Boosting with the L_2-Loss: Regression and Classification

Peter Bühlmann
ETH Zürich

Bin Yu
University of California, Berkeley

August 2001

Abstract

This paper investigates a variant of boosting, L_2Boost, which is constructed from a functional gradient descent algorithm with the L_2-loss function. Based on an explicit stagewise refitting expression of L_2Boost, the case of (symmetric) linear weak learners is studied in detail in both regression and two-class 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 bounded as m tends to infinity. When the weak learner is a smoothing spline, an optimal rate of convergence result holds for both regression and two-class classification. And this boosted smoothing spline adapts to higher order, unknown smoothness. Moreover, a simple expansion of the 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 data set results are obtained to demonstrate the attractiveness of L_2Boost, particularly with a novel component-wise cubic smoothing spline as an effective and practical weak learner.

1 Introduction

Boosting is one of the most successful and practical methods that 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 large data sets. The improved performance of a weak learner through boosting has been impressive, and seems to be associated with boosting's resistance to overfitting. The burning question is why.

The rationale behind boosting separates itself from the traditional statistical procedures. It starts with a sensible estimator or classifier, the weak learner, and seeks its improvements iteratively based on its performance on the training data set. The possibility of this boosting procedure comes with the availability of large data sets 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 data set problems, a good model for the problem is hard to come by, but a sensible procedure is not. And this may explain the empirical success of boosting on large data sets. After much work on bounding the generalization error of a boosted procedure, via the VC dimensions and the distribution of margins (Schapire et al., 1998), some recent developments on boosting have been on the gradient-descent (GD) view of boosting. They are the results of efforts of many researchers (Breiman, 1999; Mason et al., 1999; Friedman et al., 2000; Collins et al., 2000). This GD view connects boosting to the more common optimization view of statistical inference, and its most obvious consequence has been the emergence of many variants of the original AdaBoost, under various loss or objective functions (Mason et al., 1999; Friedman et al., 2000; Friedman, 2001). Even though a satisfactory explanation on why boosting works does not follow directly from this GD view, some of the new boosting variants are more easily accessible for analysis. In this paper, we take advantage of this new analytic possibility on L_2-boosting procedures to build our case for understanding boosting both in regression and two-class classification. It is worth pointing out that L_2Boost is studied here also as a procedure yielding competitive results in regression and even two-class classification, in addition to its analytical tractability.

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

(i) 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 due to the dependence of this new term on the previous terms, the "complexity" of the fitted procedure is not increased by a constant amount as we got used to in linear regression, but 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.

(ii) When the weak 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.

Item (i) partially explains the "overfitting-resistance" mystery of boosting. The phenomenon is radically different from the well-known algebraic bias-variance trade-off in nonparametric regression. Item (ii) shows an interesting result about boosting in adaptive estimation: even when smoothness is unknown, L_2Boost achieves the optimal (minimax) rate of convergence.

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

Section 5 deals with the two-class classification problem. The optimality in item (ii) above 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. Furthermore, we approximate the 0-1 loss function via a smoothed version to show that the 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 get 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 regression and why previous attempts were not successful at decomposing the generalization error into additive bias and variance terms (cf. Geman et al. 1992; Breiman, 1998, and references therein).

We support the theory and explanations by simulated and real data sets which demonstrate the attractiveness of L_2Boost, particularly in the presence of very high-dimensional predictors and with a novel component-wise cubic spline as an effective weak learner. Finally, Section 6 contains a discussion on the role of the weak learner and a summary of the paper.

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

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

based on data (Y_i, X_i) $(i = 1, \ldots n)$. We consider here both cases where the response Y is continuous (regression problem) or discrete (classification problem), since boosting is potentially useful in both cases; X denotes here a d-dimensional covariable. 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:

$$\begin{aligned}
&C(y, f) = \exp(yf) \text{ with } y \in \{-1, 1\}\text{: AdaBoost cost function,}\\
&C(y, f) = \log_2(1 + \exp(-2yf)) \text{ with } y \in \{-1, 1\}\text{: LogitBoost cost function,}\\
&C(y, f) = (y - f)^2/2 \text{ with } y \in \mathbb{R} \text{ or } \in \{-1, 1\}\text{: } L_2\text{Boost cost function.}
\end{aligned} \tag{2}$$

The population minimizers of (1) are then

$$\begin{aligned}
&F(x) = \frac{1}{2}\log\left(\frac{\mathbb{P}[Y = 1|X = x]}{\mathbb{P}[Y = -1|X = x]}\right) \text{ for AdaBoost and LogitBoost cost,}\\
&F(x) = \mathbb{E}[Y|X = x] \text{ for } L_2\text{Boost cost.}
\end{aligned} \tag{3}$$

Estimation of such an $F(\cdot)$ from data can be done by functional gradient descent which is a constrained minimization of the empirical risk

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

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

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

where $\hat{\theta}$ is an estimated finite or infinite-dimensional parameter. For example, the weak learner $h(\cdot, \hat{\theta})$ could be a decision tree where $\hat{\theta}$ describes the axis to be split, the split points and the location parameter in terminal nodes. How to fit $h(x, \theta)$ from data is part of the weak learner and can be done according to a base algorithm. For example, least squares fitting yields

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

for some data $(U, X) = \{U_i, X_i; i = 1, \ldots, n\}$. The general description of FGD is as follows (cf. Friedman, 2001).

Generic functional gradient descent

Step 1 (initialization). Given data $\{Y_i, X_i; i = 1, \ldots, n\}$, fit an initial weak learner

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

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

Step 2 (projecting gradient to weak learner). Compute the negative gradient vector

$$U_i = -\frac{\partial C(Y_i, F)}{\partial F}|_{F=\hat{F}_m(X_i)},\ i = 1, \ldots, n,$$

evaluated at the current $\hat{F}_m(\cdot)$. Then, fit the weak learner to the gradient vector

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

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

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

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

Update,

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

Step 4 (iteration). Increase m by one and repeat Steps 2 and 3.

We call $\hat{F}_m(\cdot)$ the AdaBoost-, LogitBoost- or L_2Boost-estimate, according to the implementing cost function in (2). Note that L_2Boost has a simple structure: the negative gradient in Step 2 is the classical residual vector $U_i = Y_i - \hat{F}_m(X_i)$ $(i = 1, \ldots, n)$ and the line search in Step 3 is trivial with $\hat{w}_{m+1} = 1$. L_2Boosting is thus nothing else than repeated least squares fitting of residuals (cf. Friedman, 2001). With $m = 1$ (one boosting step), it has already been proposed by Tukey (1977) under the name "twicing".

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

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

since $\mathbb{E}[Y|X = x] = \mathbb{P}[Y = 1|X = x] - \mathbb{P}[Y = -1|X = x]$. 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. (1999) and Collins et al. (2000) describe when boosting-type algorithms, i.e. functional gradient descent, converge numerically. This tells us that, under certain conditions, the test set or generalization error for boosting eventually stabilizes. But it doesn't imply that the eventually stable solution is the best, or that overfitting could happen long before reaching convergence. Indeed, we will show in Section 3 that L_2Boost with "contracting" linear learners converges to the fully saturated model, i.e. $\hat{F}_\infty(X_i) = Y_i$ for all $i = 1, \ldots, n$, fitting the data perfectly.

3 Theory for L_2Boosting 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 Friedman et al. (2000) paper. He made amply clear there that this stagewise fitting had gotten a bad reputation among statisticians and didn't get the attention it deserved. The success of boosting definitely serves as an eye-opener for us to take a fresh look at stagewise fitting.

Consider the regression model

$$\begin{aligned} &Y_i = f(x_i) + \varepsilon_i,\ i = 1, \ldots, n, \\ &\varepsilon_1, \ldots, \varepsilon_n \text{ i.i.d. with } \mathbb{E}[\varepsilon_i] = 0,\ \mathrm{Var}(\varepsilon_i) = \sigma^2, \end{aligned} \tag{6}$$

where $f(\cdot)$ is a real-valued, typically nonlinear function, and the covariables $x_i \in \mathbb{R}^d$ are deterministic (e.g. conditioning on the design). Represent a weak learner 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 covariables $x_1, \ldots, x_n$ are absorbed in the operator notation $\mathcal{S}$. In the sequel, we often use the notation Y for the vector $(Y_1, \ldots, Y_n)^T$, F_j for the vector $(F_j(x_1), \ldots, F_j(x_n))^T$ and analogously for f_j; it should always become clear from the context whether we mean a single variable Y or function $F_j(\cdot)$, or the vectors as above.

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

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

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

$$\mathcal{B}_m Y = \hat{F}_m.$$

According to Proposition 1, there is a relatively direct link between the boosting operator and the weak learner $\mathcal{S}$. We exploit this in the sequel.

We focus here on linear weak 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, or smoothing operators such as kernel and smoothing spline estimators.

Proposition 2. *Consider a linear weak learner $\mathcal{S}$ with eigenvalues $\{\lambda_k;\ k = 1, \ldots, n\}$, based on deterministic covariables $x_1, \ldots, x_n$. 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 will become even more transparent when specializing to the case where $\mathcal{S}$ 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 kernel with 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,

$$\begin{aligned} &\mathcal{B}_m = U D_m U^T,\ D_m = \mathrm{diag}(1 - (1 - \lambda_k)^{m+1}), \\ &k\text{th column-vector of } U \text{ being the } k\text{th eigenvector of } \mathcal{S} \text{ to the eigenvalue } \lambda_k. \end{aligned} \tag{7}$$

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 (expected) mean squared error

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

which averages over the observed covariables. Note that if the design is stochastic with a probability distribution, the MSE measure above is asymptotically equivalent, as $n \to \infty$, to the generalization error $\mathbb{E}[(\hat{F}_m(X) - f(X))^2]$, 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 set. We show in Figure 1 the difference between the two measures for a finite sample case.

Theorem 1. *Consider a linear, symmetric weak learner $\mathcal{S}$ 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 bias, variance and averaged mean squared error for L_2Boost are*

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

A proof is given in the Appendix. Theorem 1 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 and 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 $0 < \lambda_k \leq 1$. An important example for such a linear weak learner are cubic smoothing splines which have two eigenvalues equal to one and all others strictly between zero and one: this will be treated even in more detail in Section 3.1.

Theorem 2. *Under the assumptions in Theorem 1 with $0 < \lambda_k \leq 1,\ k = 1, \ldots, n$,*

(1) $bias(m; \mathcal{S}; f)$ decays exponentially fast with increasing m,
$variance(m, \mathcal{S}; \sigma^2)$ exhibits exponentially small increase with increasing m,
$\lim_{m \to \infty} MSE(m, \mathcal{S}; f, \sigma^2) = \sigma^2$.

(2) Moreover, let $\mu = U^T f = (\mu_1, ..., \mu_n)^T$ be the function vector in the linear space spanned by column vectors of U.
(i) 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 $\mathcal{S}$.
(ii) 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.

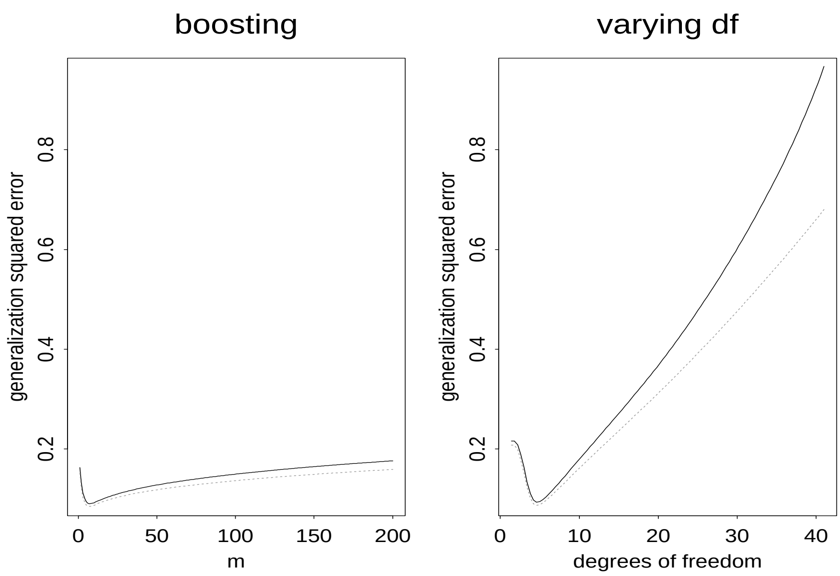

Figure 1: Generalization mean squared error $\mathbb{E}[(Y-\hat{f}(X))^2]$ (solid line) and MSE criterion from (8) (dotted line) from 100 simulations of model (11) with design uniformly distributed on $[-1/2, 1/2]$, each with $n = 100$. Left: L_2Boost with cubic smoothing spline having df=3, as a function of boosting iterations m. Right: Cubic smoothing spline for various degrees of freedom (various amount of smoothing).

Assertion (1) is a direct consequence of Theorem 1. A proof of assertion (2) is given in the Appendix.

Theorem 2 comes as a surprise: it shows a very interesting bias-variance trade-off that hasn't 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 3.1.1). 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 afterwards, due to the exponential increase and decrease in the bias and variance terms. Condition $\mu_k^2/\sigma^2 > 1/(1-\lambda_k)^2 - 1$ in part 2(i) can be interpreted as that f is relatively complex compared to the linear learner in the k-th component direction. For example, a large right-hand side implies λ_k close to 1, that is, the learner employs very little shrinkage or smoothing in the kth direction or the learner is actually strong in the kth direction. Then for boosting to bring improvement, μ_k has to be large relative to the noise level σ or the function f has to be complex in that direction. When the right hand side of the condition is small, that is, the learner does much smoothing (shrinkage) or is weak, the condition is more easily satisfied. Hence we see improvements for boosting with weak learners most of the time. The assertion in 2(ii) shows that boosting always beats the unbiased estimator Y as does James-Stein estimator in the space spanned by U. In this space, the original linear estimator is a component-wise shrinkage estimator, while the James-Stein estimator uses a uniform shrinkage.

The phenomenon in Theorem 2 generalizes qualitatively to higher order moments.

Theorem 3. *Under the assumptions in Theorem 1 with $0 < \lambda_k \le 1,\ k = 1, \dots, n$ and assuming $\mathbb{E}|\varepsilon_1|^p < \infty$ for $p \in \mathbb{N}$,*

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

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

A proof is given in the Appendix. Theorem 3 will be used later to argue that the expected 0-1 loss in classification also exhibits only exponentially small amount of overfitting as boosting iterations $m \to \infty$.

Boosting until theoretical convergence with $m = \infty$ is typically not a good advice: 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 which fits the data perfectly. Therefore, one should monitor an estimate of MSE, for example by using a test set or cross-validation.

Finally, boosting sometimes doesn't change the procedure at all.

Corollary 1. *Under the assumptions in Theorem 1 with $\lambda_k \in \{0, 1\},\ k = 1, \dots, n$, or equivalently, $\mathcal{S}$ is a linear projection operator, $\mathcal{B}_m \equiv \mathcal{S}$ for $m = 1, 2, \dots$.*

3.1 Smoothing splines as weak learners

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, defined on an interval $[a, b]$, as

$$\mathcal{F}^{(\nu)} = \{f : \int_a^b [f^{(\nu)}(x)]^2 dx < \infty\}. \tag{9}$$

Let $\mathcal{S}Y = g_r$ be the smoothing spline solution to the penalized least squares problem

$$g_r = g_r(\lambda) = \text{argmin}_{f \in \mathcal{F}^{(r)}} \frac{1}{n} \sum_i [Y_i - f(x_i)]^2 + \lambda \int [f^{(r)}(x)]^2 dx \tag{10}$$

Theorem 4. *(Optimality of L_2Boost for smoothing splines). Suppose $\mathcal{S}$ 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 $\mathcal{F}^{(\nu)}$ 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 $\hat{F}_{m(n)}$ achieves the optimal minimax rate $n^{-2\nu/(2\nu+1)}$ of the smoother function class $\mathcal{F}^{(\nu)}$ in terms of MSE.*

A proof is given in the Appendix. This result states that first, boosting smoothing splines is minimax optimal for a given smoothness class and second, boosting also adapts to higher order smoothness. Gu (1987) analyzes twicing ($m = 1$) and shows that twicing can adapt to a higher order smoothness $\nu \le 2r$. With boosting we can adapt to an arbitrarily higher order smoothness since we can refit as many times as we want. For cubic smoothing spline learners with $r = 2$, the optimal rate $n^{-4/5}$ 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 optimal boosting iteration m is selected either through a fixed test set or, more often, through cross validation based on many random splits of the original data set.

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. The former is algebraic and the latter is exponential. The advantage of the new exponential trade-off is the flatter near-optimal region for the optimal smoothing parameter (or boosting iteration) and bounded complexity (variance) for all smoothing parameter values or boosting iterations. An example was shown in Figure 1 with the simulated data from the next Section.

3.1.1 Simulation results with cubic smoothing spline as weak learners

The relevance of the theoretical results above depends on the underlying problem and the sample size. We consider a representative example for the model in (6),

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

The weak learner $\mathcal{S}$ is chosen as a cubic smoothing spline which satisfies linearity, symmetry and the eigenvalue-conditions used in Theorems 1–3.

The complexity of $\mathcal{S}$, or the strength of the weak leaner, 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 weak 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, we use shrinkage (Friedman, 2001) to replace $\mathcal{S}$ by

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

For $\mathcal{S}$ is 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 zero than $\{\lambda_k\}_k$ for the original $\mathcal{S}$. With small ν, we thus get a weaker learner than the original $\mathcal{S}$: shrinkage acts here similarly as changing the degrees of freedom of the original $\mathcal{S}$ to a lower value. We will see its effect in more complex examples in Sections 4.2 and 6. 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 the estimators rising from different shrinkage factors and different iterations of boosting). Figure 2 displays MSE from specification (11) with $x_1, \ldots, x_n$ i.i.d. realizations from $\mathcal{N}(0,1)$, as a function of m and ν. It shows that the boosting question from above 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 4.

(2) Provided that the learner is sufficiently weak, boosting always improves, as we show in Theorem 2.

(3) The initial $\mathcal{S}$ with 20 degrees of freedom is too strong (too small amount of smoothing) and boosting decreases performance due to overfitting.

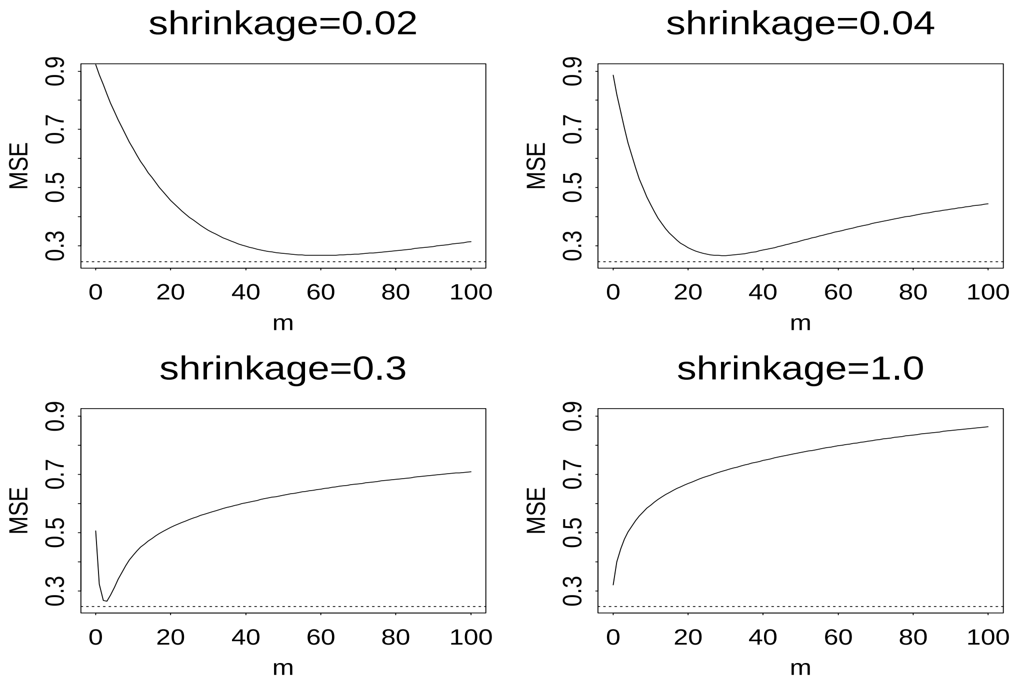

Figure 2: Traces of MSE as a function of boosting iterations m, for four different shrinkage factors. Dotted line represents minimum, achieved with $m = 0, \nu = 0.76$. The data is from model (11) with sample size $n = 100$.

(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.

Statements (2)–(4), numerically found for this example with a linear weak learner, have also been shown to hold empirically for many data sets and with nonlinear learners in AdaBoost and LogitBoost in classification. Statement (1), dealing with optimality, is asymptotically explained by our Theorem 4. Also the recent work of Jiang (2000) hints at a consistency result for AdaBoost: he shows that in the asymptotic sense and for classification, AdaBoost visits nearly-optimal (nearly-Bayes) procedures during the evolution of AdaBoost. However, this asymptotic consistency result doesn't explain optimality or AdaBoost's good finite-sample performance.

All these behaviors happen well before the asymptopia $m = \infty$ or the convergence of the boosting algorithm. 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 upper bound, 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. For $\nu = 0.02$, we would need $m = 1691$ and the MSE at stopping is 1.146. This is still far from the stabilizing MSE ($m = \infty$) which is $\sigma^2 = 2$, see Theorem 2. The positive aspect is that we would stop before reaching the extreme overfitting situation. This little illustration describes impressively how slow convergence in boosting could be.

We also illustrate here the adaptivity of L_2Boost with smoothing splines to higher smoothness, see Theorem 4. We use simulated data from model (11) with $x_1, \ldots, x_n$ i.i.d. realizations from Uniform($[-1/2, 1/2]$), for sample sizes $n = 10$ up to 1000. Table 1 reports

sample size n	optimal smoothing spline	optimal L_2Boost	gain
10	$7.787 \cdot 10^{-1}$	$9.968 \cdot 10^{-1}$	-28.0%
25	$3.338 \cdot 10^{-1}$	$3.349 \cdot 10^{-1}$	-0.3%
50	$1.657 \cdot 10^{-1}$	$1.669 \cdot 10^{-1}$	-0.7%
100	$9.332 \cdot 10^{-2}$	$9.050 \cdot 10^{-2}$	0.9%
1000	$1.285 \cdot 10^{-2}$	$1.128 \cdot 10^{-2}$	12.2%

Table 1: Generalization mean squared error for simulated data from (11). 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 mean squared error with L_2Boost, indicates an advantage for boosting.

on the performance of the best cubic smoothing spline (with optimal penalty parameter) in comparison to L_2Boosting a cubic smoothing spline with "fixed" penalty $\lambda = c$, a constant (with optimal number of boosting iterations). We evaluate generalization error $\mathbb{E}[(f(X) - \hat{f}(X))^2]$ by averaging over 100 simulations from the model (11). We observe in Table 1 the following. 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 mid-range with $25 \leq n \leq 100$, the differences between optimal smoothing spline and optimal L_2Boost are negligible. For the large sample size $n = 1000$, we see an advantage of L_2Boost which is consistent with the theory: the underlying regression function in (11) is infinitely smooth and L_2Boost adapts to higher order smoothness.

4 L_2Boosting 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 which is doing some sort of variable selection, and the most prominent examples are trees.

4.1 Component-wise smoothing spline as weak learner

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

$$\hat{\iota} = \mathrm{argmin}_{\iota} \sum_{i=1}^{n} (Y_i - \hat{g}(X_{i,\iota}))^2,$$

where $\hat{g}$ is the smoothing spline as defined in (10). Thus, the component-wise smoothing spline learner is given by the function

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

Boosting stumps and component-wise smoothing splines yields an additive model whose terms are fitted in a stagewise fashion. The reason being that an additive combination of a stump or a component-wise smoothing spline $\hat{F}_0 + \sum_{m=1}^{M} \hat{f}_m$, with $\hat{f}_m(x)$

depending functionally only on $x_{\hat{\iota}}$ for some component $\hat{\iota} \in \{1, \ldots, d\}$, can be re-expressed as

$$\sum_{j=1}^{d} \hat{m}_j(x_j), \; x \in \mathbb{R}^d.$$

The estimated functions $\hat{m}_j(\cdot)$ when using boosting are fitted in stagewise fashion and different from the backfitting estimates in additive models (cf. Hastie and Tibshirani, 1990). Boosting stumps or component-wise 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 will show in Section 4.2 that when the dimension d is large, boosting also outperforms the alternative classical additive modeling with variable selection, using backfitting.

4.2 Numerical results

We first consider a real data set about ozone concentration in the Los Angeles basin which has been analyzed in Breiman (1998). The dimension of the predictor space is $d = 8$ and sample size is $n = 330$. We compare here L_2Boosting with classical additive models using backfitting and with MARS. L_2Boost is used with stumps and with component-wise cubic smoothing splines having 5 degrees of freedom (cf. Hastie and Tibshirani, 1990); additive model-backfitting is with smoothing splines using the default in S-Plus; MARS is run by using the default parameters as implemented in S-plus, library(mda). We estimate generalization mean squared error error $\mathbb{E}[(Y - \hat{F}(X))^2]$ with $\hat{F}(x) = \hat{\mathbb{E}}[Y|X = x]$ by randomly splitting the data in 297 training and 33 test observations and averaging 50 times over such random partitions. Table 2 displays the results. We conclude that L_2Boost with

method	mean squared error
L_2Boost with component-wise spline	17.50 (5)
L_2Boost with stumps	20.96 (26)
additive model (backfitted)	17.41
MARS	18.09

Table 2: Test set mean squared errors for ozone data. L_2Boost with optimal number of boosting iterations, given in parentheses. The component-wise spline is a cubic smoothing spline with df = 5.

component-wise splines is better than with trees and that it is among the best, together with classical backfitting of additive models.

Next, we show a simulated example in very high dimensions 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,

$$
\begin{aligned}
&Y = \sum_{j=1}^{100} \left(1 + (-1)^j A_j X_j + B_j \sin(6X_j)\right) \sum_{j=1}^{50} (1 + X_j/50) + \varepsilon,\\
&A_1, \ldots, A_{100} \text{ i.i.d. Unif}([0.6, 1]) \text{ and}\\
&B_1, \ldots, B_{100} \text{ i.i.d. Unif}([0.8, 1.2]), \text{ independent from the } A_j\text{'s},\\
&X \sim \text{Unif}([0,1]^{100}) \text{ where all components are i.i.d.} \sim \text{Unif}([0,1]),\\
&\varepsilon \sim \mathcal{N}(0, 2).
\end{aligned}
\tag{12}
$$

Samples of size $n = 200$ are generated by i.i.d. simulation of the pairs (Y_i, X_i) from model (12).

We use the same methods as above. However, we use classical additive modeling with a forward variable selection (inclusion) strategy because $d = 100$ is very large compared to $n = 200$. We evaluate $\mathbb{E}[(\hat{F}(X) - \mathbb{E}[Y|X = x])^2]$ at the true conditional expectation which can be done in simulations; note that $\mathbb{E}[Y|X = x]$ is still random depending on the realizations of the random coefficients A_j, B_j. Already a stump appeared to be too strong for L_2Boost and we therefore used shrunken learners $\nu\mathcal{S}$ with $\nu = 0.5$ chosen ad-hoc; we also allow the non-boosting procedures to be shrunken with $\nu = 0.5$. Table 4.2 shows the average performance over 10 simulations from model (12). L_2Boost is the

method	mean squared error
L_2Boost with shrunken component-wise 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)
MARS	25.19
shrunken MARS	15.05

Table 3: Generalization mean squared errors for simulated data from (12) with $n = 200$. L_2Boost with optimal number of boosting iterations, given in parentheses; additive model with optimal number of selected variables, given in parentheses. The component-wise spline is a cubic smoothing spline with df = 5; shrinkage factor is always $\nu = 0.5$.

winner over additive models and MARS, and the component-wise spline is a better learner than stumps. It is mainly in such high-dimensional situations where boosting has a clear advantage over other flexible nonparametric methods; and not so much in examples where the dimension p is "mid-range" relative to sample size.

5 Boosting in two-class problem

For classification, the evaluating performance criterion of the boosting procedure is often its misclassification rate. Consider a training sample

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

and a new test observation $(Y, X) \in \{-1, 1\} \times \mathbb{R}^d$, being independent from the training sample but having the same distribution. Using classifiers of the form (5), the misclassification rate can be expressed in terms of the (estimated) margin $Y\hat{F}_m$,

$$\mathbb{P}[Y\hat{F}_m(X) < 0] = \mathbb{E}[\mathbf{1}_{[Y\hat{F}_m(X)<0]}].$$

As usual, $\mathbb{P}$ and $\mathbb{E}$ are over all the random variables in the training set (13) and the testing observation, unless specified otherwise.

Insights about the expected zero-one loss function can be gained by approximating it, for theoretical purposes, with a smoothed version

$$\mathbb{E}[C_\gamma(Y\hat{F}_m(X))],$$
$$C_\gamma(z) = (1 - \frac{\exp(z/\gamma)}{2})\mathbf{1}_{[z<0]} + \frac{\exp(-z/\gamma)}{2}\mathbf{1}_{[z\geq 0]},\ \gamma > 0.$$

The parameter γ controls the quality of approximation.

Proposition 3. *Assume that the joint distribution of $Z = Y\hat{F}_m(X)$ has a density $g(z)$ which is bounded for z in an neighborhood around zero. Then,*

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

A proof is given in the Appendix. Proposition 3 shows that the misclassification rate can be approximated by an expected cost function which is infinitely often differentiable.

5.1 Misclassification rate via tapered moments of the margin

Proposition 3 motivates to study misclassification rate through $\mathbb{E}[C_\gamma(Z)]$ with $Z = Y\hat{F}_m(X)$. Applying a Taylor series expansion of $C_\gamma(\cdot)$ around $Z^* = YF(X)$, i.e. the margin with the true $F(\cdot)$, we obtain

$$\mathbb{E}[C_\gamma(Z)] = \mathbb{E}[C_\gamma(Z^*)] + \sum_{k=1}^{\infty} \frac{1}{k!}\mathbb{E}[C_\gamma^{(k)}(Z^*)(Z - Z^*)^k]. \tag{14}$$

The derivatives of $C_\gamma(\cdot)$ are

$$C_\gamma^{(k)}(z) = \frac{1}{\gamma^k}\exp(\frac{-|z|}{\gamma})(-\mathbf{1}_{[z<0]} + (-1)^k\mathbf{1}_{[z\geq 0]}). \tag{15}$$

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

$$\mathbb{E}[C_\gamma^{(k)}(Z^*)(Z - Z^*)^k] = \sum_{y\in\{-1,1\}} \int C_\gamma^{(k)}(yF(x))y^k b_k(x)\mathbb{P}[Y = y|X = x]dP_X(x),$$
$$b_k(x) = \mathbb{E}[(\hat{F}_m(x) - F(x))^k],\ \text{where expectation is over the training set in (13).} \tag{16}$$

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

From (14) and (16) we see that the generalization error of any procedure is approximately, in addition to the approximate Bayes risk $\mathbb{E}[C_\gamma Z^*]$, the sum of moments $b_k(x)$, tapered by $C_\gamma^{(k)}(yF(x))/k!$. Given a small value of γ (to ensure good approximation in Proposition 3), the tapering weights decay very quickly as $yF(x)$ moves away from zero. 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 misclassification rate. It may then be difficult to observe overfitting on test data, because most of the test data would not have many explanatories x near the decision boundary and therefore, the overfitting is hardly visible.

The first two terms in the approximation (14) are the tapered bias- and the tapered L_2-term, see (16). The higher order terms can be expanded as terms of interactions between the centered moments and the bias term (all tapered),

$$b_k(x) = \mathbb{E}[(\hat{F}_m(x) - F(x))^k] = \sum_{j=0}^{k} \binom{k}{j} b_1(x)^k \mathbb{E}[(\hat{F}_m(x) - \mathbb{E}[\hat{F}_m(x)])^{k-j}]. \tag{17}$$

This seemingly trivial approximation has three important consequences. The first is that bias (after tapering) as the first term in (14) and multiplicative terms in higher moments, see (17), plays a bigger role in 0-1 loss classification than in L_2-regression. Second, in the case of boosting, since all the (tapered) centered moment terms in (16) are bounded by expressions with exponentially diminishing increments as boosting iterations m gets large (see Section 3, particularly Theorem 3), driving the tapered bias to zero with many boosting iterations (exponentially fast) has a very mild increasing effect on the other terms in the summation (they are all exponentially small), and it is thus very worthwhile for lowering the generalization error. The third consequence of the approximation in (14), (16) and (17) is to explain why the previous attempts were not successful at decomposing the generalization error into additive bias and variance terms (cf. Geman et al. 1992; Breiman, 1998, and references therein). This, because except for the first two terms, all other important terms include the bias-term also in a multiplicative fashion (see (17)) for each term in the summation (14), instead of a pure additive way.

We conclude heuristically that the exponentially diminishing centered moment increase with the number of boosting iterations (as stated in Theorem 3), together with the tapering in the smoothed 0-1 loss yield the overall, often strong, overfitting-resistance performance of boosting in classification.

5.2 Acceleration of F and classification noise

As seen in Section 5.1, 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 $\{x; F(x) = 0\}$, 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 zero. $F(\cdot)$ is said to have a large acceleration if its gradient is large (element-wise 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 $\mathbb{P}[W = -1] = \pi$, $0 \le \pi \le 1/2$. The noisy response variable is

$$\tilde{Y} = WY, \tag{18}$$

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

$$\mathbb{P}[\tilde{Y} = 1|X = x] = \mathbb{P}[Y = 1|X = x](1 - 2\pi) + \pi,\ 0 \le \pi \le 1/2. \tag{19}$$

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

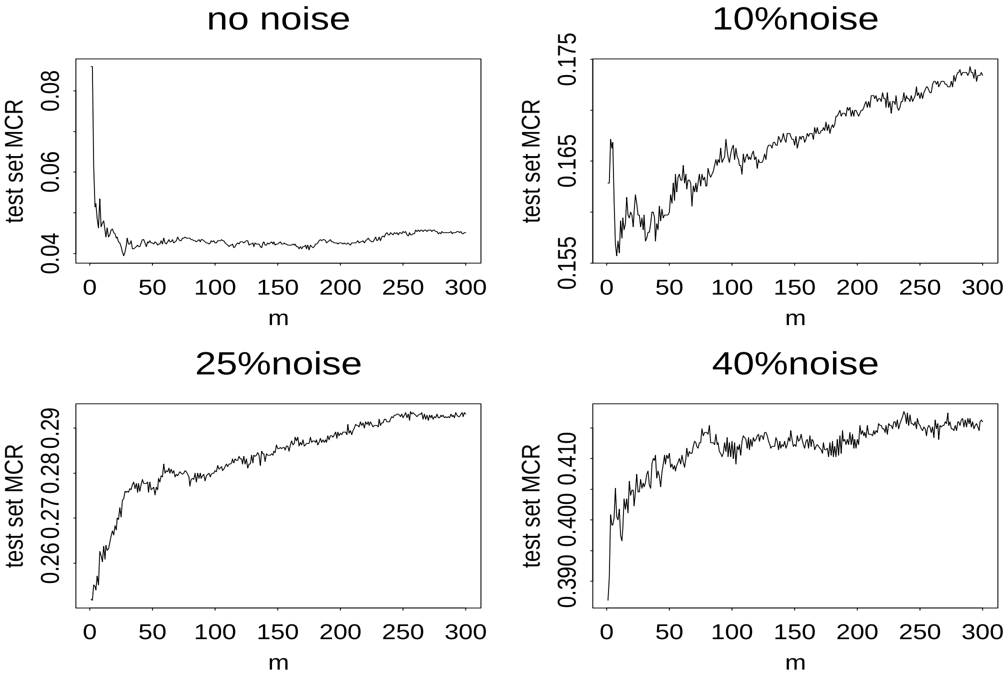

Figure 3: Test set misclassification rates (MCR) for LogitBoost with stumps. Breast cancer data with different noise levels π (in %) as described in (18).

straightforward calculation then shows,

$$\mathrm{grad}(\tilde{F}(x))|_{x=0} \searrow 0 \text{ as } \pi \nearrow 1/2. \tag{20}$$

The noisier the problem, the smaller the acceleration of $\tilde{F}(\cdot)$ and thus less resistance against overfitting since the tapering weights in (15) are becoming larger in noisy problems. This adds deeper insight to the known empirical fact that boosting doesn't work well in noisy problems; see Dietterich (1999). The reason is that overfitting 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 (http://www.ics.uci.edu/~mlearn/MLRepository) which has been analyzed by many others. We see there that already a stump 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 $\tilde{\mathcal{C}}_{Bayes}(\cdot)$ in the noisy problem has misclassification rate

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

relating it to the Bayes classifier $\mathcal{C}_{Bayes}$ of the original non-noisy problem. This is easily derived using (19). With high noise, the expression is largely dominated by the constant term π, indicating that there isn't 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 due to the high noise level causing a large misclassification benchmark rate.

5.3 L_2Boost in classification

The (true) margin is referred as

$$YF_{lo}(X),\ Y \in \{-1,1\},\ F_{lo}(x) = \frac{1}{2}\log\left(\frac{p(x)}{1-p(x)}\right),\ p(x) = \mathbb{P}[Y=1|X=x].$$

We emphasize here the notation $F_{lo}(\cdot)$ to denote half of the **l**og-**o**dds ratio. Many reasonable loss functions for classification are functions of the margin. For example, $\exp(-YF_{lo}(X))$ (exponential loss for AdaBoost), $\log_2(1+\exp(-2YF_{lo}(X)))$ (negative log-likelihood for LogitBoost). The L_2-loss is also a function of the margin since its minimizer $\mathbb{E}(Y|X=x)$ can be written as a function of the margin $YF_{lo}(X)$ as stated in the following proposition.

Proposition 4.

$$|Y - \mathbb{E}[Y|X=x]|^2 = |Y - (2p(x)-1)|^2 = 4/(1+exp(2YF_{lo}(x))^2. \tag{21}$$

A proof is given in the Appendix. This is what Friedman et al. (2000) called "squared error (p)".

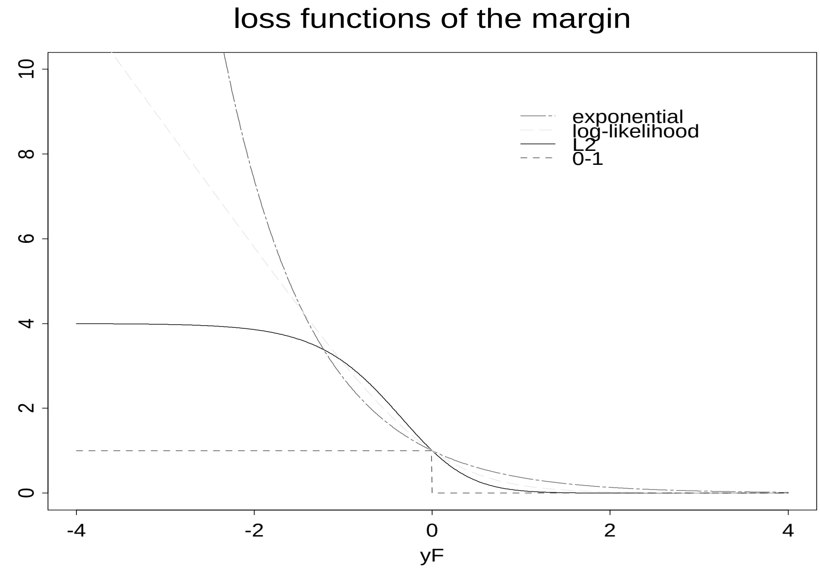

Figure 4: Various loss functions of YF_{lo}.

Figure 4 shows that the L_2-loss is a competitive approximation to the 0-1 loss relative to the negative log-likelihood or exponential loss: particularly when the margin takes large negative values (strong misclassifications). Near the classification boundary $YF(X)=0$, for the (small) positive margins, the L_2-loss is a tighter bound than exponential or log-likelihood, but for the (small) negative margins, it is reversed. Therefore the L_2-loss seems at least as good as the exponential or log-likelihood loss to bound the 0-1 loss.

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

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

where ε_i are independent, mean zero 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 5. *(optimality of L2Boost for smoothing splines in classification). Consider the case with one-dimensional predictor $x \in \mathbb{R}$. Suppose $p \in \mathcal{F}^{(\nu)}$, see (9), and $\mathcal{S}$ is a smoothing spline linear learner $g_r(\lambda_0)$ of degree r, corresponding to a fixed smoothing parameter λ_0. If $\nu \geq r$, then there is an $m = m(n) = O(n^{2r/(2\nu+1)}) \to \infty$ such that $\hat{F}_{m(n)}$ achieves the optimal minimax rate $n^{-2\nu/(2\nu+1)}$ of the smoother function class $\mathcal{F}^{(\nu)}$ for estimating $2p(\cdot) - 1$ in terms of MSE.*

It follows that if the underlying p belongs to one of the smooth function class $\mathcal{F}^{(\nu)}$, L_2Boosting achieves the Bayes risk. In view of the results in Marron (1983), Theorem 5 also implies that L_2Boost with smoothing splines gives the minimax optimal rates of convergence for the classification problem over the global smoothness class $\mathcal{F}^{(\nu)}$, defined by

$$min_{\hat{F}_n} max_{p \in \mathcal{F}^{(\nu)}} \{\mathbb{P}[Y\hat{F}_n(X) < 0] - BayesRisk\}$$

where $\hat{F}_n$ is *any* classification rule based on n i.i.d. observations (Y_i, X_i). Mammen and Tsybakov (1999) consider different function classes which are locally constrained near the decision boundary and show that faster 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 since, as seen from our smoothed 0-1 loss expansion in Section 5.1, the actions happen near the decision boundary. These new rates are achieved by avoiding the plug-in classification rules via estimating p and going directly to empirical minimizations of the 0-1 loss function over regularized classes of decision regions. However, computationally such a minimization could be very difficult. It remains open whether boosting can achieve these new optimal convergence rates in Mammen and Tsybakov (1999).

5.4 Comparing L2Boost with LogitBoost on Real Data Sets

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

L_2WCBoost algorithm

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

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

Update

$$\tilde{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + \hat{f}_{m+1}(\cdot),$$
$$\hat{F}_{m+1}(x) = \text{sign}(\tilde{F}_{m+1}(x)) \min\left(1, |\tilde{F}_{m+1}(x)|\right).$$

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

We compare L_2Boost, our L_2WCBoost and LogitBoost using tree learners and our component-wise smoothing spline learner from Section 4.1. We consider first a variety of 2-class problems from the UCI machine learning repository (http://www.ics.uci.edu/~mlearn/MLRepository): Breast cancer, Ionosphere, Monk 1 (full data set with $n = 432$) and the

dataset	n	p	learner	L_2Boost	L_2WCBoost	LogitBoost
Breast cancer	699	9	stumps	0.037 (176)	0.040 (275)	0.039 (27)
			comp. spline	0.036 (126)	0.043 (73)	0.038 (5)
Sonar	210	60	stumps	0.228 (62)	0.190 (335)	0.158 (228)
			comp. spline	0.178 (51)	0.168 (47)	0.148 (122)
Ionosphere	351	34	stumps	0.088 (25)	0.079 (123)	0.070 (157)
Heart (without costs)	270	13	stumps	0.167 (4)	0.175 (3)	0.159 (3)
Australian credit	690	14	stumps	0.123 (22)	0.123 (19)	0.131 (16)
Monk 1	432	7	larger tree	0.002 (42)	0.004 (12)	0.000 (6)

Table 4: Test set misclassification rates for L_2Boost, L_2WCBoost (with constraints) and LogitBoost. Optimal 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.

Heart, Sonar and Australian credit data sets from the Statlog project. Monk 1 has Bayes error equal to zero. The estimated test set misclassification rates, using an average of 50 random divisions into training with 90% and test set with 10% of the data, are given in Table 4. The comparison is made when using the optimal number of boosting for every boosting algorithm; these numbers are given in parentheses. The component-wise 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 most sense and we also found empirically that it is there where it improves upon the tree-based stumps. L_2WCBoost performs overall a bit better than L_2Boost, although in half of the data sets L_2Boost was better. LogitBoost was better than L_2Boost in 4 out of 6 and better than L_2WCBoost in 5 out of 6 data sets, 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 data set, the difference is far from being significant since sample size is much too small.

Therefore, we consider next simulated data to compare the L_2WCBoost (the slightly better of the L_2procedures) with the LogitBoost algorithm. It allows a more accurate comparison by choosing large test sets. We generate data with two classes from the model in Friedman et al. (2000) with a non-additive decision boundary,

$$X \sim \mathcal{N}_{10}(0, I), \ Y|X = x \sim 2 \text{ Bernoulli}(p(x)) - 1,$$
$$\log(p(x)/(1 - p(x))) = 10 \sum_{j=1}^{10} x_j (1 + \sum_{k=1}^{6} (-1)^k x_k). \tag{22}$$

The (training) sample size is $n = 2000$. It is interesting to consider the performance on a single training and test data which resembles the situation in practice. For that purpose 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. Additionally, we consider an average performance over 10 independent realizations from the model: here, we choose test set size as 10'000 which is still large compared to the training sample size $n = 2000$. The latter is the same set-up as in Friedman et al. (2000).

Figure 5 displays the results on a single data set. L_2WCBoost and LogitBoost perform about equally well. There is a slight advantage for L_2WCBoost with the larger tree learner. It has been pointed out by Friedman et al. (2000) that stumps aren't good

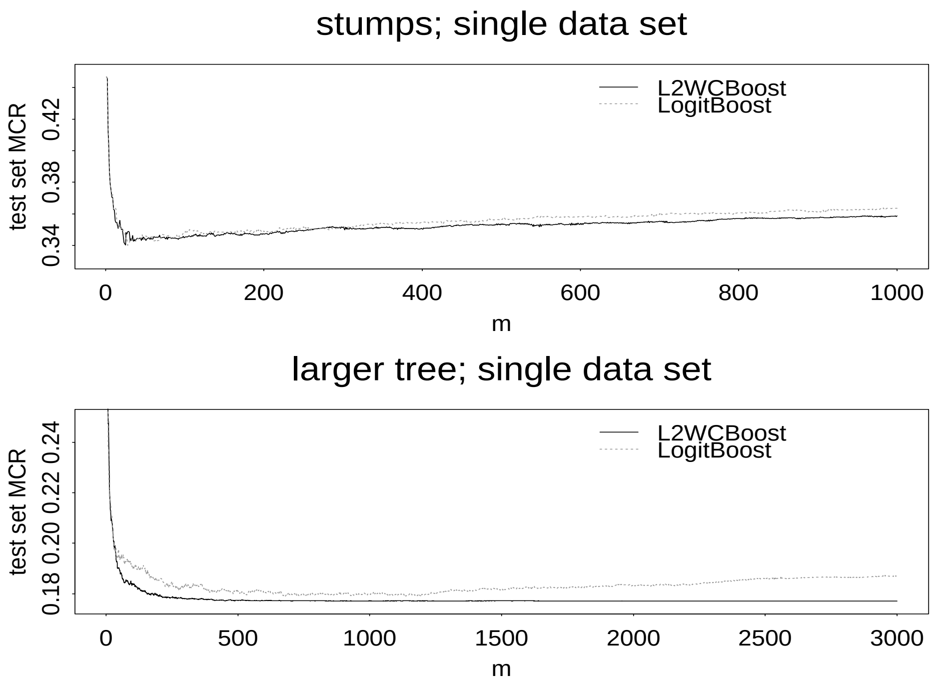

Figure 5: Test set misclassification rates (MCR) for a single realization from model (22). Top: stumps as learner. Bottom: larger tree as learner.

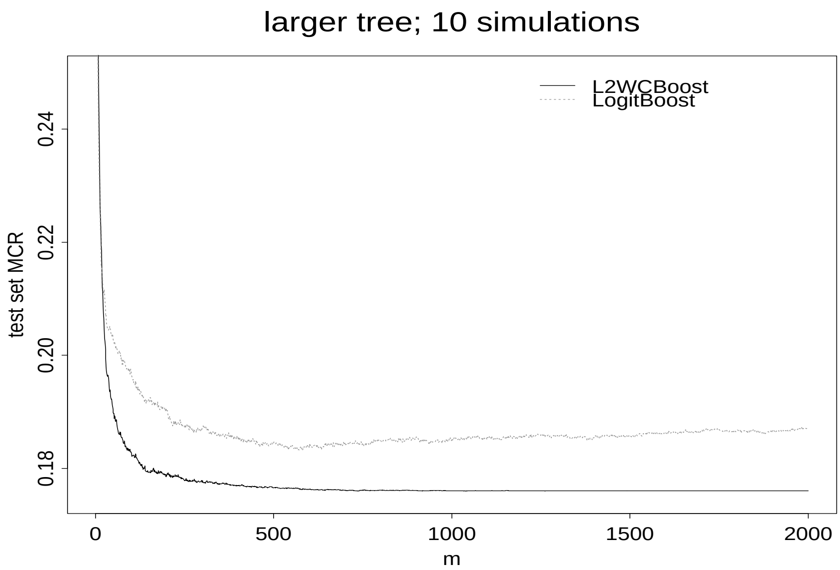

Figure 6: Test set misclassification rates (MCR) averaged over 10 realization from model (22). Larger tree as learner.

learners because the true decision boundary is non-additive. With stumps as learners, the optimally stopped LogitBoost has a tiny advantage over L_2WCBoost (by about 0.13 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 (by about 1.54 standard errors, given the data).

Figure 6 shows the averaged performance over 10 independent realizations with larger trees as weak learner: it indicates a more substantial advantage than on the single data represented by Figure 5. With 10 independent realizations, we test whether an optimally stopped Boosting algorithm yields a significantly better test set misclassification rate. In 9 out 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 5.

test	p-value
t	0.0015
Wilcoxon	0.0039
sign	0.0215

Table 5: Comparison of L_2WCBoost and LogitBoost: two-sided testing for equal test set performance. Low p-values are always in favor of L_2WCBoost.

Thus, for the model (22) from Friedman et al. (2000), we find a significant advantage of L_2WCBoost over LogitBoost.

6 Discussion and concluding remarks

As seen in Section 3 (see also Figure 2), L_2Boosting is only successful if the learner is weak. If the learning procedure is too strong, then even at the first boosting iteration the MSE is not improved over the original. 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 numerical convergence of the boosting algorithm to the fully saturated model (at least for linear learners with eigenvalues bounded away from zero). Of course, weakness of a learner depends also on the underlying problem, i.e. the data generating distribution: see also assertion (2) in Theorem 2. It is often hard to decide a-priori whether a learner is weak; a posteriori, we can estimate generalization error in boosting and hence obtain information about strength of the learner. The degree of weakness of a learner can be increased by 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 3.1, saying that boosting weak smoothing splines is as good or even better than using an optimal smoothing spline (which is then a strong learner) since boosting adapts to unknown smoothness. In high dimensions, we see most practical advantages of boosting, particularly when the dimension of the predictor space is very large relative to sample size: we have exemplified this in Section 4.2.

The fact that L_2Boost depends critically on the strength of the learner is also (empirically) true for other versions of boosting. We show in Figure 7 some results for LogitBoost with trees and projection pursuit learners with one ridge function (one term) for the breast cancer data. All tree learners are sufficiently weak for the problem and a shrunken large tree seems best. Interestingly, the projection pursuit learner is already strong and boost-

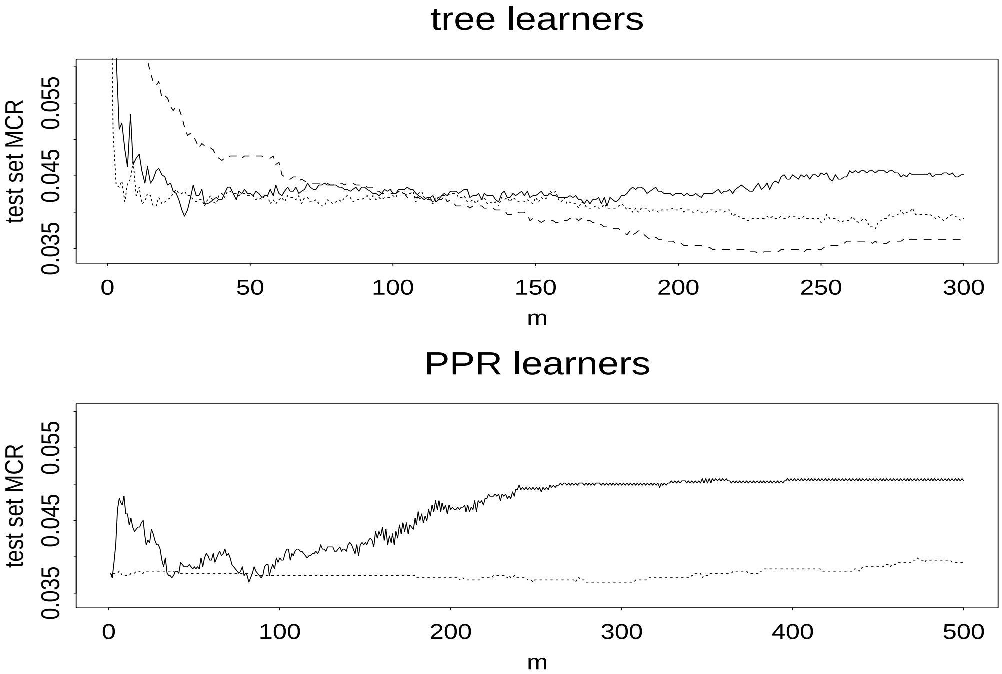

Figure 7: Test set misclassification rates (MCR) of LogitBoost for breast cancer data. Top: decision tree learners, namely stumps (solid line), large unpruned tree (dotted line) and shrunken large unpruned tree (dashed line) with shrinkage factor $\nu = 0.01$. Bottom: projection pursuit (PPR) learners , namely one term PPR (solid line) and shrunken one term PPR (dotted line) with shrinkage factor $\nu = 0.01$.

ing doesn't pay off. This matches the intuition that projection pursuit is very flexible, 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 misclassification curve shows overfitting and stabilizes at a relatively high value. Presumably, the boosting estimate with $m = 2$ "overshoots" in the second functional gradient descent step because the learner is too strong. Additional boosts correct again and increase the performance (for a while). We observed a similar pattern with projection pursuit learners in another simulated example which allows for very accurate estimation of the misclassification rate. Such a phenomenon is typically not observed with tree learners, but it is not inconsistent with our theoretical arguments.

Most of the empirical studies of boosting in the literature use a tree-structured learner. The complexity is often low for tree procedures because they are fitted by a stagewise selection of variables and split points, as in CART and other tree algorithms (while the complexity would be much higher, or it would be a much stronger learner, if the tree were fitted by a global search for split points – which is of course infeasible). This is another example of a local greedy method leading to low increase in complexity, similarly as we analyzed for L_2Boosting. When the predictor space is high-dimensional, the substantial amount of variable selection done with a tree procedure is particularly helpful in leading to a low complexity (or weak) learner. And this feature may be very desirable.

In this paper, we mainly investigated L_2Boost, taking advantage of its analytical tractability, and demonstrated its practical effectiveness. In particular, we showed that

1. L_2Boost is appropriate both for 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 (this is also true with other boosting algorithms). In the linear weak learner case, m controls a new exponential bias-variance trade-off.

3. L_2Boost with smoothing splines results in optimal minimax rates of convergence, both in regression and classification. Moreover, the algorithm adapts to unknown smoothness.

4. 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 is likely 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.

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

6. Boosting learners which involve only one (selected) predictor variable yields an additive model fit. We propose component-wise cubic smoothing splines, which are of such type; they are most often better learners than tree-structured stumps, especially for continuous predictors.

Acknowledgements

We would like to thank Trevor Hastie and Leo Breiman for very helpful discussions. Partial support to B. Yu is gratefully acknowledged from the National Science Foundation (DMS-9803063 and FD01-12731) and the Army Research Office (DAAG55-98-1-0341 and DAAD19-01-1-0643).

Appendix

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 $u_j = Y - F_{j-1}$ and $F_j = F_{j-1} + f_j$ (there is no need for a line search) with $f_j = \mathcal{S}u_j$. Thus,

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

implying $u_j = (I - \mathcal{S})^j Y$ for $j = 1, 2, \ldots, m$. Since $F_0 = \mathcal{S}Y$ we obtain $\hat{F}_m = \sum_{j=0}^{m} \mathcal{S}(I - \mathcal{S})^j Y$. Using a telescope-sum argument, this equals $(I - (I - \mathcal{S})^{m+1})Y$. □

Proof of Theorem 1.
The bias term is

$$\mathrm{bias}(m, \mathcal{S}; f) = (\mathbb{E}[\mathcal{B}_m Y] - f)^T (\mathbb{E}[\mathcal{B}_m Y] - f) = ((\mathcal{B}_m - I)f)^T ((\mathcal{B}_m - I)f).$$

According to (7), using orthonormality of U,

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

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

For the variance, consider

$$\text{Cov}(\mathcal{B}_m Y) = \mathcal{B}_m \text{Cov}(Y)\mathcal{B}_m^T = \sigma^2 \mathcal{B}_m \mathcal{B}_m^T = \sigma^2 U\text{diag}((1-(1-\lambda_k)^{m+1})^2)U^T,$$

using (7) and orthonormality of U. Then,

$$\text{variance}(m, \mathcal{S}; \sigma^2) = tr[\text{Cov}(\mathcal{B}_m Y)] = \sigma^2 \sum_{k=1}^{n} (1-(1-\lambda_k)^{m+1})^2,$$

again using orthonormality of U. □

Proof of Theorem 2.

Assertion (1) is an immediate consequence of Theorem 1.

Without loss of generality, assume $\lambda_k < 1$ for all k. If not, restrict the summation to those k's which satisfy $\lambda_k < 1$.

(i) Denote by $\mu = U^T f \in \mathbb{R}^n$. For $x \geq 0$, let

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

Then

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

It is easy to derive

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

It follows that

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

which are both negative under the inequality condition in (i); also $g'(x) < 0$ for $x \in (0,1)$ under this condition. Hence $g(1) < g(0)$ which means boosting improves.

(ii) Rewrite

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

Since 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 $(\mu_k^2 + \sigma^2)(1-\lambda_k)^{m+1} - 2\sigma^2 \leq -\sigma^2$ for all k (with $\lambda_k < 1$). It follows that

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

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

Proof of Theorem 3.
Write the summands with the higher order moment as

$$\mathbb{E}[(\hat{F}_m(x_i) - f(x_i))^p] = \sum_{j=0}^{p} \binom{p}{j} b_m(x_i)^j \mathbb{E}[(\hat{F}_m(x_i) - \mathbb{E}[\hat{F}_m(x_i)])^{p-j}], \tag{23}$$

where $b_m(x_i) = \mathbb{E}[\hat{F}_m(x_i) - f(x_i)]$ 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 which goes to zero as $m \to \infty$. The centered moments can be written as

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

where we used assertion 1 from Proposition 1. Since $(I - \mathcal{S})^{m+1}$ is a map (matrix) taking values exponentially close to zero as $m \to \infty$, we obtain

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

for some constant $C_q > 0$. From this last bound and using (23) together with the fact that the bias $b_m(x_i) = O(\exp(-C_b m))$ $(m \to \infty)$ for some constant $C_b > 0$ (see Theorem 2), we complete the proof. $\square$

Proof of Theorem 4.
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 (cf. Utreras, 1983; Wahba, 1990, p.61) that the eigenvalues of $\mathcal{S}$ take the form in decreasing order

$$\lambda_1 = ... = \lambda_r = 1, \;\; \lambda_k = \frac{n q_{k,n}}{n\lambda_0 + n q_{k,n}} \text{ for } k = r+1, ..., n.$$

Moreover, for n large, $q_{k,n} \approx Ak^{-2r} := Aq_k$ where A is universal and depends on the asymptotic density of the design points x_i. For the true function $f \in \mathcal{F}^{(\nu)}$,

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

Let $c_0 = c/A$, then

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

Then the bias term can be bounded as follows.

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

where

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

Taking derivative gives

$$
h'(x) = \frac{2r}{x} \frac{1}{c_0 x^{2r} + 1}[(2m+2) - \frac{\nu}{r}(c_0 x^{2r} + 1)].
$$

Hence for any given positive integer n_1, if $x \leq n_1$ and $m \geq \frac{\nu}{2r}(c_0 n_1^{2r} + 1) - 1$, $h(x)$ is increasing and so is $\exp(h(x))$, and

$$
\exp(h(x)) \leq \exp((h(n_1)) = (1 - 1/(c_0 n_1^{2r} + 1))^{2m+2} n_1^{-2\nu}.
$$

On $[n_1 + 1, n]$,

$$
\exp(h(x)) \leq (1 - 1/(c_0 n^{2r} + 1))^{2m+2} n_1^{-2\nu}.
$$

Putting them together we get for any given n_1 and $m \geq \frac{\nu}{2r}(c_0 n_1^{2r} + 1) - 1$,

$$
\mathrm{bias}(m, \mathcal{S}; f) \leq M n_1^{-2\nu}[2(1 - 1/(c_0 n^{2r} + 1))^{2m+2}]
$$

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

Now let's deal with the variance term. For any $n_1 > r$,

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

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

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

It follows that

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

if we take $m = m(n_1) = \frac{\nu}{2r}(c_0 n_1^{2r} + 1) - 1 = O(n_1^{2r})$. Hence for this choice of $m(n_1)$,

$$
\mathrm{variance}(m(n_1), \mathcal{S}; \sigma^2) \leq O(n_1/n).
$$

Together with the bound for the bias we get

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

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 $\mathcal{F}^{(\nu)}$. □

Proof of Proposition 3.
Denote by $C(z) = \mathbf{1}_{[z<0]}$. We first show that

$$\sup_{|z|>\gamma\log(\gamma^{-1})} |C(z) - C_\gamma(z)| = \gamma/2. \tag{24}$$

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

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

proving (24).

On the other hand,

$$\int_{|z|\le\gamma\log(\gamma^{-1})} |C(z) - C_\gamma(z)|g(z)dz \le \sup_z |g(z)|\gamma\log(\gamma^{-1}) = O(\gamma\log(\gamma^{-1})). \tag{25}$$

Hence, by (24) and (25),

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

□

Proof of Proposition 4.
The minimizer of the L_2-loss is $\mu(x) = 2p(x) - 1$ and the minimized L_2-loss is

$$(Y - \mu(x))^2 = 1 - 2Y(2p(x) - 1) + (2p(x) - 1)^2.$$

Let $yF(x) = u$. Then, for $y = 1$: $F(x) = u$ and $p(x) = \exp(2u)/(1 + \exp(2u))$, which give

$$\begin{aligned}
(Y - \mu(x))^2 &= 1 - 2Y(2p(x) - 1) + (2p(x) - 1)^2 \\
&= 1 - 2(\exp(2u) - 1)/(\exp(2u) + 1) + (\exp(2u) - 1)^2/((\exp(2u) + 1)^2 \\
&= 4/(1 + \exp(2u))^2
\end{aligned}$$

For $y = -1$: $F(x) = -u$ and $p(x) = \exp(-2u)/(1 + \exp(-2u))$, which give

$$\begin{aligned}
(Y - \mu(x))^2 &= 1 - 2(-\exp(2u) + 1)/(\exp(2u) + 1) + (\exp(2u) - 1)^2/((\exp(2u) + 1)^2 \\
&= 4/(1 + \exp(2u))^2
\end{aligned}$$

□

References

[1] Buja, A. (2000). Comment on "Additive logistic regression: a statistical view of boosting". Ann. Statist. **28**, 387–391.

[2] Breiman, L. (1998). Arcing classifiers. Ann. Statist. **26**, 801-824.

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

[4] Breiman, L. (2000). Some infinity theory for predictor ensembles. Tech. Report 579, Dept. of Statist., Univ. of Calif., Berkeley.

[5] Collins, M., Schapire, R.E. and Singer, Y. (2000). Logistic regression, AdaBoost and Bregman distances. Proc. Thirteenth Annual Conference Computational Learning Theory.

[6] Dietterich, T.G. (1999). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning **40**, 139–157.

[7] Freund, Y. (1995). Boosting a weak learning algorithm by majority. Information and Computation **121**, 256–285.

[8] 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.

[9] Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. To appear in the Annals of Statist.

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

[11] Geman, S., Bienenstock, E. and Doursat, R. (1992). Neural networks and the bias/variance dilemma. Neural Computations **4**, 1–58.

[12] Gu, C. (1987). What happens when bootstrapping the smoothing spline? Commun. Statist. Part A - Theory Meth. **16**, 3275–3284.

[13] Hastie, T.J. and Tibshirani, R.J. (1990). *Generalized Additive Models*. Chapman & Hall.

[14] Jiang, W. (2000). Process consistency for AdaBoost. Tech. Report, Dept. of Statistics, Northwestern University.

[15] Mammen, E. and Tsybakov, A.B. (1999). Smooth discriminant analysis. Ann. Statist. **27**, 1808–1829.

[16] Marron, J.S. (1983). Optimal rates of convergence to Bayes risk in nonparametric discrimination. Ann. Statist. **11**, 1142–1155.

[17] Mason, L., Baxter, J. Bartlett, P. and Frean, M. (1999). Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press.

[18] Schapire, R.E. (1990). The strength of weak learnability. Machine Learning **5**, 197–227.

[19] Schapire, R.E., Freund, Y., Bartlett, P. and Lee, W.S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. Ann. Statist. **26**, 1651–1686.

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

[21] Utreras, F. (1983). Natural spline functions, their associated eigenvalue problem. Numer. Math. **42**, 107–117.

[22] Wahba, G. (1990). *Spline Models for Observational Data.* Soc. for Industrial and Applied Mathematics.