Sparse Boosting

Peter Bühlmann *
ETH Zürich

Bin Yu
University of California, Berkeley

August 2005

Abstract

We propose Sparse Boosting (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, e.g. 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: Boosting, Final Prediction Error (FPE), gMDL, Lasso, Model Selection, Nonnegative garotte, Nonparametric methods, Regression

Heading: Sparse Boosting

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

*Corresponding author. Address: Seminar für Statistik, ETH Zürich, CH-8092 Zürich, Switzerland; Email: buhlmann@stat.math.ethz.ch

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 version 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, cf. 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 (cf. 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 (cf. Freund and Schapire (1996), Breiman (1998), Friedman et al. (2000), Friedman (2001)), componentwise smoothing splines which involve selection of the best single predictor variable (cf. Bühlmann and Yu (2003)) or componentwise linear least squares in linear models with selection of the best single predictor variable (cf. Mallat and Zhang (1993), Bühlmann (2004)).

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 (cf. Mallat and Zhang, 1993; Bühlmann, 2004)
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 residual sum of squares most when using least squares fitting:

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

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 (cf. 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)

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

$\hat{g}_j(\cdot)$ 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, cf. 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 (cf. 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:

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

$\hat{g}_{j,k}(\cdot,\cdot)$ 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 3.3.

In all the examples above, the selector is given by

$$\hat{\mathcal{S}} = \operatorname*{arg\,min}_{\mathcal{S}\in\Gamma} \sum_{i=1}^{n} (U_i - (\mathcal{H}_{\mathcal{S}}\mathbf{U})_i)^2 \tag{3}$$

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 implementing 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, cf. 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, i.e. 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 (cf. Bühlmann and Yu, 2003)

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

where $\hat{\mathcal{S}}_m$ denotes the selector in iteration m. Moreover, if all the $\mathcal{H}_{\mathcal{S}}$ are linear (i.e. 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 (e.g. 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 (cf. Green and Silverman, 1994), and is used in Bühlmann (2004). 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

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

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

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

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

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

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 residual sum of squares, the selected $\tilde{\mathcal{S}}_m$ in

SparseL_2Boost minimizes a model selection criterion over all possible selectors. In particular, this means that SparseL_2Boost can not be represented anymore as a linear combination of base procedures 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 $\mathcal{B}_m(\mathcal{S})$. With a slight abuse of terminology, we still use wordings such as "SparseL_2Boost with componentwise linear least squares", meaning that the selector $\hat{\mathcal{S}}$ in the componentwise linear lest squares procedure as described in section 2.1 has to be replaced by $\tilde{\mathcal{S}}$.

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,

$$\begin{aligned} &Y_i = \sum_{j=1}^n \beta_j x_i^{(j)} + \varepsilon_i, \ i = 1, \ldots, n, \\ &\sum_{i=1}^n x_i^{(j)} x_i^{(k)} = \delta_{jk}, \end{aligned} \tag{8}$$

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

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

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 $\mathbf{X}$ (cf. (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 OLS 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

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

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,

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

We show in Figure 1 the nonnegative garrote threshold function in comparison to hard- and soft-thresholding. Hard-thresholding, corresponding to subset selection, is using ordinary least squares if $|Z_j|$ is larger than the threshold while the nonnegative garrote shrinks the OLS estimator a bit and even more shrinking for soft-thresholding, corresponding to the Lasso (Tibshirani, 1996). Therefore, for the same amount of "complexity" or "degrees of freedom" (which is in case of hard-thresholding the number of ordinary least

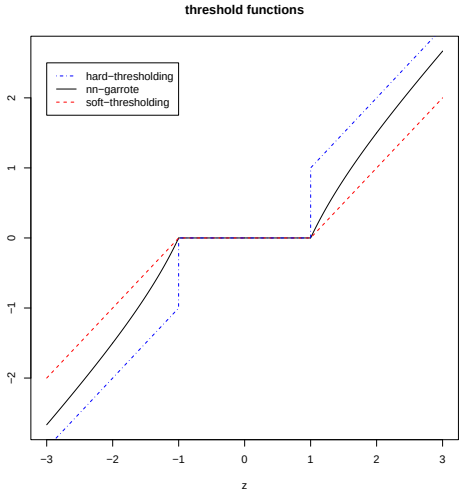

Figure 1: Threshold functions for subset selection or hard-thresholding (dashed-dotted line), nonnegative garrote (solid line) and lasso or soft-thresholding (dashed line).

squares estimated variables), hard-thresholding or subset selection will typically select the fewest number of variables (non-zero coefficient estimates) while the nonnegative garrote will include some 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 selected variable. This observation can also be compared with some numerical results in Table 2.

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

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 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 our 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 (cf. Speed and Yu (1993) and the references therein). We employ here instead a minimum description length criterion, gMDL, (cf. Hansen and Yu, 2001) which bridges the AIC and BIC criteria by having a data-dependent penalty. 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 $C(\cdot,\cdot)$). This criterion measures the code length needed based on a mixture code to transmit the response vector based on model M_k which balances the fit and a data-driven complexity of the model. See Hansen and Yu (2001) for details on its derivation through the coding interpretation.

In the SparseL_2Boost algorithm in section 2.3.1, if we take

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

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

$$\hat{m} = \operatorname*{arg\,min}_{1\le m\le M} C_{gMDL}(RSS_m, \mathrm{trace}(\mathcal{B}_m)), \tag{15}$$

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 one real data set, 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

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

where ε is independent from X. The sample size is chosen as $n = 50$ and the predictor-dimension is $p \in \{50, 100\}$. 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),

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)
(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)

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 $n = 50$.

SparseL_2Boost outperforms L_2Boosting. Furthermore, both methods are essentially insensitive when increasing the number of ineffective variables, i.e. from the case $p = 50$ to $p = 100$ (the decrease from $pp = 50$ to $p = 100$ in SparseL_2Boost with Σ as in (17) is non-significant). And finally, the gMDL rule for the stopping iteration $\hat{m}$ works very well in comparison to the oracle performance.

Next, we consider the ability of selecting the correct variables: the results are given

Σ , dim.		SparseL_2Boost	L_2Boosting
(17), $p = 50$:	ℓ^0-norm	5	13.68
	non-selected T	0	0
	selected F	1	9.68
(17), $p = 100$:	ℓ^0-norm	5.78	21.2
	non-selected T	0	0
	selected F	1.78	17.2
(18), $p = 50$:	ℓ^0-norm	4.98	9.12
	non-selected T	0	0
	selected F	0.98	5.12
(18), $p = 100$:	ℓ^0-norm	5.5	12.44
	non-selected T	0	0
	selected F	1.5	8.44

Table 2: 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). The methods using the oracle m which minimize MSE are denoted by an asterisk *. Model (16), sample size is $n = 50$.

in Table 2. In the orthogonal case, we have seen through explicit expressions that SparseL_2Boost gives sparser results than L_2Boosting. This is confirmed in our ℓ_0-sparse model (16), because Table 2 shows that SparseL_2Boost indeed selects much fewer predictors than L_2Boosting. Moreover, for this model, SparseL_2Boost is a good model selection estimator while L_2Boosting is much worse, if one takes a look at the numbers of the non-selected true predictors or the selected false predictors in the table.

3.1.2 A non-sparse model with respect to the ℓ^0-norm

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 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. The squared error performance is given in Table 3, supporting our expectations.

SparseL_2Boost	L_2Boosting	SparseL_2Boost*	L_2Boosting*
3.64 (0.188)	2.19 (0.083)	3.61 (0.189)	2.08 (0.078)

Table 3: Mean squared error (MSE) in model (19) with $p = 50$. 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.

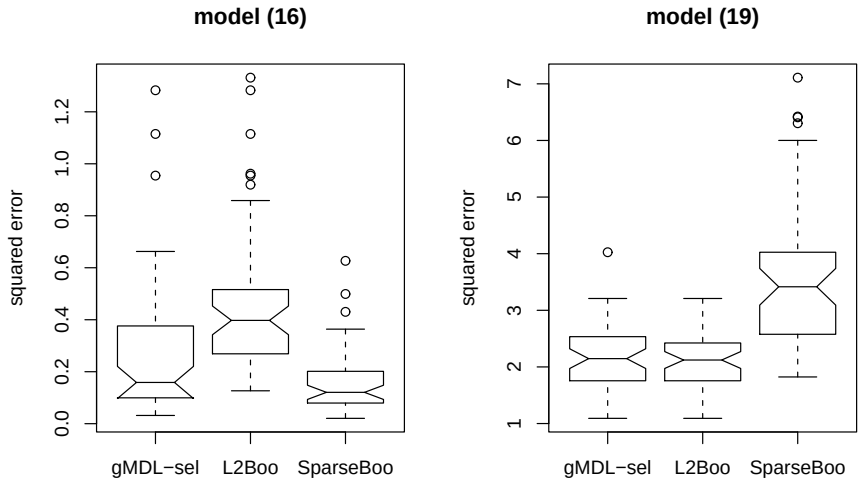

Figure 2: Out-of-sample squared error losses, $\underset{X}{ave}[(\hat{f}(X) - f(X))^2]$ ($f(x) = \mathbb{E}[Y|X = x]$), from the 50 simulations for the models in (16) and (19) with $p = 50$. gMDL-sel-L_2Boost (gMDL-sel), L_2Boosting (L2Boo) and SparseL_2Boost (SparseBoo). Sample size is $n = 50$.

gMDL-sel-L_2Boost 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 datasets). 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 also 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:

	SparseL_2Boost	L_2Boosting
10-fold CV score	16.52	16.57

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 dataset and choose the method according to the better gMDL-score, i.e. gMDL-sel-L_2Boost (see section 2.5). Some results are given in Table 4. While SparseL_2Boost is about as good as L_2Boosting in terms of predictive accuracy, as mentioned above, it yields a sparser model fit (10 terms instead of 18 terms by L_2boost). Based on SparseL_2Boost, an estimate for the error variance is $n^{-1}\sum_{i=1}^n (Y_i - \hat{Y}_i)^2 = 15.56$ and the goodness of fit equals $R^2 = \sum_{i=1}^n (\hat{Y}_i - \overline{Y})^2 / \sum_{i=1}^n (Y_i - \overline{Y}_i)^2 = 0.71$.

	gMDL-score	RSS	ℓ^0-norm (number of terms)
SparseL_2Boost (#)	2.853	15.56	10
L_2Boost	2.862	15.24	18

Table 4: gMDL-score, $n^{-1}\times$ residual sum of squares (RSS) and number of selected terms (ℓ^0-norm) for the ozone dataset with 45 predictor variables consisting of intercept, main effects and second-order interaction terms. (#) gMDL-sel-L_2Boost selects SparseL_2Boost as the better method.

3.3 Nonparametric function estimation with second-order interactions

Consider the Friedman #1 model (Friedman, 1991),

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

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 `d.f.` $= 5$ for every thin plate spline), yields a second-order interaction model; see also section 2.1. We demonstrate in Table 5 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

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 5: Mean squared error (MSE) in model (20). All other specifications are described in the caption of Table 1, except for MARS which is constrained to second-order interaction terms.

estimation of the boosting iterations by gMDL did not do as well as in section 3.1 since the oracle methods perform significantly better. Both boosting methods nevertheless are quite a bit better than MARS.

When increasing the noise level, using $\mathrm{Var}(\varepsilon) = 16$, we obtain the following MSEs for $p = 10$: 11.70 for SparseL_2Boost and 24.11 for MARS. Thus, for lower signal to noise ratios, our SparseL_2Boost algorithm is much better than MARS.

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, e.g. having linear complexity in the number of predictor variables p when using componentwise linear least squares or componentwise smoothing splines (cf. 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. 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 $\mathcal{H}_{\hat{\mathcal{S}}}$ 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

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

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 monotonously 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

$$\begin{aligned} &RSS_k - RSS_{k+1} > \delta^2,\ k = 1, 2, ..., m-1, \\ &RSS_m - RSS_{m+1} \leq \delta^2, \end{aligned} \tag{21}$$

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 (21) implies that for all $i \in \{1, \ldots, n\}$

$$\begin{aligned} &((1-\nu)^{2(m_i-1)} - (1-\nu)^{2m_i})Z_i^2 > \delta^2, \\ &((1-\nu)^{2m_i} - (1-\nu)^{2(m_i+1)})Z_i^2 \leq \delta^2. \end{aligned} \tag{22}$$

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 (22).

Let us first, for the sake of insight, replace the "$\leq$" in (22) by "$\approx$": we will deal later in which sense such an approximate equality holds. If $m_i \geq 1$, 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

$$(1-\nu)^{m_i} \approx \frac{\delta}{\sqrt{1 - (1-\nu)^2}|Z_i|}. \tag{23}$$

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

$$\hat{\beta}^{(m)}_{Boost,i} \approx \begin{cases} Z_i - \lambda, & \text{if } Z_i \geq \lambda, \\ 0, & \text{if } |Z_i| < \lambda, \\ Z_i + \lambda, & \text{if } Z_i \leq -\lambda. \end{cases} \tag{24}$$

where $\lambda = \frac{\delta}{\sqrt{1-(1-\nu)^2}}$ (note that m is connected to δ, and hence to λ via the criterion in (21)). 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 (23). 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 (24). 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 (22), the derivation as for (23) and the choice of δ, this says that

$$\begin{aligned} (1-\nu)^{m_i} &> V_i, \\ (1-\nu)^{m_i+1} &\leq V_i, \end{aligned} \tag{25}$$

and hence

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

by using (25). Thus,

$$\begin{aligned} &(1-\nu)^{m_i} = V_i + ((1-\nu)^{m_i} - V_i) = V_i(1 + \Delta(\nu, V_i)/V_i) = V_i(1+e_i(\nu)), \\ &|e_i(\nu)| = |\Delta(\nu, V_i)/V_i| \leq \nu/(1-\nu). \end{aligned} \tag{26}$$

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 (26). □

Proof of Theorem 1. The proof is based on similar ideas as for Theorem 2. The MS-L_2Boosting in iteration m aims to minimize

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

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

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

where $\text{trace}(\mathcal{B}_m) = \sum_{i=1}^n (1-(1-\nu)^{m_i})$. Moreover, we run MS-$L_2$Boosting until the stopping iteration m satisfies the following:

$$\begin{aligned} &MSB_k - MSB_{k+1} > 0, \ k = 1, 2, \ldots, m-1, \\ &MSB_m - MSB_{m+1} \le 0. \end{aligned} \tag{27}$$

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 MS-L_2Boosting 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,

$$\begin{aligned} &(1-\nu)^{2(m_i-1)} Z_i^2 (1-(1-\nu)^2) - \gamma_n \nu (1-\nu)^{m_i-1} > 0, \\ &(1-\nu)^{2m_i} Z_i^2 (1-(1-\nu)^2) - \gamma_n \nu (1-\nu)^{m_i} \le 0. \end{aligned} \tag{28}$$

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 (27) holds or of the m_i such that (28) holds.

Similarly to the proof of Theorem 2, we look at this expression first in terms of an approximate equality to zero, i.e. ≈ 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 "≈" 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. □

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

References

[1] Akaike, H. (1970). Statistical predictor identification. Ann. Inst. Statist. Math. **22** 202–217.

[2] Breiman, L. (1995). Better subset regression using the nonnegative garrote. Technometrics **37**, 373–384.

[3] Breiman, L. (1996). Bagging predictors. Machine Learning **24**, 123–140.

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

[5] Breiman, L. (1999). Prediction games & arcing algorithms. Neural Computation **11**, 1493-1517.

[6] Bühlmann, P. (2004). Boosting for high-dimensional linear models. To appear in Ann. Statist.

[7] Bühlmann, P. and Yu, B. (2003). Boosting with the L_2loss: regression and classification. J. Amer. Statist. Assoc. **98**, 324–339.

[8] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression. Ann. Statist. **32**, 407–499 (with discussion).

[9] Freund, Y. and Schapire, R.E. (1996). Experiments with a new boosting algorithm. In Machine Learning: Proc. Thirteenth International Conference, pp. 148–156. Morgan Kauffman, San Francisco.

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

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

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

[13] Green, P.J. and Silverman, B.W. (1994). *Nonparametric Regression and Generalized Linear Models.* Chapman and Hall.

[14] Hansen, M. and Yu, B. (2001). Model selection and minimum description length principle. J. Amer. Statist. Assoc. **96**, 746–774.

[15] Lugosi, G. and Vayatis, N. (2004). On the Bayes-risk consistency of regularized boosting methods. Ann. Statist. **32**, 30–55 (disc. pp. 85–134).

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

[17] Meinshausen, N. (2005). Lasso with relaxation. Preprint.

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

[19] Rätsch, G., Onoda, T. and Müller, K.-R. (2001). Soft margins for AdaBoost. Machine Learning **42**, 287–320.

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

[21] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. J. Roy. Statist. Soc., Ser. B, **58**, 267–288.

[22] Tukey, J.W. (1977). *Exploratory data analysis.* Addison-Wesley, Reading, MA.