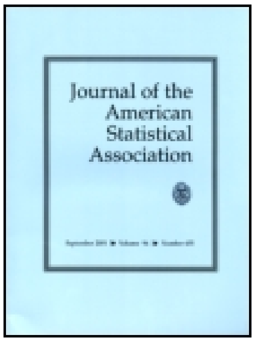

Journal of the American Statistical Association

Journal of the American Statistical Association

ISSN: 0162-1459 (Print) 1537-274X (Online) Journal homepage: https://www.tandfonline.com/loi/uasa20

Boosting With the L_2 Loss

Regression and Classification

Peter Bühlmann & Bin Yu

To cite this article: Peter Bühlmann & Bin Yu (2003) Boosting With the L_2 Loss, Journal of the American Statistical Association, 98:462, 324-339, DOI: 10.1198/016214503000125

To link to this article: https://doi.org/10.1198/016214503000125

Published online: 31 Dec 2011.

Boosting With the L_2 Loss: Regression and Classification

Peter BÜHLMANN and Bin YU

This article investigates a computationally simple variant of boosting, L_2Boost, which is constructed from a functional gradient descent algorithm with the L_2-loss function. Like other boosting algorithms, L_2Boost uses many times in an iterative fashion a prechosen fitting method, called the learner. Based on the explicit expression of refitting of residuals of L_2Boost, the case with (symmetric) linear learners is studied in detail in both regression and classification. In particular, with the boosting iteration m working as the smoothing or regularization parameter, a new exponential bias–variance trade-off is found with the variance (complexity) term increasing very slowly as m tends to infinity. When the learner is a smoothing spline, an optimal rate of convergence result holds for both regression and classification and the boosted smoothing spline even adapts to higher-order, unknown smoothness. Moreover, a simple expansion of a (smoothed) 0–1 loss function is derived to reveal the importance of the decision boundary, bias reduction, and impossibility of an additive bias–variance decomposition in classification. Finally, simulation and real dataset results are obtained to demonstrate the attractiveness of L_2Boost. In particular, we demonstrate that L_2Boosting with a novel component-wise cubic smoothing spline is both practical and effective in the presence of high-dimensional predictors.

KEY WORDS: Functional gradient descent; LogitBoost; Minimax error rate; Nonparametric classification; Nonparametric regression; Smoothing spline.

1. INTRODUCTION

Boosting is one of the most successful and practical methods that has recently come from the machine learning community. Since its inception in 1990 (Schapire 1990; Freund 1995; Freund and Schapire 1996), it has been tried on an amazing array of datasets. The improved performance through boosting of a fitting method, called the *learner*, has been impressive and seems to be associated with boosting's resistance to overfitting. The burning question is why this is so.

The rationale behind boosting diverges from that of the traditional statistical procedures. It starts with a sensible estimator or classifier, the learner, and seeks its improvements iteratively based on its performance on the training dataset. The possibility of this boosting procedure comes with the availability of large datasets where one can easily set aside part of it as the test set (or use cross-validation based on random splits). It seemingly bypasses the need to get a model for the data and the pursuit of the optimal solution under this model as the common practice in traditional statistics. For large dataset problems with high-dimensional predictors, a good model for the problem is hard to come by, but a sensible procedure is not. This may explain the empirical success of boosting on large, high-dimensional datasets. After much work on bounding the test set error (generalization error) of a boosted procedure via the Vapnik–Červonenkis (VC) dimensions and the distribution of so-called margins (Schapire, Freund, Bartlett, and Lee 1998), some recent developments on boosting have been on the gradient descent view of boosting. These are the results of efforts of many researchers (e.g., Breiman 1999; Mason, Baxter, Bartlett, and Frean 2000; Friedman, Hastie, and Tibshirani 2000; Collins, Schapire, and Singer 2000). This gradient descent view connects boosting to the more common optimization view of statistical inference; its most obvious consequence has been the emergence of many variants of the original AdaBoost, under various loss or objective functions (Mason et al. 2000; Friedman et al. 2000; Friedman 2001). Even though a satisfactory explanation of why boosting works does not follow directly from this gradient descent view, some of the new boosting variants are more easily accessible for analysis. In this article we take advantage of this new analytic possibility on L_2 boosting procedures to build our case for understanding boosting in both regression and classification. It is worth pointing out that L_2Boost is studied here as a procedure yielding competitive statistical results in regression and classification, in addition to its computational simplicity and analytical tractability.

After a brief overview of boosting from the gradient descent point of view in Section 2, the case of (symmetric) linear learners in regression is covered in Section 3, building on the known fact that L_2Boost is a stagewise refitting of the residuals (Friedman 2001). We derive two main rigorous results:

1. With the boosting iteration m working as the smoothing or regularization parameter, a new exponential bias–variance trade-off is found. When the iteration m increases by 1, one more term is added in the fitted procedure, but because of this new term's dependence on the previous terms, the "complexity" of the fitted procedure is increased not by a constant amount as we became used to in linear regression, but rather by an exponentially diminishing amount as m gets large. At the iteration limit, the complexity or variance term is bounded by the noise variance in the regression model.
2. When the learner is a smoothing spline, L_2Boost achieves the optimal rate of convergence for one-dimensional function estimation. Moreover, this boosted smoothing spline adapts to higher-order, *unknown* smoothness.

Result 1 partially explains the "overfitting resistance" mystery of boosting. This phenomenon is radically different from the well-known algebraic bias–variance trade-off in nonparametric regression. Result 2 gives an interesting result about boosting in adaptive estimation: Even when smoothness is unknown, L_2Boost achieves the optimal (minimax) rate of convergence.

Peter Bühlmann is Associate Professor, Seminar für Statistik, ETH Zürich, CH-8032 Zürich, Switzerland (E-mail: *buhlmann@stat.math.ethz.ch*). Bin Yu is Professor, Department of Statistics, University of California, Berkeley, CA 94720 (E-mail: *binyu@stat.berkeley.edu*). The authors thank Trevor Hastie and Leo Breiman for very helpful discussions and two referees for constructive comments. Yu was partially supported by National Science Foundation grants DMS-9803063 and FD01-12731 and Army Research Office grants DAAG55-98-1-0341 and DAAD19-01-1-0643.

Journal of the American Statistical Association
June 2003, Vol. 98, No. 462, Theory and Methods
DOI 10.1198/016214503000125

Section 4 proposes L_2Boost with a novel componentwise smoothing spline learner as a very effective procedure for carrying out boosting for high-dimensional regression problems with continuous predictors. This proposed L_2Boost is shown to outperform L_2Boost with stumps (i.e., a tree with two terminal nodes) and other more traditional competitors, particularly when the predictor space is very high dimensional.

Section 5 deals with classification, first with the two-class problem and then with the multiclass problem using the "one against all" approach. The optimality in result 2 also holds for classification: L_2Boost achieves the optimal (minimax) rate of convergence to the Bayes risk over an appropriate smoothness function class, the risk of the best among all classification procedures. Section 6 also approximates the 0–1 loss function via a smoothed version to show that the test set (generalization) error of any procedure is approximately (in addition to the Bayes risk) a sum of tapered moments. As a consequence of this approximation, we gain more insight into why bias plays a bigger role in 0–1 loss classification than in L_2 regression, why there is even more "resistance against overfitting" in classification than in regression, and why previous attempts were not successful at decomposing the test set (generalization) error into additive bias and variance terms (e.g., Geman, Bienenstock, and Doursat 1992; Breiman 1998).

Section 7 provides support for the theory and explanations for classification by simulated and real datasets that demonstrate the attractiveness of L_2Boost. Finally, Section 8 contains a discussion on the role of the learner and a summary of the article.

2. BOOSTING: STAGEWISE FUNCTIONAL GRADIENT DESCENT

The boosting algorithms can be seen as functional gradient descent techniques. The task is to estimate the function $F : \mathbb{R}^d \to \mathbb{R}$, minimizing an expected cost

$$\mathbb{E}[C(Y, F(X))], \qquad C(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+ \tag{1}$$

based on data (Y_i, X_i) $(i = 1, \dots, n)$. Here we consider cases where the univariate response Y is both continuous (regression problem) and discrete (classification problem), because boosting is potentially useful in both cases. Here X denotes a d-dimensional predictor variable. The cost function $C(\cdot,\cdot)$ is assumed to be smooth and convex in the second argument, to ensure that the gradient method works well. The most prominent examples are

$$C(y, f) = \exp(yf) \text{ with } y \in \{-1, 1\}\text{: AdaBoost cost function,}$$

$$C(y, f) = \log_2(1 + \exp(-2yf)) \text{ with } y \in \{-1, 1\}\text{: LogitBoost cost function,} \tag{2}$$

$$C(y, f) = (y - f)^2/2 \text{ with } y \in \mathbb{R} \text{ or } \in \{-1, 1\}\text{: } L_2\text{Boost cost function.}$$

The population minimizers of (1) are then (Friedman et al. 2000)

$$F(x) = \frac{1}{2}\log\left(\frac{\mathbb{P}[Y = 1|X = x]}{\mathbb{P}[Y = -1|X = x]}\right) \quad \text{for AdaBoost and LogitBoost cost,} \tag{3}$$

$$F(x) = \mathbb{E}[Y|X = x] \quad \text{for } L_2\text{Boost cost.}$$

Estimation of such an $F(\cdot)$ from data can be done via a constrained minimization of the empirical risk,

$$n^{-1}\sum_{i=1}^{n} C(Y_i, F(X_i)), \tag{4}$$

applying functional gradient descent. This gradient descent view has been recognized and refined by various authors, including Breiman (1999), Mason et al. (2000), Friedman et al. (2000), and Friedman (2001). In summary, the minimizer of (4) is imposed to satisfy a "smoothness" (or "regularization") constraint in terms of an additive expansion of ("simple") learners (fitted functions),

$$h(x, \hat{\theta}), \qquad x \in \mathbb{R}^d,$$

where $\hat{\theta}$ is an estimated finite or infinite-dimensional parameter. For example, the learner $h(\cdot, \hat{\theta})$ could be a decision tree where $\hat{\theta}$ describes the axis to be split, the split points and the fitted values for every terminal node (the constants in the piecewise constant fitted function). How to fit $h(x, \theta)$ from data is part of the learner and can be done according to a basis algorithm. For example, least squares fitting yields

$$\hat{\theta}_{U,X} = \arg\min_{\theta}\sum_{i=1}^{n}(U_i - h(X_i; \theta))^2$$

for some data $(U, X) = \{(U_i, X_i); i = 1, \dots, n\}$. The general description of functional gradient descent follows (see also Friedman 2001).

Generic functional gradient descent

Step 1 (initialization). Given data $\{(Y_i, X_i);\ i = 1, \dots, n\}$, fit a real-valued, (initial) learner,

$$\widehat{F}_0(x) = h(x; \hat{\theta}_{Y,X}).$$

When using least squares, $\hat{\theta}_{Y,X} = \arg\min_{\theta}\sum_{i=1}^{n}(Y_i - h(X_i; \theta))^2$. Set $m = 0$.

Step 2 (projection of gradient to learner). Compute the negative gradient vector,

$$U_i = -\left.\frac{\partial C(Y_i, F)}{\partial F}\right|_{F=\widehat{F}_m(X_i)}, \qquad i = 1, \dots, n,$$

evaluated at the current $\widehat{F}_m(\cdot)$. Then fit the real-valued learner to the gradient vector,

$$\hat{f}_{m+1}(x) = h(x, \hat{\theta}_{U,X}).$$

When using least squares, $\hat{\theta}_{U,X} = \arg\min_{\theta}\sum_{i=1}^{n}(U_i - h(X_i; \theta))^2$.

Step 3 (line search). Do one-dimensional numerical search for the best step-size,

$$\hat{w}_{m+1} = \arg\min_{w}\sum_{i=1}^{n} C(Y_i, \widehat{F}_m(X_i) + w_{m+1}\hat{f}_{m+1}(X_i)).$$

Update

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

Step 4 (iteration). Increase m by 1, and repeat steps 2 and 3.

The learner $h(x, \hat{\theta}_{U,X})$ in step 2 can often be viewed as an estimate of $\mathbb{E}[U_i|X = x]$, and it takes values in $\mathbb{R}$, even in case of a classification problem with Y_i in a finite set. We call $\widehat{F}_m(\cdot)$ the AdaBoost, LogitBoost, or L_2Boost estimate, according to the implementing cost function in (2).

L_2Boost has a simple structure: The negative gradient in Step 2 is the classical residual vector, and the line search in Step 3 is trivial.

L_2Boost algorithm

Step 1 (initialization). Follow step 1 of the generic functional gradient descent, using a least squares fit (maybe including some regularization).
Step 2. Compute residuals $U_i = Y_i - \widehat{F}_m(X_i)$ $(i = 1, \ldots, n)$ and fit the real-valued learner to the current residuals by (regularized) least squares as in step 2 of the generic functional gradient descent; the fit is denoted by $\hat{f}_{m+1}(\cdot)$.
Update

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

Step 3 (iteration). Increase iteration index m by 1, and repeat step 2.

L_2 boosting is thus nothing else than repeated least squares fitting of residuals (Friedman 2001). With $m = 1$ (one boosting step), it has already been proposed by Tukey (1977) under the name "twicing."

For a continuous $Y \in \mathbb{R}$, a regression estimate for $\mathbb{E}[Y|X = x]$ is directly given by the L_2Boost estimate $\widehat{F}_m(\cdot)$. For a two-class problem with $Y \in \{-1, 1\}$, a classifier under equal misclassification costs is given by

$$sign(\widehat{F}_m(x)), \tag{5}$$

because $\mathbb{E}[Y|X = x] = \mathbb{P}[Y = 1|X = x] - \mathbb{P}[Y = -1|X = x]$. AdaBoost and LogitBoost estimates aim to estimate

$$F(x) = \frac{1}{2}\log\left(\frac{\mathbb{P}[Y = 1|X = x]}{\mathbb{P}[Y = -1|X = x]}\right).$$

Hence an appropriate classifier is again given by (5).

Mason et al. (2000) and Collins et al. (2000) have described when boosting-type algorithms (i.e., functional gradient descent) converge numerically. This tells us that under certain conditions, the test set (generalization) error for boosting eventually stabilizes. But it does not imply that the eventually stable solution is the best, or that overfitting could happen long before reaching convergence. Indeed, we show in Section 3 that L_2Boost with "contracting" linear learners converges to the fully saturated model, that is, $\widehat{F}_\infty(X_i) = Y_i$ for all $i = 1, \ldots, n$, fitting the data exactly.

Obviously, L_2Boost and other functional gradient descent methods depend on the choice of the learner. As the boosting iteration runs, the boosted procedure has more terms and hence becomes more complex. It is intuitively clear that L_2 boosting is not worthwhile if the learner is already complex (e.g., fitting many parameters), so that every boosting iteration contributes to additional overfitting. We make this rigorous in Section 3 for linear learners. Thus the learner in boosting should be "simple"; it typically involves only few parameters and has low variance relative to bias. We say that such a learner is "weak," an informal terminology from machine learning. Weakness of a learner does depend on the signal-to-noise ratio of the data; it is well known that if the noise level is low, a statistical method has less a tendency to overfit. Of course, there are many possibilities for choosing a weak learner; examples include stumps that are using trees with two terminal nodes only, using smoothing methods with large amount of smoothing, or shrinking a learner with a small shrinkage factor. Illustrations of these are given in Sections 3.2.2, 4.2, 7, and 8.

3. L_2 BOOSTING WITH LINEAR LEARNERS IN REGRESSION

The nature of stagewise fitting is responsible to a large extent for boosting's resistance to overfitting. The same view has been expressed in Buja's (2000) discussion of the article of Friedman et al. (2000). He made amply clear there that this stagewise fitting had gotten a bad reputation among statisticians and was not getting the attention that it deserved. The success of boosting definitely serves as an eye-opener for us to take a fresh look at stagewise fitting.

3.1 Theory

Consider the regression model

$$Y_i = f(x_i) + \varepsilon_i, \quad i = 1, \ldots, n,$$
$$\varepsilon_1, \ldots, \varepsilon_n \text{ iid with } \mathbb{E}[\varepsilon_i] = 0,\ \mathrm{var}(\varepsilon_i) = \sigma^2, \tag{6}$$

where $f(\cdot)$ is a real-valued, typically nonlinear function and the predictors $x_i \in \mathbb{R}^d$ are deterministic (e.g., conditioning on the design).

We can represent a learner, evaluated at the predictors $x_1, \ldots, x_n$ as an operator $\mathcal{S} : \mathbb{R}^n \to \mathbb{R}^n$, mapping the responses $Y_1, \ldots, Y_n$ to some fitted values in $\mathbb{R}^n$. The predictors $x_1, \ldots, x_n$ are absorbed in the operator notation $\mathcal{S}$. In the sequel we often use the notation Y for the vector $(Y_1, \ldots, Y_n)^T$ and $\widehat{F}_j$ for the vector $(\widehat{F}_j(x_1), \ldots, \widehat{F}_j(x_n))^T$ and analogously for $\hat{f}_j$; it should always be clear from the context whether we mean a single variable Y or function $\widehat{F}_j(\cdot)$, or the vectors as earlier.

Proposition 1. The L_2Boost estimate in iteration m can be represented as

$$\widehat{F}_m = \sum_{j=0}^{m} \mathcal{S}(I - \mathcal{S})^j Y = (I - (I - \mathcal{S})^{m+1})Y.$$

A proof is given in the Appendix. We define the boosting operator $\mathcal{B}_m : \mathbb{R}^n \to \mathbb{R}^n$ as

$$\mathcal{B}_m = I - (I - \mathcal{S})^{m+1}$$

so that $\mathcal{B}_m Y = \widehat{F}_m$.

Proposition 1 describes an explicit relationship between the boosting operator and the learner $\mathcal{S}$. We exploit this in the sequel.

We focus now on linear learners $\mathcal{S}$. Examples include least squares fitting in linear models, more general projectors to a given class of basis functions such as regression splines, and smoothing operators such as kernel and smoothing spline estimators.

Proposition 2. Consider a linear learner $\mathcal{S}$ with eigenvalues $\{\lambda_k;\ k = 1, \ldots, n\}$, based on deterministic predictors $x_1, \ldots, x_n$. Then the eigenvalues of the L_2Boost operator $\mathcal{B}_m$ are $\{(1 - (1 - \lambda_k)^{m+1};\ k = 1, \ldots, n\}$.

Proof. This is a direct consequence of Proposition 1.

Our analysis becomes even more transparent when specialized to the case in which (the matrix) $\mathcal{S} = \mathcal{S}^T$ is symmetric. An important example is the smoothing spline operator (see Wahba 1990; Hastie and Tibshirani 1990), which is a more data-adaptive smoothing technique than, say, a kernel with a global bandwidth. All eigenvalues of $\mathcal{S}$ are then real, and $\mathcal{S}$ as well as $\mathcal{B}_m$ can be diagonalized with an orthonormal transform,

$$\mathcal{B}_m = U D_m U^T, \quad D_m = \operatorname{diag}(1-(1-\lambda_k)^{m+1}),$$
with the kth column vector of U being the kth eigenvector of $\mathcal{S}$ to the eigenvalue λ_k. (7)

The matrix U is orthonormal, satisfying $UU^T = U^T U = I$.

We are now able to analyze a relevant generalization measure in this setting—the (averaged) mean squared error (MSE),

$$MSE = n^{-1}\sum_{i=1}^{n}\mathbb{E}[(\widehat{F}_m(x_i) - f(x_i))^2], \tag{8}$$

which averages over the observed predictors. Note that if the design is stochastic with a probability distribution, then the foregoing MSE measure is asymptotically equivalent (as $n \to \infty$) to the prediction (generalization) error $\mathbb{E}[(\widehat{F}_m(X) - f(X))^2]$, where X is a new test observation from the design-generating distribution but independent from the training set and the expectation is over the training and test sets. Figure 1 illustrates the difference between the two measures for a finite-sample case.

Proposition 3. Consider a linear, symmetric learner $\mathcal{S} = \mathcal{S}^T$ with eigenvalues $\{\lambda_k; k = 1,\ldots,n\}$ and eigenvectors building the columns of the orthonormal matrix U. Assume data being generated from the model (6) and denote by $f = (f(x_1),\ldots,f(x_n))^T$ the vector of the true regression function $f(\cdot)$ evaluated at x_i's. Then the squared bias, variance, and averaged MSE for L_2Boost are

$$\begin{aligned}\text{bias}^2(m,\mathcal{S}; f) &= n^{-1}\sum_{i=1}^{n}(\mathbb{E}[\widehat{F}_m(x_i)] - f(x_i))^2\\ &= n^{-1} f^T U \operatorname{diag}((1-\lambda_k)^{2m+2}) U^T f,\\ \text{var}(m,\mathcal{S};\sigma^2) &= n^{-1}\sum_{i=1}^{n}\text{var}(\widehat{F}_m(x_i))\\ &= \sigma^2 n^{-1}\sum_{k=1}^{n}(1-(1-\lambda_k)^{m+1})^2,\end{aligned}$$

and

$$MSE(m,\mathcal{S}; f,\sigma^2) = \text{bias}^2(m,\mathcal{S}; f) + \text{var}(m,\mathcal{S};\sigma^2).$$

A proof is given in the Appendix. Proposition 3 describes an exact result for the MSE in terms of the chosen L_2Boost procedure. It is clear that the iteration index m acts as a "smoothing parameter" to control the bias–variance trade-off.

Given the underlying problem (i.e., f and σ^2) and given a learner $\mathcal{S}$ (implying U and the set of eigenvalues), we analyze the bias–variance trade-off as a function of boosting iterations. For that purpose, we assume that all eigenvalues satisfy $0 < \lambda_k \le 1$. An important example for such a linear learner are cubic smoothing splines that have two eigenvalues equal to 1 and all others strictly between 0 and 1; this is treated in more detail in Section 3.2.

Theorem 1. Under the assumptions in Proposition 3 with $0 < \lambda_k \le 1$, $k = 1,\ldots,n$,

a. $bias^2(m;\mathcal{S}; f)$ decays exponentially fast with increasing m, $\text{var}(m,\mathcal{S};\sigma^2)$ exhibits exponentially small increase with increasing m, and $\lim_{m\to\infty} MSE(m,\mathcal{S}; f,\sigma^2) = \sigma^2$.
b. If $\lambda_k < 1$ for at least one $k \in \{1,\ldots,n\}$ ($\mathcal{S}$ is not the identity operator I), there is an m, such that at the mth iteration, the boosting MSE is strictly lower than σ^2.
c. Moreover, let $\mu = U^T f = (\mu_1,\ldots,\mu_n)^T$ be the function vector in the linear space spanned by column vectors of U (μ represents f in the coordinate system of the eigenvectors of $\mathcal{S}$).
   If $\mu_k^2/\sigma^2 > 1/(1-\lambda_k)^2 - 1$ for all k with $\lambda_k < 1$, then boosting improves the MSE over the linear learner $\mathcal{S}$.

Assertion a is a direct consequence of Proposition 3. A proof of assertions b and c is given in the Appendix.

Theorem 1 comes as a surprise: it shows a very interesting bias–variance trade-off that has not been seen in the literature. As the boosting iteration (or smoothing parameter) m varies, both the bias and variance (complexity) term change exponentially, with the bias decreasing exponentially fast and the variance increasing with exponentially diminishing terms eventually. This contrasts the standard algebraic trade-off commonly seen in nonparametric estimation. Figure 1 illustrates the difference for a cubic smoothing spline learner for data from model (11) in Section 3.2.2. The exponential trade-off not only gets very close to the optimal of the MSE of that from the smoothing splines (by varying the smoothing parameter), but also stays really flat afterward, due to the exponential

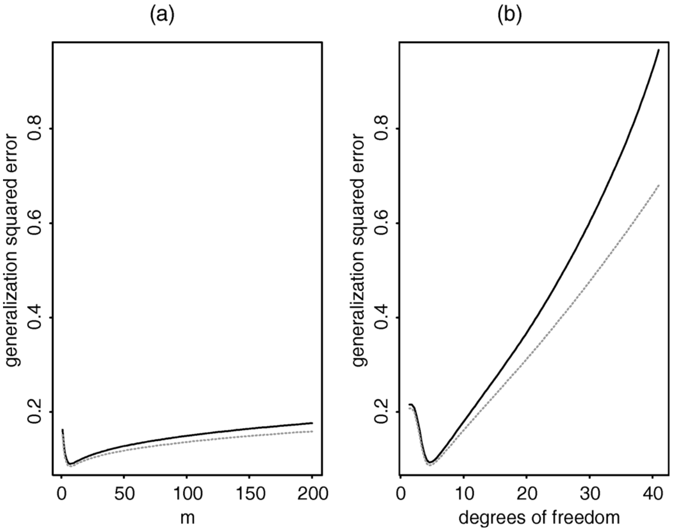

Figure 1. Mean Squared Prediction Error $\mathbb{E}[(Y - \hat{f}(X))^2]$ (——) and MSE Criterion From (8) (.....). From 100 Simulations of Model (11) With Design Uniformly Distributed on [−1/2, 1/2], Each With n = 100. (a) L_2Boost with cubic smoothing spline having df = 3, as a function of boosting iterations m. (b) Cubic smoothing spline for various degrees of freedom (various amount of smoothing).

increase and bias and variance decrease. Condition $\mu_k^2/\sigma^2 > 1/(1-\lambda_k)^2 - 1$ in part c can be interpreted as follows. A large left side, μ_k/σ^2, means that f is relatively complex compared with the noise level σ^2. A small right side, $1/(1-\lambda_k)^2 - 1$, means that λ_k is small; that is, the learner uses very strong shrinkage or smoothing in the kth eigenvector direction, or the learner is actually weak in the kth direction. Thus, for boosting to bring improvement, μ_k must be large relative to the noise level σ^2 and/or λ_k is restricted to be sufficiently small. Hence we see improvements for boosting with weak learners most of the time. The assertion in b shows that boosting always beats the unbiased estimator Y, as does the James–Stein estimator in the space spanned by U.

Boosting until theoretical convergence with $m = \infty$ is typically not a good strategy; the MSE with $m = \infty$ is not smaller than the noise level σ^2. The reason is that boosting infinitely often yields the fully saturated model that fits the data exactly. Therefore, one should monitor an estimate of MSE by, for example, using a test set or cross-validation.

Finally, boosting sometimes does not change the procedure at all.

Corollary 1. Under the assumptions in Proposition 3 with $\lambda_k \in \{0, 1\}, k = 1, \ldots, n$, ($\mathcal{S}$ is a linear projection operator), $\mathcal{B}_m \equiv \mathcal{S}$ for $m = 1, 2, \ldots$.

The phenomenon in Theorem 1 generalizes qualitatively to higher-order moments.

Theorem 2. Under the assumptions in Theorem 1 with $0 < \lambda_k \le 1, k = 1, \ldots, n$, and assuming that $\mathbb{E}|\varepsilon_1|^p < \infty$ for $p \in \mathbb{N}$,

$$n^{-1}\sum_{i=1}^{n}\mathbb{E}[(\widehat{F}_m(x_i) - f(x_i))^p] = \mathbb{E}[\varepsilon_1^p] + O(\exp(-Cm)) \qquad (m \to \infty),$$

where $C > 0$ is a constant independent of m (but depending on n and p).

A proof is given in the Appendix. We use Theorem 2 to argue that the expected 0–1 loss in classification also exhibits only an exponentially small amount of overfitting as boosting iterations $m \to \infty$.

3.2 Smoothing Splines as Learners

3.2.1 Theory. A special class of symmetric linear learners are the smoothing spline operators when the predictors are one-dimensional. Denote the function class of the νth-order smoothness ($\nu \in \mathbb{N}$), defined on an interval $[a, b]$, as

$$\mathcal{W}_2^{(\nu)} = \{f\colon\ f(\nu-1)\text{-times continuously differentiable and } \int_a^b [f^{(\nu)}(x)]^2\,dx < \infty\}. \tag{9}$$

The space $\mathcal{W}_2^{(\nu)}$ is also known as the Sobolev space. Let $\mathcal{S}Y = g_r$ ($r \in \mathbb{N}$) be the smoothing spline solution to the penalized least squares problem

$$g_r = g_r(\lambda) = \arg\min_{f \in \mathcal{W}_2^{(r)}} \frac{1}{n}\sum_i [Y_i - f(x_i)]^2 + \lambda \int [f^{(r)}(x)]^2\,dx. \tag{10}$$

For theoretical purposes, we assume a condition on the design of the predictor variables:

(A) The predictor variables $x_1, \ldots, x_n \in [a, b]$ ($-\infty < a < b < \infty$) are deterministic and satisfy; there exists a positive constant B such that for every n,

$$\frac{\sup_{x \in [a,b]} \inf_{1 \le i \le n} |x - x_i|}{\inf_{1 \le i \ne j \le n} |x_i - x_j|} \le B < \infty.$$

Assumption (A) holds for the uniform design and is almost surely fulfilled for iid realizations from a suitably regular probability distribution on $[a, b]$.

Theorem 3 (Optimality of L_2Boost for Smoothing Splines). Consider the model in (6) with $x_i \in \mathbb{R}$ satisfying (A). Suppose that $\mathcal{S}$ is a smoothing spline learner $g_r(\lambda_0)$ of degree r corresponding to a fixed smoothing parameter λ_0. If the true function f is in $\mathcal{W}_2^{(\nu)}$ with $\nu \ge r$ ($\nu, r \in \mathbb{N}$), then there is an $m = m(n) = O(n^{2r/(2\nu+1)}) \to \infty$ such that $\widehat{F}_{m(n)}$ achieves the optimal minimax rate $n^{-2\nu/(2\nu+1)}$ of the (smoother than degree r) function class $\mathcal{W}_2^{(\nu)}$ in terms of MSE as defined in (8).

A proof is given in the Appendix. This result states that boosting smoothing splines is minimax optimal and also adapts to higher-order smoothness. Even if the smoothing spline learner has only degree r, we can adapt to higher-order smoothness ν and achieve the optimal MSE rate. Interestingly, any fixed smoothing parameter λ_0 of the smoothing spline learner can be used; from the asymptotic standpoint, this means that the smoothing parameter λ_0 is large, that is, a smoothing spline learner learner with low variance.

Gu (1987) analyzed twicing ($m = 1$) and showed that it can adapt to a higher-order smoothness $\nu \le 2r$. With boosting, we can adapt to an arbitrarily higher-order smoothness, because we can refit as many times as we want. For cubic smoothing spline learners with $r = 2$, the optimal rate $n^{-4/5}$ for $\nu = 2$ is achieved by $m = O(n^{4/5})$. If the underlying smoothness is, say, $\nu = 3 > 2 = r$, then the boosted cubic smoothing spline can achieve the optimal rate $n^{-6/7}$ for the smoother class with $m = O(n^{4/7})$. In practice, the data-driven "optimal" boosting iteration m is selected either through a fixed test set or via cross-validation. We note here that boosting also adapts to lower-order smoothness $\nu < r$. But this is also true for smoothing splines (without boosting). Hence boosting is not offering more for this case.

For a given smoothness ν, both the ordinary smoothing spline (with $r = \nu$) and boosting achieve the optimal rate, but they trade off bias and variance along different regularization paths. As mentioned already, the advantage of the new exponential trade-off in boosting is the flatter near-optimal region for the optimal smoothing parameter (or boosting iteration). An example was shown in Figure 1 with simulated data from the next section.

3.2.2 Simulation Results With Cubic Smoothing Spline as Learners. The relevance of the foregoing theoretical results depends on the underlying problem and the sample size. We consider a representative example for the model in (6),

$$\begin{aligned} &f(x) = 0.8x + \sin(6x),\ x \in \mathbb{R}^1, \\ &\quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2),\ \sigma^2 = 2,\ \text{ sample size } n = 100. \end{aligned} \tag{11}$$

The learner $\mathcal{S}$ is chosen as a cubic smoothing spline that satisfies linearity, symmetry, and the eigenvalue conditions used in Theorems 1 and 2.

The complexity of $\mathcal{S}$, or the strength of the learner, is chosen here in terms of the so-called *degrees of freedom* (df), which equals the trace of $\mathcal{S}$ (Hastie and Tibshirani 1990). To study the interaction of the learner with the underlying problem, we fix the model as in (11) and a cubic smoothing spline learner with df $= 20$. To decrease the learner's complexity (or increase the learner's weakness), we use shrinkage (Friedman 2001) to replace $\mathcal{S}$ by

$$\mathcal{S}_\nu = \nu\mathcal{S}, \qquad 0 < \nu \le 1.$$

For $\mathcal{S}$, a smoothing spline estimator, shrinkage with ν small corresponds to a linear operator $\mathcal{S}_\nu$ whose eigenvalues $\{\nu\lambda_k\}_k$ are closer to 0 than $\{\lambda_k\}_k$ for the original $\mathcal{S}$. With small ν, we thus get a weaker learner than the original $\mathcal{S}$; here shrinkage acts similarly as changing the degrees of freedom of the original $\mathcal{S}$ to a lower value. We see its effect in more complex examples in Sections 4.2 and 8. The boosting question becomes whether even a very weak $\mathcal{S}_\nu$, with ν very small, can be boosted with m large to achieve almost-optimal performance (defined through numerical search among all of the estimators rising from different shrinkage factors and different iterations of boosting). Figure 2 displays MSE from specification (11) with $x_1, \ldots, x_n$ iid realizations from $\mathcal{N}(0, 1)$, as a function of m and ν. It shows that the earlier boosting question has a positive answer for this case. That is, we observe the following from this example:

1. Boosting with a large number of iterations has the potential to make a very weak learner (with ν very small) almost optimal when compared with the best shrunken learner $\nu_{opt}S$. This is consistent with the asymptotic result in Theorem 3.
2. Provided that the learner is sufficiently weak, boosting always improves, as we show in Theorem 1.
3. When the initial learner $\mathcal{S}$ is too strong, boosting decreases performance due to overfitting. (In our case, the learner with 20 df is too strong or has a too small amount of smoothing.)
4. Boosting very weak learners is relatively safe, provided that the number of iterations is large; the MSE with ν very low is flat for large number of iterations m.

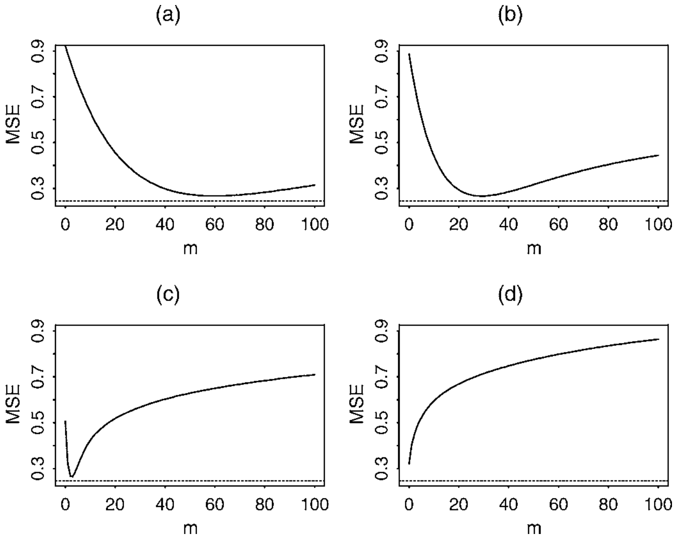

Figure 2. Traces of MSE as a Function of Boosting Iterations m, for Four Different Shrinkage Factors: (a) .02, (b) .04, (c) .3, (d) 1.0. Dotted line represents minimum, achieved with m = 0, ν = .76. The data are from model (11) with sample size n = 100.

Statements 2–4, numerically found for this example with a linear learner, have also been shown to hold empirically for many datasets and with nonlinear learners in AdaBoost and LogitBoost in classification. Statement 1, dealing with optimality, is asymptotically explained by our Theorem 3. Also, the recent work of Jiang (2003) hinted at a consistency result for AdaBoost; he showed that in the asymptotic sense and for classification, AdaBoost visits optimal (Bayes) procedures during the evolution of AdaBoost. However, this asymptotic consistency result does not explain optimality or AdaBoost's good finite-sample performance.

All of these behaviors happen well before the asymptopia, $m = \infty$, or the convergence of the boosting algorithm. For example, when having perfect knowledge of the MSE at every boosting iteration, as in our simulation example, a suitable stopping criterion would be the relative difference of MSE between the current and previous iteration. When using the tolerance, say 10^{-4}, for this relative difference, we would need $m = 761$, when using S_ν with $\nu = 1$, and the resulting MSE at stopping is 1.088, exhibiting an overfit. For $\nu = .02$, we would need $m = 1{,}691$, and the MSE at stopping is 1.146, which overfits again. This is still far from the stabilizing MSE ($m = \infty$), which is $\sigma^2 = 2$ (see Theorem 1). This little illustration describes how slow convergence in boosting could be.

Here we also demonstrate empirically the adaptivity of L_2Boost with smoothing splines to higher smoothness (see Theorem 3). We use simulated data from model (11) with $x_1, \ldots, x_n$ iid realizations from uniform($[-1/2, 1/2]$), for sample sizes $n = 10$ up to 1,000. Table 1 reports the performance of the best cubic smoothing spline (with optimal penalty parameter) in comparison with L_2Boosting a cubic smoothing spline with constant penalty $\lambda_0 = c$ (and using optimal number of boosting iterations). We evaluate the MSE $\mathbb{E}[(f(X) - \hat{f}(X))^2]$ (X a new test observation) by averaging over 100 simulations from the model (11). We observe the following in Table 1. For $n = 10$, the smoothing spline learner in L_2Boost is too strong, and actually the best performance is with $m = 0$ (no boosting). In the midrange with $25 \le n \le 100$, the differences between optimal smoothing spline and optimal L_2Boost are negligible.

Table 1. MSE for Simulated Data From (11)

Sample size n	Optimal smoothing spline	Optimal L_2Boost	Gain
10	$7.787 \cdot 10^{-1}$	$9.968 \cdot 10^{-1}$	−28.0%
25	$3.338 \cdot 10^{-1}$	$3.349 \cdot 10^{-1}$	−.3%
50	$1.657 \cdot 10^{-1}$	$1.669 \cdot 10^{-1}$	−.7%
100	$9.332 \cdot 10^{-2}$	$9.050 \cdot 10^{-2}$	.9%
1,000	$1.285 \cdot 10^{-2}$	$1.128 \cdot 10^{-2}$	12.2%

NOTE: Optimal cubic smoothing spline (with best penalty parameter) and optimal L_2Boost with smoothing spline (with best number of boosting iterations). Positive gain, which measures the relative improvement of MSE with L_2Boost, indicates an advantage for boosting.

For the large sample size, $n = 1{,}000$, we see an advantage of L_2Boost that is consistent with the theory: The underlying regression function in (11) is infinitely smooth, and L_2Boost exhibits an adaptation to higher-order smoothness.

3.3 Estimating the Optimal Number of Boosting Iterations

The number m of boosting iterations is a tuning parameter in L_2 and other boosting methods. Theorem 1 and Figure 1 indicate a certain resistance against overfitting as m gets large, and thus the tuning of L_2Boost should not be difficult. Nevertheless, a method is needed to choose the value of m based on data, or to estimate the optimal iteration number m.

A straightforward way that we found to work well is given by a five-fold cross-validation for estimating the MSE $\mathbb{E}[(Y - \widehat{F}_m(X))^2]$ in regression or the misclassification error $\mathbb{P}[sign(\widehat{F}_m(X)) \neq Y]$ in classification. An estimate $\hat{m}$ for the number of boosting iterations is then given as the minimizer of such a cross-validated error.

4. L_2 BOOSTING FOR REGRESSION IN HIGH DIMENSIONS

When the dimension d of the predictor space is large, the learner $\mathcal{S}$ is typically nonlinear. In very high dimensions, it becomes almost a necessity to use a learner that is doing some sort of variable selection. One of the most prominent examples are trees.

4.1 Componentwise Smoothing Spline as the Learner

As an alternative to tree learners with two terminal nodes (stumps), here we propose componentwise smoothing splines. A componentwise smoothing spline is defined as a smoothing spline with *one selected* explanatory variable $x_{\hat{\iota}}$ ($\hat{\iota} \in \{1, \ldots, d\}$), where

$$\hat{\iota} = \arg\min_{\iota} \sum_{i=1}^{n} (Y_i - \hat{g}_\iota(x_{i,\iota}))^2,$$

where $\hat{g}_\iota$ is the smoothing spline as defined in (10) using the predictor x_ι. Thus the componentwise smoothing spline learner is given by the function

$$\hat{g}_{\hat{\iota}} : x \mapsto \hat{g}_{\hat{\iota}}(x_{\hat{\iota}}), \qquad x \in \mathbb{R}^d.$$

Boosting stumps and componentwise smoothing splines yields an additive model whose terms are fitted in a stagewise fashion. This is because an additive combination of a stump or a componentwise smoothing spline $\widehat{F}_0 + \sum_{m=1}^{M} \hat{f}_m$, with $\hat{f}_m(x)$ depending functionally only on $x_{\hat{\iota}_m}$ for some component $\hat{\iota}_m \in \{1, \ldots, d\}$, can be reexpressed as an additive function, $\sum_{j=1}^{d} \hat{m}_j(x_j)$, $x \in \mathbb{R}^d$. The estimated functions $\hat{m}_j(\cdot)$ when using boosting are fitted in stagewise fashion and different from the backfitting estimates in additive models (Hastie and Tibshirani 1990). Boosting stumps or componentwise smoothing splines is particularly attractive when aiming to fit an additive model in very high dimensions with d larger or of the order of sample size n. Boosting has then much greater flexibility to add complexity, in a stagewise fashion, to certain components $j \in \{1, \ldots, d\}$ and may even drop some of the variables (components). We give examples in Section 4.2 illustrating that when the dimension d is large, boosting outperforms the alternative classical additive modeling with variable selection, using backfitting.

4.2 Numerical Results

We first consider the often-analyzed dataset of ozone concentration in the Los Angeles basin, which has been also considered by Breiman (1998) in connection with boosting. The dimension of the predictor space is $d = 8$, and the sample size is $n = 330$. Here we compare L_2 boosting with classical additive models using backfitting and with multivariate adaptive regression splines (MARS). L_2Boost is used with stumps and with componentwise cubic smoothing splines with 5 df (Hastie and Tibshirani 1990), and the number of iterations is estimated by five-fold cross-validation as described in Section 3.3. Additive model backfitting is used with the default smoothing splines in S-PLUS, and MARS is run by using the default parameters as implemented in S-PLUS, library(mda), which is an additive fit using a forward selection method (the default parameters do not allow for interactions). We estimate mean squared prediction error, $\mathbb{E}[(Y - \widehat{F}(X))^2]$ with $\widehat{F}(x) = \widehat{\mathbb{E}}[Y|X = x]$, by randomly splitting the data into 297 training observations and 33 test observations and averaging 50 times over such random partitions; Table 2 displays the results. We conclude that L_2Boost with componentwise splines is better than with trees and that it is among the best, together with classical backfitting of additive models. Moreover, estimation of the number of boosting iterations works very satisfactorily, exhibiting a performance that is very close to the optimum.

Next we show a simulated example in very high dimension relative to sample size, where L_2Boost as a stagewise method is better than backfitting for additive models with variable selection. The simulation model is

$$Y = \sum_{j=1}^{100} (1 + (-1)^j A_j X_j + B_j \sin(6X_j)) \sum_{j=1}^{50} (1 + X_j/50) + \varepsilon,$$

$$\begin{aligned} & A_1, \ldots, A_{100} \text{ iid unif}([0.6, 1]), \\ & B_1, \ldots, B_{100} \text{ iid unif}([0.8, 1.2]), \text{ independent} \\ & \quad \text{from the } A_j\text{'s}, \\ & X \sim \text{unif}([0, 1]^{100}) \text{ where all components are} \\ & \qquad \text{iid} \sim \text{unif}([0, 1]), \\ & \varepsilon \sim \mathcal{N}(0, 2). \end{aligned} \tag{12}$$

Samples of size $n = 200$ are generated from model (12); for each model realization (realized coefficients $A_1, \ldots, A_{100}$,

Table 2. Cross-Validated Test Set MSEs for Ozone Data

Method	*MSE*
L_2Boost with componentwise spline	17.78
L_2Boost with stumps	21.33
Additive model (backfitted)	17.41
MARS	18.09
With optimal no. of iterations	
L_2Boost with componentwise spline	17.50 (5)
L_2Boost with stumps	20.96 (26)

NOTE: L_2Boost with estimated and optimal (minimizing cross-validated test set error) number of boosting iterations, given in parentheses. The componentwise spline is a cubic smoothing spline with df = 5.

Table 3. MSEs for Simulated Data From (12) With $n = 200$

Method	*MSE*
L_2Boost with shrunken componentwise spline	11.87
L_2Boost with shrunken stumps	12.76
MARS	25.19
Shrunken MARS	15.05
With optimal no. of iterations or variables	
L_2Boost with shrunken componentwise spline	10.69 (228)
L_2Boost with shrunken stumps	12.54 (209)
Additive model (backfitted and forward selection)	16.61 (1)
Shrunken additive model (backfitted and forward selection)	14.44 (19)

NOTE: L_2Boost with estimated and optimal (minimizing MSE) number of boosting iterations, given in parentheses; additive model with optimal number of selected variables, given in parentheses. The componentwise spline is a cubic smoothing spline with df $= 5$; shrinkage factor is always $\nu = .5$.

$B_1, \ldots, B_{50}$), we generate 200 iid pairs (Y_i, X_i) $(i = 1, \ldots, n = 200)$.

We take the same approach as before, but using classical additive modeling with a forward variable selection (inclusion) strategy because $d = 100$ is very large compared with $n = 200$; this should be similar to MARS which by default doesn't allow for interactions. We evaluate $\mathbb{E}[(\widehat{F}(X) - \mathbb{E}[Y|X])^2]$ (X a new test observation) at the true conditional expectation, which can be done in simulations. Already a stump appeared to be too strong for L_2Boost, and we thus used shrunken learners $\nu\mathcal{S}$ with $\nu = .5$ chosen ad hoc; we also allowed the nonboosting procedures to be shrunken with $\nu = .5$. Table 3 shows the average performance over 10 simulations from model (12). L_2Boost is the winner over additive models and MARS, and the componentwise spline is a better learner than stumps. Note that only the methods in the upper part of Table 3 are fully data-driven (the additive model fits use the number of variables minimizing the true MSE). We believe that boosting has a clear advantage over other flexible nonparametric methods mainly in such high-dimensional situations, and not so much in examples where the dimension d is "midrange" or even small relative to sample size.

For simulated data, assessing significance for differences in performance is easy. Because all of the different methods are used on the same data realizations, pairwise comparisons should be made. Table 4 displays the p values of paired Wilcoxon tests. We conclude that for this simulation model (12), L_2Boost with componentwise smoothing spline is best and significantly outperforms the more classical methods, such as MARS or additive models.

Table 4. p Values From Paired Two-Sided Wilcoxon Tests for Equal MSEs

	L_2Boost shrunken stumps	*Shrunken MARS*	*Shrunken additive model*
L_2Boost shrunken componentwise spline	.193	.010	.004
L_2Boost shrunken stumps		.010	.014
Shrunken MARS			.695

NOTE: Simulated data from (12) with $n = 200$ and 10 independent model realizations. Low p values of two-sided tests are always in favor of the method in the row; that is, the sign of test statistic always points toward favoring the method in the row. L_2Boost with the estimated number of boosting iterations; specification of the methods as in Table 3.

5. L_2 BOOSTING FOR CLASSIFICATION

The L_2Boost algorithm can also be used for classification, and it enjoys a computational simplicity in comparison with AdaBoost and LogitBoost.

5.1 Two-Class Problem

Consider a training sample

$$(Y_1, X_1), \ldots, (Y_n, X_n) \text{ iid}, \qquad Y_i \in \{-1, 1\}, \qquad X_i \in \mathbb{R}^d. \tag{13}$$

L_2 boosting then yields an estimate $\widehat{F}_m$ for the unknown function $\mathbb{E}[Y|X = x] = 2p(x) - 1$, where $p(x) = \mathbb{P}[Y = 1|X = x]$; the classification rule is given by (5). Note that also in two-class problems, the generic functional gradient descent repeatedly fits some learner $h(x, \theta)$ taking values in $\mathbb{R}$.

In addition to the L_2Boost algorithm, we propose a modification called "L_2Boost with constraints" (L_2WCBoost). It proceeds as L_2Boost, except that $\widehat{F}_m(x)$ is constrained to be in $[-1, 1]$: this is natural because the target $F(x) = \mathbb{E}[Y|X = x] \in [-1, 1]$ is also in this range.

L_2WCBoost algorithm

Step 1. $\widehat{F}_0(x) = h(x; \hat{\theta}_{Y,X})$ by least squares fitting. Set $m = 0$.
Step 2. Compute residuals $U_i = Y_i - \widehat{F}_m(X_i)$ $(i = 1, \ldots, n)$. Then fit $(U_1, X_1), \ldots, (U_n, X_n)$ by least squares,

$$\hat{f}_{m+1}(x) = h(x, \hat{\theta}_{U,X}).$$

Update

$$\widetilde{F}_{m+1}(\cdot) = \widehat{F}_m(\cdot) + \hat{f}_{m+1}(\cdot)$$

and

$$\widehat{F}_{m+1}(x) = sign(\widetilde{F}_{m+1}(x)) \min\bigl(1, |\widetilde{F}_{m+1}(x)|\bigr).$$

Note that if $|\widetilde{F}_{m+1}(x)| \le 1$, the updating step is as in the L_2Boost algorithm.
Step 3. Increase m by 1 and go back to step 2.

5.1.1 Theory. For theoretical purposes, consider the model in (13) but with fixed, real-valued predictors,

$$Y_i \in \{-1, 1\} \text{ independent}, \qquad \mathbb{P}[Y_i = 1|x_i] = p(x_i),$$
$$x_i \in \mathbb{R} \quad (i = 1, \ldots, n). \tag{14}$$

Estimating $\mathbb{E}[Y|x_i] = 2p(x_i) - 1$ in classification can be seen as estimating the regression function in a heteroscedastic model,

$$Y_i = 2p(x_i) - 1 + \varepsilon_i \qquad (i = 1, \ldots, n),$$

where ε_i are independent mean 0 variables, but with variance $4p(x_i)(1 - p(x_i))$. Because the variances are bounded by 1, the arguments in the regression case can be modified to give the optimal rates of convergence results for estimating p.

Theorem 4 (Optimality of L_2Boost for Smoothing Splines in Classification). Consider the model in (14) with x_i satisfying (A); see Section 3.2.1. Suppose that $p \in \mathcal{W}_2^{(\nu)}$ [see (9)] and that $\mathcal{S}$ is a smoothing spline linear learner $g_r(\lambda_0)$ of degree r, corresponding to a fixed smoothing parameter λ_0 [see (10)]. If

$\nu \geq r$, then there is an $m = m(n) = O(n^{2r/(2\nu+1)}) \to \infty$ such that $\widehat{F}_{m(n)}$ achieves the optimal minimax rate $n^{-2\nu/(2\nu+1)}$ of the smoother function class $\mathcal{W}_2^{(\nu)}$ for estimating $2p(\cdot) - 1$ in terms of MSE as defined in (8).

It is well known that 2 times the L_1 norm bounds from above the difference between the generalization error of a plug-in classifier (expected 0–1 loss error for classifying a new observation) and the Bayes risk (cf. theorem 2.3 of Devroye, Györfi, and Lugosi 1996). Furthermore, the L_1 norm is upper bounded by the L_2 norm. It follows from Theorem 4 that if the underlying p belongs to one of the smooth function class $\mathcal{W}_2^{(\nu)}$, then L_2 boosting converges to the average Bayes risk (ABR),

$$ABR = n^{-1} \sum_{i=1}^{n} \mathbb{P}[sign(2p(x_i) - 1) \neq Y_i].$$

Moreover, the L_2 bound implies the following.

Corollary 2. Under the assumptions and specifications in Theorem 4,

$$n^{-1} \sum_{i=1}^{n} \mathbb{P}[sign(\widehat{F}_{m(n)}(x_i)) \neq Y_i] - ABR = O(n^{-\nu/(2\nu+1)}).$$

Because the functions in $\mathcal{W}_2^{(\nu)}$ are bounded, using Hoeffding's inequality we can show that for both terms in Corollary 2, replacing the average by an expectation with respect to a density for x would cause an error of order $o(n^{-\nu/(2\nu+1)})$ because $\nu/(2\nu + 1) < 1/2$. Hence the foregoing result also holds if we replace the averaging by an expectation with respect to a randomly chosen x with a design density and replace the ABR by the corresponding Bayes risk.

For the generalization error with respect to a random x, Yang (1999) showed that the foregoing $n^{-\nu/(2\nu+1)}$ rate is also minimax optimal for classification over the Lipschitz family $\mathcal{L}_2^{(\nu)}$,

$$\mathcal{L}_2^{(\nu)} = \{p : \|p(x + h) - p(x)\|_2 < Ch^\nu,\ \|p\|_2 < C\}, \quad (15)$$

where $\|p\|_2 = (\int p^2(x)\, dx)^{1/2}$. Yang (1999) used a hypercube subclass in $\mathcal{L}_2^{(\nu)}$ to prove the lower bound rate $n^{-\nu/(2\nu+1)}$. This hypercube subclass also belongs to our $\mathcal{W}_2^{(\nu)}$. Hence the rate $n^{-\nu/(2\nu+1)}$ also serves as a lower bound for our $\mathcal{W}_2^{(\nu)}$. Thus we have proved that the minimax optimal rate $n^{-\nu/(2\nu+1)}$ of convergence for the classification problem over the global smoothness class $\mathcal{W}_2^{(\nu)}$.

Marron (1983) gave a faster classification minimax rate of convergence $n^{-2\nu/(2\nu+1)}$ for a more restrictive global smoothness class, and the same rate holds for us if we adopt that class. On the other hand, Mammen and Tsybakov (1999) considered different function classes that are locally constrained near the decision boundary and showed that a faster rate than the parametric rate n^{-1} can even be achieved. The local constrained classes are more natural in the classification setting with the 0–1 loss because, as seen from our smoothed 0–1 loss expansion in Section 6.1, the actions happen near the decision boundary. These new rates are achieved by avoiding the plug-in classification rules through estimating p. Instead, empirical minimization of the 0–1 loss function over regularized classes of decision regions is used. However, computationally such a minimization could be very difficult. It remains an open question as to whether boosting can achieve the new optimal convergence rates of Mammen and Tsybakov (1999).

5.2 Multiclass Problem

The multiclass problem has response variables $Y_i \in \{1, 2, \ldots, J\}$, taking values in a finite set of labels. An estimated classifier, under equal misclassification costs, is then given by

$$\widehat{\mathcal{C}}(x) = \arg \max_{j \in \{1,\ldots,J\}} \widehat{\mathbb{P}}[Y = j | X = x].$$

The conditional probability estimates $\hat{p}_j(x) = \widehat{\mathbb{P}}[Y = j | X = x]$ $(j = 1, \ldots, J)$ can be constructed from J different two-class problems, where each two-class problem encodes the events $\{Y = j\}$ and $\{Y \neq j\}$, this is also known as the "one against all" approach (Allwein, Schapire, and Singer 2001). Thus the L_2Boost algorithm for multiclass problems works as follows:
Step 1. Compute $\widehat{F}_m^{(j)}(\cdot)$ as an estimate of $p_j(\cdot)$ with L_2- or L_2WCBoost $(j = 1, \ldots, J)$ on the basis of binary response variables

$$Y_i^{(j)} = \begin{cases} 1 & \text{if } Y_i = j \\ -1 & \text{if } Y_i \neq j \end{cases}, \quad i = 1, \ldots, n.$$

Step 2. Construct the classifier as

$$\widehat{\mathcal{C}}_m(x) = \arg \max_{j \in \{1,\ldots,J\}} \widehat{F}_m^{(j)}(x).$$

The optimality results from Theorem 4 carry over to the multiclass case. Of course, other codings of a multiclass problem into multiple two-class problems can be done (see Allwein et al. 2001). The "one against all" scheme did work well in the cases that we were examining (see also Sec. 7.2).

6. UNDERSTANDING THE 0–1 LOSS IN TWO-CLASS PROBLEMS

6.1 Generalization Error via Tapered Moments of the Margin

Here we consider again the two-class problem as in (13). The performance is often measured by the generalization error

$$\mathbb{P}[sign(\widehat{F}_m(X)) \neq Y] = \mathbb{P}[Y \widehat{F}_m(X) < 0] = \mathbb{E}[\mathbf{1}_{[Y \widehat{F}_m(X) < 0]}], \quad (16)$$

where $\mathbb{P}$ and $\mathbb{E}$ are over all of the random variables in the training set (13) and the testing observation $(Y, X) \in \{-1, 1\} \times \mathbb{R}^d$, which is independent of the training set. Insights about the expected 0–1 loss function in (16), the generalization error, can be gained by approximating it, for theoretical purposes, with a smoothed version,

$$\mathbb{E}[C_\gamma(Y \widehat{F}_m(X))],$$

$$C_\gamma(z) = \left(1 - \frac{\exp(z/\gamma)}{2}\right) \mathbf{1}_{[z<0]} + \frac{\exp(-z/\gamma)}{2} \mathbf{1}_{[z \geq 0]}, \quad \gamma > 0.$$

The parameter γ controls the quality of approximation.

Proposition 4. Assume that the distribution of $Z = Y \widehat{F}_m(X)$ has a density $g(z)$ that is bounded for z in a neighborhood around 0. Then

$$|\mathbb{P}[Y \widehat{F}_m(X) < 0] - \mathbb{E}[C_\gamma(Y \widehat{F}_m(X))]| = O(\gamma \log(\gamma^{-1})) \quad (\gamma \to 0).$$

A proof is given in the Appendix. Proposition 4 shows that the generalization error can be approximated by an expected cost function that is infinitely often differentiable.

Proposition 4 motivates to study generalization error through $\mathbb{E}[C_\gamma(Z)]$ with $Z = Y\widehat{F}_m(X)$. Applying a Taylor series expansion of $C_\gamma(\cdot)$ around $Z^* = YF(X)$, we obtain

$$\mathbb{E}[C_\gamma(Z)] = \mathbb{E}[C_\gamma(Z^*)] + \sum_{k=1}^{\infty} \frac{1}{k!}\mathbb{E}\big[C_\gamma^{(k)}(Z^*)(Z-Z^*)^k\big]. \tag{17}$$

Thereby the variables Z and Z^* denote the so-called estimated and true margin; a positive margin denotes a correct classification (and vice versa), and the actual value of the margin describes closeness to the classification boundary $\{x : F(x) = 0\}$. The derivatives of $C_\gamma(\cdot)$ are

$$C_\gamma^{(k)}(z) = \frac{1}{\gamma^k}\exp\left(\frac{-|z|}{\gamma}\right)(-\mathbf{1}_{[z<0]} + (-1)^k\mathbf{1}_{[z\geq 0]}). \tag{18}$$

Using conditioning on the test observations (Y, X), the moments can be expressed as

$$\begin{aligned}&\mathbb{E}\big[C_\gamma^{(k)}(Z^*)(Z-Z^*)^k\big]\\ &\quad = \sum_{y\in\{-1,1\}} \int C_\gamma^{(k)}(yF(x))y^k b_k(x)\mathbb{P}[Y=y|X=x]\,dP_X(x),\\ &\qquad b_k(x) = \mathbb{E}[(\widehat{F}_m(x) - F(x))^k], \text{ where expectation}\\ &\qquad \text{is over the training set in (13).}\end{aligned} \tag{19}$$

Thereby $P_X(\cdot)$ denotes the distribution of the predictors X.

From (17) and (19), we see that the smooth approximation to the generalization error of any procedure is essentially, in addition to the approximate Bayes risk $\mathbb{E}[C_\gamma Z^*]$, the averaged (as in (19)) sum of moments $b_k(x)$, tapered by $C_\gamma^{(k)}(yF(x))/k!$, which decays very quickly as $yF(x)$ moves away from 0. This exploits from a different view, the known fact that only the behavior of $b_k(x)$ in the neighborhood of the classification boundary $\{x; F(x) = 0\}$ matters to the generalization error.

The first two terms in the approximation (17) are the tapered bias and the tapered L_2 term; see (19) with $k = 1$ and 2. The higher-order terms can be expanded as terms of interactions between the centered moments and the bias term (all tapered),

$$\begin{aligned}b_k(x) &= \mathbb{E}[(\widehat{F}_m(x) - F(x))^k]\\ &= \sum_{j=0}^{k}\binom{k}{j} b_1(x)^j\mathbb{E}[(\widehat{F}_m(x) - \mathbb{E}[\widehat{F}_m(x)])^{k-j}].\end{aligned} \tag{20}$$

This seemingly trivial approximation has three important consequences. The first consequence is that bias (after tapering) as the first term in (17) and multiplicative terms in higher moments [see (20)] plays a bigger role in (smoothed) 0–1 loss classification than in L_2 regression. Second, in the case of boosting, because all of the (tapered) centered moment terms $b_k(x)$ in (19) are bounded by expressions with exponentially diminishing increments as boosting iterations m get large (see Sec. 3, particularly Theorem 2) we gain resistance against overfitting. In view of the additional tapering, this resistance is even stronger than in regression (see the rejoinder of Friedman et al. 2000 for a relevant illustration). The third consequence of the approximations in (17), (19), and (20) is to suggest why the previous attempts were not successful at decomposing the 0–1 prediction (generalization) error into additive bias and variance terms (Geman et al. 1992; Breiman 1998; references therein). This is because, except for the first two terms, all other important terms include the bias term also in a multiplicative fashion [see (20)] for each term in the summation (17), instead of in a pure additive way.

We conclude heuristically that the exponentially diminishing increase of centered moments as the number of boosting iterations grow (as stated in Theorem 2), together with the tapering in the smoothed 0–1 loss, yields the overall (often strong) overfitting-resistance performance of boosting in classification.

6.2 Acceleration of F and Classification Noise

As seen in Section 6.1, resistance against overfitting is closely related to the behavior of $F(\cdot)$ at the classification boundary. If the true $F(\cdot)$ moves away quickly from the classification boundary $\{x; F(x) = 0\}$, then the relevant tapering weights $C_\gamma^{(k)}(yF(x))$ decay very fast. This can be measured with $\mathrm{grad}(F(x))|_{x=0}$, the gradient of F at 0. $F(\cdot)$ is said to have a large acceleration if its gradient is large (elementwise in absolute values, or in Euclidean norm). Thus a large acceleration of $F(\cdot)$ should result in strong resistance against overfitting in boosting.

Noise negatively affects the acceleration of $F(\cdot)$. Noise in model (13), often called "classification noise," can be thought of in a constructive way. Consider a random variable $W \in \{-1, 1\}$, independent from (Y, X) with $\mathbb{P}[W = -1] = \pi$, $0 \leq \pi \leq 1/2$. The noisy response variable is

$$\widetilde{Y} = WY, \tag{21}$$

changing the sign of Y with probability π. Its conditional probability is easily seen to be

$$\mathbb{P}[\widetilde{Y} = 1|X = x] = \mathbb{P}[Y = 1|X = x](1-2\pi) + \pi, \qquad 0 \leq \pi \leq 1/2. \tag{22}$$

Denote by $\widetilde{F}(\cdot)$ the noisy version of $F(\cdot)$, with $\mathbb{P}[\widetilde{Y} = 1|X = x]$ replacing $\mathbb{P}[Y = 1|X = x]$ for $F(\cdot)$ being either half of the log-odds ratio or the conditional expectation [see (3)]. A straightforward calculation then shows

$$\mathrm{grad}(\widetilde{F}(x))|_{x=0} \searrow 0 \text{ as } \pi \nearrow 1/2. \tag{23}$$

The noisier the problem, the smaller the acceleration of $\widetilde{F}(\cdot)$ and thus the lower the resistance against overfitting, because the tapering weights in (18) are becoming larger in noisy problems. This adds insights to the known empirical fact that boosting does not work well in noisy problems (see Dietterich 2000). Typically, overfitting then kicks in early, and many learners are too strong in noisy problems. Using LogitBoost (Friedman et al. 2000), this effect is demonstrated in Figure 3 for the breast cancer data available at (*http://www.ics.uci.edu/~mlearn/MLRepository*), which has been analyzed by many others. We see there that a stump already becomes too strong for LogitBoost in the 25% or 40% noise-added breast cancer data.

Of course, adding noise makes the classification problem harder. The optimal Bayes classifier $\widetilde{\mathcal{C}}_{Bayes}(\cdot)$ in the noisy prob-

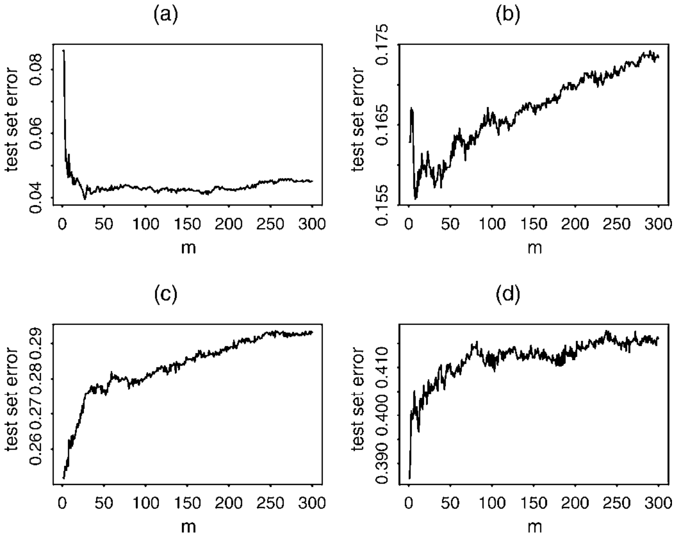

Figure 3. Test Set Misclassification Errors for LogitBoost With Stumps. Breast cancer data with different noise levels π (in %) as described in (21): (a) no noise; (b) 10% noise; (c) 25% noise; (d) 40% noise.

lem has generalization error

$$\mathbb{P}[\widetilde{Y}(X) \neq \widetilde{\mathcal{C}}_{Bayes}(X)] = \pi + (1-2\pi)\mathbb{P}[Y(X) \neq \mathcal{C}_{Bayes}(X)], \qquad 0 \leq \pi \leq 1/2,$$

relating it to the Bayes classifier $\mathcal{C}_{Bayes}$ of the original nonnoisy problem. This is easily derived using (22). With high noise, the expression is largely dominated by the constant term π, indicating that there is not much to gain by using a clever classifier (say, close to optimal Bayes) instead of a naive one. Even more so, the *relative* improvement, which is often used to demonstrate the better performance of a powerful classifier (over a benchmark), becomes less dramatic because the high noise level causes a large misclassification benchmark rate.

7. COMPARING L_2BOOST WITH LOGITBOOST ON REAL AND SIMULATED DATASETS

We compare L_2Boost, our L_2WCBoost, and LogitBoost using tree learners and our componentwise smoothing spline learner from Section 4.1.

7.1 Two-Class Problems

We consider first a variety of two-class problems from the UCI machine learning repository (*http://www.ics.uci.edu/~mlearn/MLRepository*): breast cancer, ionosphere, monk 1 (full dataset with $n = 432$) and the heart, sonar, and Australian credit datasets from the Statlog project. Monk 1 has a Bayes error equal to 0. The estimated test set misclassification errors, using an average of 50 random divisions into training with 90% of the data and test set with 10% of the data, are given in Table 5. The comparison is made when using the optimal (with respect to cross-validated test set error) number of boosting iterations for every boosting algorithm; these numbers are given in parentheses. As has been well documented (e.g., Breiman 1998; Dietterich 2000; Friedman et al. 2000), boosting with trees is usually much better than CART, displayed in Table 6. Here we use two versions of CART, one in which an initial large tree is pruned, with the amount of pruning estimated by cross-validation and the other using the default settings in S-PLUS for an unpruned tree; see Table 6. The componentwise learner is used only for the breast cancer data with ordinal predictors and for the sonar data with continuous predictors. For the latter, spline smoothing makes the most sense, and we also found empirically that there is where it improves on the tree-based stumps. Estimating the number of boosting iterations can be done by five-fold cross-validation as described in Section 3.3. The effect of using the optimal number of boosting iterations instead of an estimated number is typically small; for example, with the breast cancer data, estimating the number of boosting iterations by five-fold cross-validation yields a final performance for L_2WCBoost with stumps as .043, compared to .040 when using the optimal stopping. Overall, L_2WCBoost performs a bit better than L_2Boost, although in half of the datasets L_2Boost was better. LogitBoost was better than L_2Boost in four out of six datasets and better than L_2WCBoost in five out of six datasets, but most often only by a small amount. The biggest difference in performance is for the Sonar data, which has the most extreme ratio of dimension d to sample size n. But also for this dataset, the difference is far from being significant because sample size is much too small.

Therefore, we next consider simulated data to compare the L_2WCBoost (the slightly better of the L_2 procedures) with the LogitBoost algorithm. The simulation allows a more accurate comparison by choosing large test sets. We generate data with two classes from the model of Friedman et al. (2000) with a nonadditive decision boundary,

$$X \sim \mathcal{N}_{10}(0, I), \quad Y|X = x \sim 2\ \text{Bernoulli}(p(x)) - 1,$$

$$\log(p(x)/(1-p(x))) = 10 \sum_{j=1}^{6} x_j \left(1 + \sum_{k=1}^{6} (-1)^k x_k\right). \qquad (24)$$

Table 5. Test Set Misclassification Errors for L_2Boost, L_2WCBoost (with constraints), and LogitBoost

Dataset	*n*	*d*	*Learner*	*L_2Boost*	*L_2WCBoost*	*LogitBoost*
Breast cancer	699	9	Stumps	.037 (176)	.040 (275)	.039 (27)
			Componentwise spline	.036 (126)	.043 (73)	.038 (5)
Sonar	210	60	Stumps	.228 (62)	.190 (335)	.158 (228)
			Componentwise spline	.178 (51)	.168 (47)	.148 (122)
Ionosphere	351	34	Stumps	.088 (25)	.079 (123)	.070 (157)
Heart (without costs)	270	13	Stumps	.167 (4)	.175 (3)	.159 (3)
Australian credit	690	14	Stumps	.123 (22)	.123 (19)	.131 (16)
Monk 1	432	7	Larger tree	.002 (42)	.004 (12)	0 (6)

NOTE: The optimal (minimizing cross-validated test set error) number of boosts is given in parentheses; if the optimum is not unique, the minimum is given. "Larger tree" denotes a tree learner such that the ancestor nodes of the terminal leaves contain at most 10 observations, the resulting average tree size (integer-rounded) for L_2WCBoost is 12 terminal nodes.

Table 6. Test Set Misclassification Errors for CART

	Breast cancer	Sonar	Ionosphere	Heart	Australian credit	Monk 1
Unpruned CART	.060	.277	.136	.233	.151	.164
Pruned CART	.067	.308	.104	.271	.146	.138

NOTE: Unpruned version with defaults from S-PLUS; pruning with cost-complexity parameter, estimated via minimal cross-validated deviance.

The (training) sample size is $n = 2{,}000$. It is interesting to consider the performance on a single set of training and test data that resembles the situation in practice. Toward that end, we choose a very large test set of size 100,000, so that variability of the test set error given the training data is very small. In addition, we consider an average performance over 10 independent realizations from the model; here we choose a test set size of 10,000, which is still large compared with the training sample size of $n = 2{,}000$. (The latter is the same setup as in Friedman et al. 2000.)

Figure 4 displays the results on a single dataset. L_2WCBoost and LogitBoost perform about equally well. L_2WCBoost with the larger regression tree learner having about nine terminal nodes has a slight advantage. Friedman et al. (2000) have pointed out that stumps are not good learners, because the true decision boundary is nonadditive. With stumps as learners, the optimally (minimizing test set error) stopped LogitBoost has a tiny advantage over the optimally stopped L_2WCBoost (better by about .13 estimated standard errors for the test set error estimate, given the training data). With larger trees, L_2Boost has a more substantial advantage over LogitBoost (better by about 1.54 standard errors, given the data).

Figure 5 shows the averaged performance over 10 independent realizations with larger trees as learner; it indicates a more substantial advantage of L_2WCBoost than on the single data represented by Figure 4. With 10 independent realizations, we test whether one of the optimally (minimizing test set error) stopped Boosting algorithms yields a significantly better test set misclassification error. In 9 of the 10 simulations, optimally stopped L_2WCBoost was better than LogitBoost. The p values for testing the hypothesis of equal performance against the two-sided alternative are given in Table 7. Thus for the model (24) from Friedman et al. (2000), we find a significant advantage of L_2WCBoost over LogitBoost.

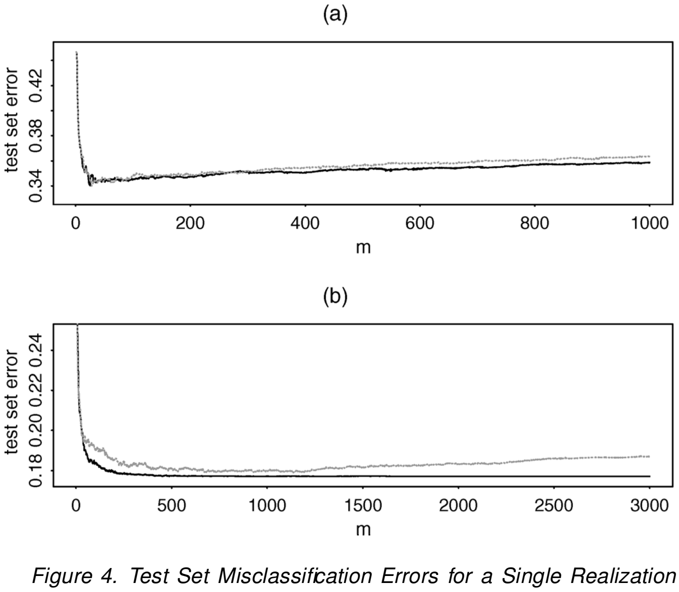

Figure 4. Test Set Misclassification Errors for a Single Realization from Model (24). (a) Stumps as learner. (b) Larger tree as learner with at most 175 observations per terminal node (the integer-rounded average tree size for L_2WCBoost is 9 terminal nodes). (—— L_2WCBoost; .... LogitBoost (in grey).)

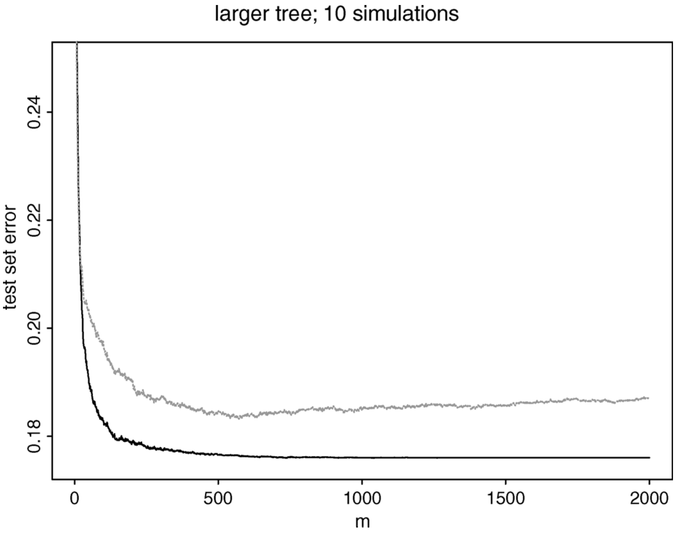

Figure 5. Test Set Misclassification Errors Averaged Over 10 Realizations From Model (24). Larger tree as learner with at most 175 observations per terminal node (the integer-rounded average tree size for L_2WCBoost is 9 terminal nodes). (—— L_2WCBoost; .... LogitBoost (in grey).)

Table 7. Comparison of L_2WCBoost and LogitBoost: Two-Sided Testing for Equal Test Set Performance

	t test	Wilcoxon test	Sign test
p value	.0015	.0039	.0215

NOTE: Low p values are always in favor of L_2WCBoost.

It is not so surprising that L_2WCBoost and LogitBoost perform similarly. In nonparametric problems, loss functions are used locally in a neighborhood of a fitting point $x \in \mathbb{R}^d$; an example is the local likelihood framework (Loader 1999). But locally, the L_2 and negative log-likelihood (with Bernoulli distribution) losses have the same minimizers.

7.2 Multiclass Problems

We also ran the L_2WCBoost algorithm using the "one against all" approach described in Section 7.2 on two often-analyzed multiclass problems. The results are given in Table 8. The test set error curve remained essentially flat after the optimal number of boosting iterations; for the Satimage data, the range of the test set error was [.096, .105] for iteration numbers 148–800 (prespecified maximum), whereas for the Letter data, the range was [.029, .031] for iteration numbers 460–1,000 (prespecified maximum). Therefore, a similar performance is expected with estimated number of iterations.

In comparison to our results with L_2WCBoost, Friedman et al. (2000) obtained the following with their LogitBoost algorithm using the fixed number of 200 boosting iterations, .102 with stumps and .088 with an 8-terminal node tree for the

Table 8. Test Set Misclassification Errors for L_2WCBoost

Dataset	n_{train}, n_{test}	*d*	*No. of classes*	*Learner*	*Error*
Satimage	4,435, 2,000	36	6	stumps	.110 (592)
				≈8 terminal node tree	.096 (148)
Letter	16,000, 4,000	16	26	≈30 terminal node tree	.029 (460)

NOTE: Optimal number of boosts (minimizing test set error) is given in parentheses; if the optimum is not unique, the minimum is given. The approximate tree size is the integer-rounded average of tree sizes used during all 800 (Satimage) or 1,000 (Letter) boosting iterations.

Satimage data and .033 with an 8-terminal node tree for the Letter data. We have also tuned CART and obtained the following test set error rates: .146 for the Satimage and .133 for the Letter dataset.

Thus we find here a similar situation as for the two-class problems. L_2WCBoost is about as good as LogitBoost, and both are better than CART (by a significant amount for the letter dataset).

8. DISCUSSION AND CONCLUDING REMARKS

As seen in Section 3, the computationally simple L_2 boosting is successful if the learner is sufficiently weak (sufficiently low variance); see also Figure 2. If the learner is too strong (too complex), then even at the first boosting iteration, the MSE is not improved over the original learner. Also in the classification case it is likely that the generalization error will then increase with the number of boosting steps, stabilizing eventually due to the numerical convergence of the boosting algorithm to the fully saturated model (at least for linear learners with eigenvalues bounded away from 0). Of course, the weakness of a learner also depends on the underlying signal-to-noise ratio (see also assertion c in Theorem 1). But the degree of weakness of a learner can always be increased by an additional shrinkage using $\mathcal{S}_\nu = \nu\mathcal{S}$ with shrinkage factor $0 < \nu < 1$. For L_2Boost with one-dimensional predictors, we have presented asymptotic results in Section 3.2 stating that boosting weak smoothing splines is as good as or even better than using an optimal nonboosted smoothing spline, because boosting adapts to unknown smoothness. In high dimensions, we see the most practical advantages of boosting; we have exemplified this in Section 4.2.

The fact that the effectiveness of L_2Boost depends on the strength of the learner and the underlying problem is also (empirically) true for other variants of boosting. Figure 6 shows some results for LogitBoost with trees and with projection pursuit learners having one ridge function (one term) for the breast cancer data. All tree learners are sufficiently weak for the problem, and the shrunken large tree seems best. Interestingly, the projection pursuit learner is already strong, and boosting does not pay off. This matches the intuition that projection pursuit is very flexible, complex, and hence strong. Boosting projection pursuit a second time ($m = 2$) reduces performance; this estimate with $m = 2$ can then be improved by further boosting. Eventually the test set error curve exhibits overfitting and stabilizes at a relatively high value. We observed a similar pattern with projection pursuit learners in another simulated example. Such a multiminimum phenomenon is typically not found with tree learners, but it is not inconsistent with our theoretical arguments.

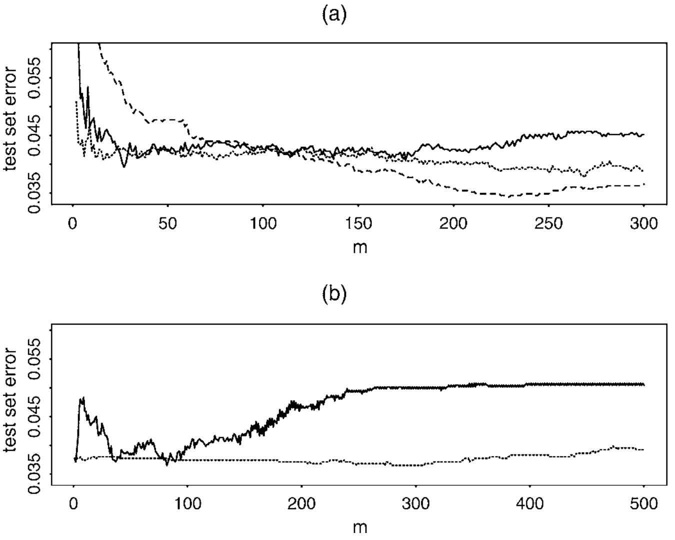

Figure 6. Test Set Misclassification Errors of LogitBoost for Breast Cancer Data. (a) Decision tree learners, namely stumps (——), large unpruned tree (····) and shrunken large unpruned tree (- - - - -) with shrinkage factor $\nu = .01$. (b) Projection pursuit (PPR) learners, namely one-term PPR (——) and shrunken one-term PPR (····) with shrinkage factor $\nu = .01$.

Most empirical studies of boosting in the literature use a tree-structured learner. Their complexity is often low, because they are themselves fitted by a stagewise selection of variables and split points, as in CART and other tree algorithms (whereas the complexity would be much higher, or it would be a much stronger learner, if the tree were fitted by a global search, which is most often infeasible). When the predictor space is high dimensional, the substantial amount of variable selection done by a tree procedure is particularly helpful in leading to a low-complexity (or weak) learner. This feature may be very desirable.

In this article, we mainly investigated the computationally simple L_2Boost by taking advantage of its analytical tractability, and we also demonstrated its practical effectiveness. Our findings can be summarized as follows:

1. L_2Boost is appropriate for both regression and classification. It leads to competitive performance also in classification relative to LogitBoost.
2. L_2Boost is a stagewise fitting procedure, with the iteration m acting as the smoothing or regularization parameter (as is also true for other boosting algorithms). In the linear learner case, m controls a new exponential bias–variance trade-off.
3. L_2Boost with smoothing splines results in optimal minimax rates of convergence in both regression and classification. Moreover, the algorithm adapts to unknown smoothness.
4. Boosting learners that involve only one (selected) predictor variable yields an additive model fit. We propose componentwise cubic smoothing splines of such type that we believe are often better learners than tree-structured stumps, especially for continuous predictors.
5. Weighting observations is *not* used for L_2Boost; we doubt that the success of general boosting algorithms is due to "giving large weights to heavily misclassified instances"

as Freund and Schapire (1996) conjectured for AdaBoost. Weighting in AdaBoost and LogitBoost comes as a consequence of the choice of the loss function and likely is not the reason for their successes. It is interesting to note here Breiman's (2000) conjecture that even in the case of AdaBoost, weighting is not the reason for success.

6. A simple expansion of the smoothed 0–1 loss reveals new insights into the classification problem, particularly an additional resistance of boosting against overfitting.

APPENDIX: PROOFS

Proof of Proposition 1

For L_2Boost with cost function $C(y,u)=(y-u)^2/2$, the negative gradient in stage j is the classical residual vector $u_j = Y-\widehat{F}_{j-1}$ and $\widehat{F}_j=\widehat{F}_{j-1}+\hat{f}_j$ (there is no need for a line search) with $\hat{f}_j=\mathcal{S}u_j$. Thus

$$u_j = Y-\widehat{F}_{j-1}=u_{j-1}-\mathcal{S}u_{j-1}=(I-\mathcal{S})u_{j-1},\quad j=1,2,\ldots,m,$$

implying that $u_j=(I-\mathcal{S})^jY$ for $j=1,2,\ldots,m$. Because $\widehat{F}_0=\mathcal{S}Y$ we obtain $\widehat{F}_m=\sum_{j=0}^m\mathcal{S}(I-\mathcal{S})^jY$. Using a telescope-sum argument, this equals $(I-(I-\mathcal{S})^{m+1})Y$.

Proof of Proposition 3

The bias term is

$$\begin{aligned}\mathrm{bias}^2(m,\mathcal{S};f)&=(\mathbb{E}[\mathcal{B}_mY]-f)^T(\mathbb{E}[\mathcal{B}_mY]-f)\\&=((\mathcal{B}_m-I)f)^T((\mathcal{B}_m-I)f).\end{aligned}$$

According to (7), using orthonormality of U,

$$\mathcal{B}_m-I=U(D_m-I)U^T=U\,\mathrm{diag}(-(1-\lambda_k)^{m+1})U^T.$$

Thus, again by orthonormality of U, the formula for the bias follows.

For the variance, consider

$$\begin{aligned}\mathrm{cov}(\mathcal{B}_mY)&=\mathcal{B}_m\,\mathrm{cov}(Y)\mathcal{B}_m^T=\sigma^2\mathcal{B}_m\mathcal{B}_m^T\\&=\sigma^2U\,\mathrm{diag}((1-(1-\lambda_k)^{m+1})^2)U^T,\end{aligned}$$

using (7) and orthonormality of U. Then

$$\mathrm{var}(m,\mathcal{S};\sigma^2)=tr[\mathrm{cov}(\mathcal{B}_mY)]=\sigma^2\sum_{k=1}^n(1-(1-\lambda_k)^{m+1})^2,$$

again using orthonormality of U.

Proof of Theorem 1

For assertion c, without loss of generality, assume that $\lambda_k<1$ for all k. If not, then restrict the summation to those k's that satisfy $\lambda_k<1$. Denote $\mu=U^Tf\in\mathbb{R}^n$. For $x\ge 0$, let

$$g(x)=n^{-1}\sum_{k=1}^n\mu_k^2(1-\lambda_k)^{2x+2}+\sigma^2n^{-1}\sum_{k=1}^n(1-(1-\lambda_k)^{x+1})^2.$$

Then

$$g(m)=MSE(m,\mathcal{S};f,\sigma^2)=\mathrm{bias}^2(m,\mathcal{S};f)+\mathrm{var}(m,\mathcal{S};\sigma^2).$$

It is easy to derive

$$\begin{aligned}g'(x)=2n^{-1}\sum_{k=1}^n&[(\mu_k^2+\sigma^2)(1-\lambda_k)^{x+1}-\sigma^2]\\&\times(1-\lambda_k)^{x+1}\log(1-\lambda_k).\end{aligned}$$

It follows that

$$g'(0)=2n^{-1}\sum_{k=1}^n[(\mu_k^2+\sigma^2)(1-\lambda_k)-\sigma^2](1-\lambda_k)\log(1-\lambda_k)$$

and

$$g'(1)=2n^{-1}\sum_{k=1}^n[(\mu_k^2+\sigma^2)(1-\lambda_k)^2-\sigma^2](1-\lambda_k)^2\log(1-\lambda_k),$$

both of which are negative under the inequality condition $\mu_k/\sigma^2>1/(1-\lambda_k)^2-1$; also, $g'(x)<0$ for $x\in(0,1)$ under this condition. Hence $g(1)<g(0)$, which means that boosting improves.

b. Rewrite

$$g(x)=n^{-1}\sum_{k=1}^n[(\mu_k^2+\sigma^2)(1-\lambda_k)^{x+1}-2\sigma^2](1-\lambda_k)^{x+1}+\sigma^2.$$

Because for all k (with $\lambda_k<1$), $(\mu_k^2+\sigma^2)(1-\lambda_k)^{x+1}-2\sigma^2\to-2\sigma^2$ as $x\to\infty$, there exists an m such that $(\mu_k^2+\sigma^2)(1-\lambda_k)^{m+1}-2\sigma^2\le-\sigma^2$ for all k (with $\lambda_k<1$). It follows that

$$g(m)\le -n^{-1}\sum_{k=1}^n(1-\lambda_k)^{m+1}\sigma^2+\sigma^2<\sigma^2.$$

It is obvious that $g(m)\to\sigma^2$ as $m\to\infty$.

Proof of Theorem 2

Write the summands with the higher-order moment as

$$\begin{aligned}&\mathbb{E}[(\widehat{F}_m(x_i)-f(x_i))^p]\\&\quad=\sum_{j=0}^p\binom{p}{j}b_m(x_i)^j\mathbb{E}[(\widehat{F}_m(x_i)-\mathbb{E}[\widehat{F}_m(x_i)])^{p-j}],\qquad\text{(A.1)}\end{aligned}$$

where $b_m(x_i)=\mathbb{E}[\widehat{F}_m(x_i)-f(x_i)]$ is the bias term. Thus we have transformed the higher-order moment as a sum of higher-order *centered* moments with the bias as a multiplier that goes to 0 as $m\to\infty$. The centered moments can be written as

$$\begin{aligned}(\widehat{F}_m(x_i)-\mathbb{E}[\widehat{F}_m(x_i)])^q&=(\mathcal{B}_mY-\mathbb{E}[\mathcal{B}_mY])_i^q\\&=(\mathcal{B}_m\varepsilon)_i^q=((I-(I-\mathcal{S})^{m+1})\varepsilon)_i^q\\&=(\varepsilon-(I-\mathcal{S})^{m+1}\varepsilon)_i^q,\end{aligned}$$

where we use Proposition 1. Because $(I-\mathcal{S})^{m+1}$ is a map (matrix) taking values exponentially close to 0 as $m\to\infty$, we obtain

$$\mathbb{E}[(\widehat{F}_m(x_i)-\mathbb{E}[\widehat{F}_m(x_i)])^q]=\mathbb{E}[\varepsilon_i^q]+O(\exp(-C_qm))\quad(m\to\infty)$$

for some constant $C_q>0$. From this last bound and using (A.1) together with the fact that the bias $b_m(x_i)=O(\exp(-C_bm))(m\to\infty)$ for some constant $C_b>0$ (see Theorem 1), we complete the proof.

Proof of Theorem 3

Let $\mathcal{S}$ be the smoothing spline operator corresponding to smoothness r and with smoothing parameter $c=\lambda_0$ (to avoid notational confusion with eigenvalues). It is well known (Utreras 1983; Wahba 1990, p. 61) that the eigenvalues of $\mathcal{S}$ take the form, in decreasing order,

$$\lambda_1=\cdots=\lambda_r=1,\qquad\lambda_k=\frac{nq_{k,n}}{n\lambda_0+nq_{k,n}}\quad\text{for}\quad k=r+1,\ldots,n.$$

Moreover, under assumption (A) and for n large, $q_{k,n}\approx Ak^{-2r}:=Aq_k$ $(0<A<\infty)$ (Utreras 1988). For the true function $f\in\mathcal{W}_2^{(\nu)}$,

$$\frac{1}{n}\sum_{k=r+1}^n\mu_k^2k^{2\nu}\le M<\infty.$$

Let $c_0 = c/A$; then

$$\lambda_k \approx \frac{q_k}{c_0+q_k} \quad \text{for} \quad k=r+1,\dots,n.$$

Then the bias term can be bounded as

$$\begin{aligned} \text{bias}^2(m,\mathcal{S};f) &= \frac{1}{n}\sum_{k=r+1}^{n} (1-\lambda_k)^{2m+2}\mu_k^2 \\ &\approx \frac{1}{n} \sum_{k=r+1}^{n} (1-q_k/(c_0+q_k))^{2m+2}k^{-2\nu}\mu_k^2k^{2\nu} \\ &\le \max_{k=r+1,\dots,n}(1-q_k/(c_0+q_k))^{2m+2}k^{-2\nu} \\ &\qquad \times \frac{1}{n}\sum_{k=r+1}^{n} \mu_k^2k^{2\nu} \\ &= \max_{k=r+1,\dots,n}\exp(h(k))\times\frac{1}{n}\sum_{k=r+1}^{n}\mu_k^2k^{2\nu}, \end{aligned}$$

where

$$\begin{aligned} h(x) &= \log[(1-x^{-2r}/(c_0+x^{-2r}))^{2m+2}x^{-2\nu}] \\ &= (2m+2)\log(1-1/(c_0x^{2r}+1)) - 2\nu\log(x). \end{aligned}$$

Taking the derivative yields

$$h'(x) = \frac{2r}{x}\frac{1}{c_0x^{2r}+1}\Big[(2m+2) - \frac{\nu}{r}(c_0x^{2r}+1)\Big].$$

Hence for any given positive integer n_1, if $x \le n_1$ and $m \ge \frac{\nu}{2r}(c_0n_1^{2r}+1)-1$, then $h(x)$ is increasing and so is $\exp(h(x))$, and

$$\exp(h(x)) \le \exp(h(n_1)) = (1-1/(c_0n_1^{2r}+1))^{2m+2}n_1^{-2\nu}.$$

On $[n_1+1, n]$,

$$\exp(h(x)) \le (1-1/(c_0n^{2r}+1))^{2m+2}n_1^{-2\nu}.$$

Putting them together, we get, for any given n_1 and $m \ge \frac{\nu}{2r}(c_0n_1^{2r}+1)-1$,

$$\text{bias}^2(m,\mathcal{S};f) \le Mn_1^{-2\nu}[2(1-1/(c_0n^{2r}+1))^{2m+2}],$$

which is of the order $O(n_1^{-2\nu})$ for $n_1 \to \infty$ and $n_1 \le n$.

We now deal with the variance term. For any $n_1 > r$,

$$\begin{aligned} \text{var}(m,\mathcal{S};\sigma^2) &= \frac{\sigma^2}{n}\Big\{r + \sum_{k=r+1}^{n}[1-(1-\lambda_k)^{m+1}]^2\Big\} \\ &\le \frac{\sigma^2n_1}{n} + \frac{\sigma^2}{n}\sum_{k=n_1+1}^{n}[1-(1-\lambda_k)^{m+1}]^2 := I_1 + I_2. \end{aligned}$$

Because $(1-x)^a \ge 1-ax$ for any $x \in [0,1]$ and $a \ge 1$,

$$1-(1-\lambda_k)^{m+1} \le 1-[1-(m+1)\lambda_k] = (m+1)\lambda_k.$$

It follows that

$$\begin{aligned} I_2 &\le \frac{\sigma^2}{n}\sum_{k=n_1+1}^{n}(m+1)^2\lambda_k^2 \approx \frac{\sigma^2(m+1)^2}{n}\sum_{k=n_1+1}^{n}\frac{1}{(c_0k^{2r}+2)^2} \\ &\le \frac{\sigma^2(m+1)^2}{n}\sum_{k=n_1+1}^{n}\frac{1}{(c_0k^{2r})^2} \le \frac{\sigma^2(m+1)^2}{n}\int_{n_1}^{\infty}\frac{1}{(c_0x^{2r})^2}\,dx \\ &= \frac{\sigma^2(m+1)^2}{c_0^2(4r-1)n}n_1/n_1^{4r} \le O(n_1/n) \end{aligned}$$

if we take $m = m(n_1) = \frac{\nu}{2r}(c_0n_1^{2r}+1)-1 = O(n_1^{2r})$. Hence, for this choice of $m(n_1)$,

$$\text{var}(m(n_1),\mathcal{S};\sigma^2) \le O(n_1/n).$$

Together with the bound for the bias, we get

$$\frac{1}{n}MSE \le O(n_1/n) + O(n_1^{-2\nu}),$$

which is minimized by taking $n_1 = O(n^{1/(2\nu+1)})$ and for $m(n) = m(n_1) = O(n^{2r/(2\nu+1)})$. The minimized MSE has the minimax optimal rate $O(n^{-2\nu/(2\nu+1)})$ of the smoother function class $\mathcal{W}_2^{(\nu)}$.

Proof of Proposition 4

Denote $C(z) = \mathbf{1}_{[z<0]}$. We first show that

$$\sup_{|z|>\gamma\log(\gamma^{-1})}|C(z)-C_\gamma(z)| = \gamma/2. \tag{A.2}$$

By symmetry, it suffices to consider $z > 0$,

$$\begin{aligned} \sup_{z>\gamma\log(\gamma^{-1})}|C(z)-C_\gamma(z)| &= C_\gamma(\gamma\log(\gamma^{-1})) \\ &= \exp(-\log(\gamma^{-1}))/2 = \gamma/2, \end{aligned}$$

proving (A.2).

On the other hand,

$$\begin{aligned} &\int_{|z|\le\gamma\log(\gamma^{-1})}|C(z)-C_\gamma(z)|g(z)\,dz \\ &\quad\le \sup_{|z|\le\gamma\log(\gamma^{-1})}|g(z)|\gamma\log(\gamma^{-1}) = O(\gamma\log(\gamma^{-1})). \end{aligned} \tag{A.3}$$

Hence, by (A.2) and (A.3),

$$\begin{aligned} &|\mathbb{E}[C(Z)-C_\gamma(Z)]| \\ &\quad\le \bigg(\int_{|z|>\gamma\log(\gamma^{-1})} + \int_{|z|\le\gamma\log(\gamma^{-1})}\bigg)|C(z)-C_\gamma(z)|g(z)\,dz \\ &\quad\le \gamma/2 + O(\gamma\log(\gamma^{-1})) \\ &\quad= O(\gamma\log(\gamma^{-1})) \quad (\gamma \to 0). \end{aligned}$$

[Received August 2001. Revised June 2002.]

REFERENCES

Allwein, E., Schapire, R., and Singer, Y. (2001), "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *Journal of Machine Learning Research*, 1, 113–141.

Breiman, L. (1998), "Arcing Classifiers," *The Annals of Statistics*, 26, 801–824.

——— (1999), "Prediction Games and Arcing Algorithms," *Neural Computation*, 11, 1493–1517.

——— (2000), "Some Infinity Theory for Predictor Ensembles," Technical Report 579, University of California, Berkeley, Dept. of Statistics.

Buja, A. (2000), Comment on "Additive Logistic Regression: A Statistical View of Boosting," by J. H. Friedman, T. Hastie, and R. Tibshirani, *The Annals of Statistics,* 28, 387–391.

Collins, M., Schapire, R. E., and Singer, Y. (2000), "Logistic Regression, AdaBoost, and Bregman Distances," in *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*.

Devroye, L., Györfi, L., and Lugosi, G. (1996), *A Probabilistic Theory of Pattern Recognition*, New York: Springer.

Dietterich, T. G. (2000), "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization," *Machine Learning*, 40, 139–157.

Freund, Y. (1995), "Boosting a Weak Learning Algorithm by Majority," *Information and Computation*, 121, 256–285.

Freund, Y., and Schapire, R. E. (1996), "Experiments With a New Boosting Algorithm," in *Machine Learning: Proceedings of the Thirteenth International Conference*, San Francisco: Morgan Kauffman, pp. 148–156.

Friedman, J. H. (2001), "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, 29, 1189–1232.

Friedman, J. H., Hastie, T., and Tibshirani, R. (2000), "Additive Logistic Regression: A Statistical View of Boosting" (with discussion), *The Annals of Statistics*, 28, 337–407.

Geman, S., Bienenstock, E., and Doursat, R. (1992), "Neural Networks and the Bias/Variance Dilemma," *Neural Computations*, 4, 1–58.

Gu, C. (1987), "What Happens When Bootstrapping the Smoothing Spline?," *Communications in Statistics, Part A—Theory and Methods*, 16, 3275–3284.

Hastie, T. J., and Tibshirani, R. J. (1990), *Generalized Additive Models*, London: Chapman and Hall.

Jiang, W. (2003), "Process Consistency for AdaBoost," *The Annals of Statistics*, to appear.

Loader, C. (1999), *Local Regression and Likelihood*, New York: Springer.

Mammen, E., and Tsybakov, A. B. (1999), "Smooth Discriminant Analysis," *The Annals of Statistics*, 27, 1808–1829.

Marron, J. S. (1983), "Optimal Rates of Convergence to Bayes Risk in Nonparametric Discrimination," *The Annals of Statistics*, 11, 1142–1155.

Mason, L., Baxter, J., Bartlett, P., and Frean, M. (2000), "Functional Gradient Techniques for Combining Hypotheses," in *Advances in Large Margin Classifiers*, eds. A. J. Smola, P. J. Bartlett, B. Schölkopf, and D. Schuurmans, Cambridge, MA: MIT Press.

Schapire, R. E. (1990), "The Strength of Weak Learnability," *Machine Learning*, 5, 197–227.

Schapire, R. E., Freund, Y., Bartlett, P., and Lee, W. S. (1998), "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," *The Annals of Statistics*, 26, 1651–1686.

Tukey, J. W. (1977), *Exploratory Data Analysis*, Reading, MA: Addison-Wesley.

Utreras, F. (1983), "Natural Spline Functions, Their Associated Eigenvalue Problem," *Numer. Math.*, 42, 107–117.

——— (1988), "Convergence Rates for Multivariate Smoothing Spline Functions," *Journal of Approximation Theory*, 52, 1–27.

Wahba, G. (1990), *Spline Models for Observational Data*, Philadelphia: Society for Industrial and Applied Mathematics.

Yang, Y. (1999), "Minimax Nonparametric Classification—Part I: Rates of Convergence," *IEEE Transactions on Information Theory*, 45, 2271–2284.