see Table 4, it yields a sparser model fit, see Tables 4 and 5.

3.3 Binary Tumor Classification Using Gene Expressions

We consider a real data set which contains $p = 7129$ gene expressions in 49 breast tumor samples using the Affymetrix technology, see West et al. (2001). After thresholding to a floor of 100 and a ceiling of 16,000 expression units, we applied a base 10 log-transformation and standardized each experiment to zero mean and unit variance. For each sample, a binary response variable $Y \in \{0,1\}$ is available, describing the status of lymph node involvement in breast cancer. The data are available at `http://mgm.duke.edu/genome/dna_micro/work/`.

Although the data has the structure of a binary classification problem, the squared error loss is quite often employed for estimation. We use L_2Boosting and SparseL_2Boost with componentwise linear least squares. We classify the label 1 if $\hat{p}(x) = \hat{\mathbb{P}}[Y + 1 | X = x] > 1/2$ and zero otherwise. The estimate for $\hat{p}(\cdot)$ is obtained as follows:

$$\begin{aligned} &\hat{p}_m(\cdot) = 1/2 + \hat{F}_m(\cdot), \\ &\hat{F}_m(\cdot) \text{ the } L_2\text{- or Sparse}L_2\text{Boost estimate using } \tilde{Y} = Y - 1/2. \end{aligned} \tag{20}$$

Note that $\hat{F}_m(\cdot)$ is an estimate of $p(\cdot) - 1/2$. Using this procedure amounts to modelling and estimating the deviation from the boundary value 1/2 (we do not use an intercept term anymore in our model). This is usually much better because the L_2- or SparseL_2Boost estimate is shrunken towards zero. When using L_2- or SparseL_2Boost on $Y \in \{0,1\}$ directly, with an intercept term, we would obtain a shrunken boosting estimate of the intercept introducing a bias rendering $\hat{p}(\cdot)$ to be systematically too small. The latter approach has been used in Bühlmann (2006) yielding worse results for L_2Boosting than what we report here for L_2Boosting using (20).

Since the gMDL criterion is relatively new, its classification counterpart is not yet well developed (see Hansen and Yu, 2002). Instead of the gMLD criterion in (14) and (15), we use the BIC score for the Bernoulli-likelihood in a binary classification:

$$BIC(m) = -2 \cdot \text{log-likelihood} + \log(n) \cdot \text{trace}(\mathcal{B}_m).$$

The AIC criterion would be another option: it yields similar, a bit less sparse results for our tumor classification problem.

We estimate the classification performance by a cross-validation scheme where we randomly divide the 49 samples into balanced training- and test-data of sizes $2n/3$ and $n/3$, respectively, and we repeat this 50 times. We also report on the average of selected predictor variables. The reports are given in Table 6.

	SparseL_2Boost	L_2Boosting
CV misclassification error	21.88%	23.13%
CV ℓ^0-norm	12.90	15.30

Table 6: Boosting with componentwise linear least squares for tumor classification data ($n = 46,\ p = 7129$). Misclassification error and average number of selected predictor variables using cross-validation (with random 2/3 training and 1/3 test sets).

The predictive performance of L_2- and SparseL_2Boosting compares favourably with four other methods, namely 1-nearest neighbors, diagonal linear discriminant analysis, support vector machine with radial basis kernel (from the R-package `e1071` and using its default values), and a forward selection penalized logistic regression model (using some reasonable penalty parameter and number of selected genes). For 1-nearest neighbors, diagonal linear discriminant analysis and support vector machine, we pre-select the 200 genes which have the best Wilcoxon score in a two-sample problem (estimated from the training data set only), which is recommended to improve the classification performance. Forward selection penalized logistic regression is run without pre-selection of genes. The results are given in Table 5 which is taken from Bühlmann (2006).

	FPLR	1-NN	DLDA	SVM
CV misclassification error	35.25%	43.25%	36.12%	36.88%

Table 7: Cross-validated misclassification rates for lymph node breast cancer data. Forward variable selection penalized logistic regression (FPLR), 1-nearest-neighbor rule (1-NN), diagonal linear discriminant analysis (DLDA) and a support vector machine (SVM)

When using SparseL_2Boost and L_2Boosting on the whole data set, we get the following results displayed in Table 8. The 12 variables (genes) which are selected by SparseL_2Boost are a subset of the 14 selected variables (genes) from L_2Boosting. Analogously as in section 3.2, we give some ANOVA-type numbers of SparseL_2Boosting: the error variability is $n^{-1}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2 = 0.052$ and the goodness of fit equals $R^2 = \sum_{i=1}^{n}(\hat{Y}_i-\overline{Y})^2/\sum_{i=1}^{n}(Y_i-\overline{Y})^2 = 0.57$, where $\hat{Y}_i = \hat{F}(X_i)$ and $\overline{Y} = n^{-1}\sum_{i=1}^{n} Y_i$.

	SparseL_2Boost (#)	L_2Boosting
BIC score	35.09	37.19
RSS	0.052	0.061
ℓ^0-norm	12	14

Table 8: Boosting with componentwise linear least squares for tumor classification ($n = 49,\ p = 7129$). BIC score, $n^{-1}\times$ residual sum of squares (RSS) and number of selected terms (ℓ^0-norm). (#) BIC-sel-L_2Boost selects SparseL_2Boost as the better method.

In summary, the predictive performance of SparseL_2Boost is slightly better than of L_2Boosting, see Table 6, and SparseL_2Boost selects a bit fewer variables (genes) than L_2Boosting, see Tables 7 and 8.

3.4 Nonparametric Function Estimation with Second-Order Interactions

Consider the Friedman #1 model Friedman (1991),

$$\begin{aligned}&Y = 10\sin(\pi X_1 X_2) + 20(X_3-0.5)^2 + 10X_4 + 5X_5 + \varepsilon,\\ &X \sim \text{Unif.}([0,1]^p),\ \varepsilon \sim \mathcal{N}(0,1),\end{aligned} \tag{21}$$

where ε is independent from X. The sample size is chosen as $n = 50$ and the predictor dimension is $p \in \{10, 20\}$ which is still large relative to n for a nonparametric problem.

SparseL_2Boost and L_2Boosting with a pairwise thin plate spline, which selects the best pair of predictor variables yielding lowest residual sum of squares (when having the same degrees of freedom $\mathtt{d.f.} = 5$ for every thin plate spline), yields a second-order interaction model; see also section 2.1. We demonstrate in Table 9 the effectiveness of these procedures, also in comparison with the MARS Friedman (1991) fit constrained to second-order interaction terms. SparseL_2Boost is a bit better than L_2Boosting. But the estimation of the boosting iterations by gMDL did not do as well as in section 3.1 since the oracle methods perform significantly better. The reason is that this example has a high signal to noise ratio. From (Hansen and Yu, 1999), the F in the gMDL penalty (see (14)) is related to the signal to noise ratio (SNR). Thus, when SNR is high, the $\log(F)$ is high too, leading to too small models in both SparseL_2Boost and L_2Boosting: that is, this large penalty forces both SparseL_2Boost and L_2Boosting to stop too early in comparison to the oracle stopping iteration which minimizes MSE. However, both boosting methods nevertheless are quite a bit better than MARS.

When increasing the noise level, using $\text{Var}(\varepsilon) = 16$, we obtain the following MSEs for $p = 10$: 11.70 for SparseL_2Boost, 11.65 for SparseL_2Boost* with the oracle stopping rule and 24.11 for MARS. Thus, for lower signal to noise ratios, stopping the boosting iterations with the gMDL criterion works very well, and our SparseL_2Boost algorithm is much better than MARS.

dim.	SparseL_2Boost	L_2Boosting	MARS	SparseL_2Boost*	L_2Boosting*
$p = 10$	3.71 (0.241)	4.10 (0.239)	5.79 (0.538)	2.22 (0.220)	2.69 (0.185)
$p = 20$	4.36 (0.238)	4.81 (0.197)	5.82 (0.527)	2.68 (0.240)	3.56 (0.159)

Table 9: Mean squared error (MSE) in model (21). All other specifications are described in the caption of Table 1, except for MARS which is constrained to second-order interaction terms.

4. Conclusions

We propose SparseL_2Boost, a gradient descent algorithm on a penalized squared error loss which yields sparser solutions than L_2Boosting or ℓ^1-regularized versions thereof. The new method is mainly useful for high-dimensional problems with many ineffective predictor variables (noise variables). Moreover, it is computationally feasible in high dimensions, for example having linear complexity in the number of predictor variables p when using componentwise linear least squares or componentwise smoothing splines (see section 2.1).

SparseL_2Boost is essentially as generic as L_2Boosting and can be used in connection with nonparametric base procedures (weak learners). The idea of sparse boosting could also be transferred to boosting algorithms with other loss functions, leading to sparser variants of AdaBoost and LogitBoost.

There is no general superiority of sparse boosting over boosting, even though we did find in four out of our five examples (two real data and two synthetic data sets) that SparseL_2Boost outperforms L_2Boosting in terms of sparsity and SparseL_2Boost is as good or better than L_2Boosting in terms of predictive performance. In the synthetic data example in section 3.1.2, chosen to be the ideal situation for L_2Boosting, SparseL_2Boost loses 70% in terms of MSE, but uses only 1/3 of the predictors. Hence if one cares about sparsity, SparseL_2Boost seems a better choice than L_2Boosting. In our framework, the boosting approach automatically comes with a reasonable notion for statistical complexity or degrees of freedom, namely the trace of the boosting operator when it can be expressed in hat matrix form. This trace complexity is well defined for many popular base procedures (weak learners) including componentwise linear least squares and decision trees, see also section 2.1. SparseL_2Boost gives rise to a direct, fast computable estimate of the out-of-sample error when combined with the gMDL model selection criterion (and thus, by-passing cross-validation). This out-of-sample error estimate can also be used for choosing the stopping iteration in L_2Boosting and for selecting between sparse and traditional boosting, resulting in the gMDL-sel-L_2Boost algorithm.

Theoretical results in the orthogonal linear regression model as well as simulation and data experiments are provided to demonstrate that the SparseL_2Boost indeed gives sparser model fits than L_2Boosting and that gMDL-sel-L_2Boost automatically chooses between the two to give a rather satisfactory performance in terms of sparsity and prediction.

5. Proofs

We first give the proof of Theorem 2. It then serves as a basis for proving Theorem 1.

Proof of Theorem 2. We represent the componentwise linear least squares base procedure as a hat operator $\mathcal{H}_{\hat{S}}$ with $\mathcal{H}_j = \mathbf{x}^{(j)}(\mathbf{x}^{(j)})^T$, where $\mathbf{x}^{(j)} = (x_1^{(j)},\ldots,x_n^{(j)})^T$; see also section 2.1. The L_2Boosting operator in iteration m is then given by the matrix

$$\mathcal{B}_m = I - (I-\nu\mathcal{H}_1)^{m_1}(I-\nu\mathcal{H}_2)^{m_2}\cdots(I-\nu\mathcal{H}_n)^{m_n},$$

where m_i equals the number of times that the ith predictor variable has been selected during the m boosting iterations; and hence $m = \sum_{i=1}^n m_i$. The derivation of the formula above is straightforward because of the orthogonality of the predictors $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$ which implies the commutation $\mathcal{H}_j\mathcal{H}_k = \mathcal{H}_k\mathcal{H}_j$. Moreover, $\mathcal{B}_m$ can be diagonalized

$$\mathcal{B}_m = \mathbf{X}D_m\mathbf{X}^T \text{ with } \mathbf{X}^T\mathbf{X} = \mathbf{X}\mathbf{X}^T = I,\ D_m = \mathrm{diag}(d_{m,1},\ldots,d_{m,n}),\ d_{m,i} = 1-(1-\nu)^{m_i}.$$

Therefore, the residual sum of squares in the mth boosting iteration is:

$$RSS_m = \|\mathbf{Y}-\mathcal{B}_m\mathbf{Y}\|^2 = \|\mathbf{X}^T\mathbf{Y}-\mathbf{X}^T\mathcal{B}_m\mathbf{Y}\|^2 = \|Z-D_mZ\|^2 = \|(I-D_m)Z\|^2,$$

where $Z = \mathbf{X}^T\mathbf{Y}$.

The RSS_m decreases monotonically in m. Moreover, the amount of decrease $RSS_m - RSS_{m+1}$ is decaying monotonously in m, because L_2Boosting proceeds to decrease the RSS as much as possible in every step (by selecting the most reducing predictor $\mathbf{x}^{(j)}$) and due to the structure of $(1-d_{m,i}) = (1-\nu)^{m_i}$. Thus, every stopping of boosting with an iteration number m corresponds to a tolerance δ^2 such that

$$\begin{aligned} &RSS_k - RSS_{k+1} > \delta^2,\ k=1,2,...,m-1,\\ &RSS_m - RSS_{m+1} \le \delta^2, \end{aligned} \tag{22}$$

that is, the iteration number m corresponds to a numerical tolerance where the difference $RSS_m - RSS_{m+1}$ is smaller than δ^2.

Since L_2Boosting changes only one of the summands in RSS_m in the boosting iteration $m+1$, the criterion in (22) implies that for all $i \in \{1,\ldots,n\}$

$$\begin{aligned} &((1-\nu)^{2(m_i-1)} - (1-\nu)^{2m_i})Z_i^2 > \delta^2,\\ &((1-\nu)^{2m_i} - (1-\nu)^{2(m_i+1)})Z_i^2 \le \delta^2. \end{aligned} \tag{23}$$

If $m_i = 0$, only the second line in the above expression is relevant. The L_2Boosting solution with tolerance δ^2 is thus characterized by (23).

Let us first, for the sake of insight, replace the "$\le$" in (23) by "$\approx$": we will deal later in which sense such an approximate equality holds. If $m_i \ge 1$, we get

$$((1-\nu)^{2m_i} - (1-\nu)^{2(m_i+1)})Z_i^2 = (1-\nu)^{2m_i}(1-(1-\nu)^2)Z_i^2 \approx \delta^2,$$

and hence

$$(1-\nu)^{m_i} \approx \frac{\delta}{\sqrt{1-(1-\nu)^2}|Z_i|}. \tag{24}$$

In case where $m_i = 0$, we obviously have that $1-(1-\nu)^{m_i} = 0$. Therefore,

$$\hat{\beta}^{(m)}_{Boost,i} = \hat{Z}_i = d_{m,i} = (1-(1-\nu)^{m_i})Z_i \approx Z_i - \frac{\delta}{\sqrt{1-(1-\nu)^2}|Z_i|}Z_i \quad \text{if } m_1 \geq 1,$$
$$\hat{\beta}^{(m)}_{Boost,i} = 0 \quad \text{if } m_i = 0.$$

Since $m_i = 0$ happens only if $|Z_i| \leq \frac{\delta}{\sqrt{1-(1-\nu)^2}}$, we can write the estimator as

$$\hat{\beta}^{(m)}_{Boost,i} \approx \begin{cases} Z_i - \lambda, & \text{if } Z_i \geq \lambda, \\ 0, & \text{if } |Z_i| < \lambda, \\ Z_i + \lambda, & \text{if } Z_i \leq -\lambda. \end{cases} \tag{25}$$

where $\lambda = \frac{\delta}{\sqrt{1-(1-\nu)^2}}$ (note that m is connected to δ, and hence to λ via the criterion in (22)). This is the soft-threshold estimator with threshold λ, as in (13). By choosing $\delta = \lambda_n\sqrt{1-(1-\nu)^2}$, we get the desired threshold λ_n.

We will now deal with the approximation in (24). By the choice of δ two lines above, we would like that

$$(1-\nu)^{m_i} \approx \lambda_n/|Z_i|.$$

As we will see, this approximation is accurate when choosing ν small. We only have to deal with the case where $|Z_i| > \lambda_n$; if $|Z_i| \leq \lambda_n$, we know that $m_i = 0$ and $\hat{\beta}_i = 0$ exactly, as claimed in the right hand side of (25). Denote by

$$V_i = V(Z_i) = \frac{\lambda_n}{|Z_i|} \in (0,1).$$

(The range $(0,1)$ holds for the case we are considering here). According to the stopping criterion in (23), the derivation as for (24) and the choice of δ, this says that

$$\begin{aligned} &(1-\nu)^{m_i} > V_i, \\ &(1-\nu)^{m_i+1} \leq V_i, \end{aligned} \tag{26}$$

and hence

$$\begin{aligned} \Delta(\nu, V_i) &\overset{\text{def}}{=} ((1-\nu)^{m_i} - V_i) \leq ((1-\nu)^{m_i} - (1-\nu)^{m_i+1}) \\ &= \frac{\nu}{1-\nu}(1-\nu)^{m_i+1} \leq \frac{\nu}{1-\nu}V_i, \end{aligned}$$

by using (26). Thus,

$$\begin{aligned} &(1-\nu)^{m_i} = V_i + ((1-\nu)^{m_i} - V_i) = V_i(1+\Delta(\nu,V_i)/V_i) = V_i(1+e_i(\nu)), \\ &|e_i(\nu)| = |\Delta(\nu,V_i)/V_i| \leq \nu/(1-\nu). \end{aligned} \tag{27}$$

Thus, when multiplying with $(-1)Z_i$ and adding Z_i,

$$\begin{aligned} \hat{\beta}^{(m)}_{Boost,i} &= (1-(1-\nu)^{m_i})Z_i = Z_i - Z_iV_i(1+e_i(\nu)) \\ &= \text{soft-threshold estimator with threshold } \lambda_n(1+e_i(\nu)), \end{aligned}$$

where $\max_{1 \le i \le n} |e_i(\nu)| \le \nu/(1-\nu)$ as in (27). $\square$

Proof of Theorem 1. The proof is based on similar ideas as for Theorem 2. The SparseL_2Boost in iteration m aims to minimize

$$MSB_m = RSS_m + \gamma_n \mathrm{trace}(\mathcal{B}_m) = \|\mathbf{Y} - \mathbf{X}\hat{\beta}^{(m)}_{ms-boost}\|^2 + \gamma_n \mathrm{trace}(\mathcal{B}_m).$$

When using the orthogonal transformation by multiplying with $\mathbf{X}^T$, the criterion above becomes

$$MSB_m = \|Z - \hat{\beta}^{(m)}_{ms-boost}\|^2 + \gamma_n \mathrm{trace}(\mathcal{B}_m),$$

where $\mathrm{trace}(\mathcal{B}_m) = \sum_{i=1}^n (1-(1-\nu)^{m_i})$. Moreover, we run SparseL_2Boost until the stopping iteration m satisfies the following:

$$\begin{aligned} &MSB_k - MSB_{k+1} > 0,\ k = 1,2,\ldots,m-1, \\ &MSB_m - MSB_{m+1} \le 0. \end{aligned} \tag{28}$$

It is straightforward to see for the orthonormal case, that such an m coincides with the definition for $\hat{m}$ in section 2.3. Since SparseL_2Boost changes only one of the summands in RSS and the trace of $\mathcal{B}_m$, the criterion above implies that for all $i = 1,\ldots,n$, using the definition of MSB,

$$\begin{aligned} &(1-\nu)^{2(m_i-1)} Z_i^2 (1-(1-\nu)^2) - \gamma_n \nu (1-\nu)^{m_i-1} > 0, \\ &(1-\nu)^{2m_i} Z_i^2 (1-(1-\nu)^2) - \gamma_n \nu (1-\nu)^{m_i} \le 0. \end{aligned} \tag{29}$$

Note that if $|Z_i|^2 \le \gamma_n \nu/(1-(1-\nu)^2)$, then $m_i = 0$. This also implies uniqueness of the iteration m such that (28) holds or of the m_i such that (29) holds.

Similarly to the proof of Theorem 2, we look at this expression first in terms of an approximate equality to zero, that is ≈ 0. We then immediately find that

$$(1-\nu)^{m_i} \approx \frac{\gamma_n \nu}{(1-(1-\nu)^2)|Z_i|^2}.$$

Hence,

$$\begin{aligned} \hat{\beta}^{(m)}_{ms-boost,i} &= (\mathbf{X}^T \mathcal{B}_m \mathbf{Y})_i = (\mathbf{X}^T \mathbf{X} D_m \mathbf{X}^T \mathbf{Y})_i = (D_m Z)_i = (1-(1-\nu)^{m_i}) Z_i \\ &\approx Z_i - \frac{\gamma_n \nu Z_i}{(1-(1-\nu)^2)|Z_i|^2} = Z_i - sign(Z_i) \frac{\gamma_n}{2-\nu} \frac{1}{|Z_i|}. \end{aligned}$$

The right-hand side is the nonnegative garrote estimator as in (12) with threshold $\gamma_n/(2-\nu)$.

Dealing with the approximation "$\approx$" can be done similarly as in the proof of Theorem 2. We define here

$$V_i = V(Z_i) = \frac{\gamma_n \nu}{(1-(1-\nu)^2)|Z_i|^2}.$$

We then define $\Delta(\nu, V_i)$ and $e_i(\nu)$ as in the proof of Theorem 2, and we complete the proof as for Theorem 2. $\square$

Acknowledgments

B. Yu would like to acknowledge gratefully the partial supports from NSF grants FD01-12731 and CCR-0106656 and ARO grants DAAD19-01-1-0643 and W911NF-05-1-0104, and the Miller Research Professorship in Spring 2004 from the Miller Institute at University of California at Berkeley. Both authors thank David Mease, Leo Breiman, two anonymous referees and the action editors for their helpful comments and discussions on the paper.

References

H. Akaike. Statistical predictor identification. *Ann. Inst. Statist. Math.*, 22:203, 1970.

L. Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37:373–384, 1995.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Arcing classifiers (with discussion). *Ann. Statist.*, 26:801–849, 1998.

L. Breiman. Prediction games & arcing algorithms. *Neural Computation*, 11:1493–1517, 1999.

P. Bühlmann. Boosting for high-dimensional linear models. *To appear in Ann. Statist.*, 34, 2006.

P. Bühlmann and B. Yu. Boosting with the $l_2$loss: regression and classification. *J. Amer. Statist. Assoc.*, 98:324–339, 2003.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression (with discussion). *Ann. Statist.*, 32:407–451, 2004.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proc. Thirteenth Intern. Conf.*, pages 148–156. Morgan Kauffman, 1996.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Ann.Statist.*, 29: 1189–1232, 2001.

J. H. Friedman. Multivariate adaptive regression splines (with discussion). *Ann.Statist.*, 19:1–141, 1991.

J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion). *Ann. Statist.*, 28:337–407, 2000.

P. J. Green and B. W. Silverman. *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman and Hall, 1994.

M. Hansen and B. Yu. Model selection and minimum description length principle. *J. Amer. Statist. Assoc.*, 96:746–774, 2001.

M. Hansen and B. Yu. *Minimum Description Length Model Selection Criteria for Generalized Linear Models*. IMS Lecture Notes – Monograph Series, Vol. 40, 2002.

M. Hansen and B. Yu. Bridging aic and bic: an mdl model selection criterion. In *IEEE Information Theory Workshop on Detection, Imaging and Estimation; Santa Fe*, 1999.

G. Lugosi and N. Vayatis. On the bayes-risk consistency of regularized boosting methods (with discussion). *Ann. Statist.*, 32:30–55 (disc. pp. 85–134), 2004.

S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Proc.*, 41:3397–3415, 1993.

N. Meinshausen. Lasso with relaxation. Technical report, 2005.

G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, 42:287–320., 2001.

G. Rätsch, A. Demiriz, and K. Bennett. Sparse regression ensembles in infinite and finite hypothesis spaces. *Machine Learning*, 48:193–221, 2002.

T. Speed and B. Yu. Model selection and prediction: normal regression. *Ann. Inst. Statist. Math.*, 45:35–54, 1993.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc., Ser. B*, 58: 267–288, 1996.

J. W. Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. Olson, J. Marks, and J. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proc. Nat. Acad. Sci. (USA)*, 98:11462–11467, 2001.