Journal of Machine Learning Research 9 (2008) 131-156 Submitted 10/05; Revised 7/07; Published 2/08

Evidence Contrary to the Statistical View of Boosting

David Mease MEASE_D@COB.SJSU.EDU
Department of Marketing and Decision Sciences
College of Business, San Jose State University
San Jose, CA 95192-0069, USA

Abraham Wyner AJW@WHARTON.UPENN.EDU
Department of Statistics
Wharton School, University of Pennsylvania
Philadelphia, PA, 19104-6340, USA

Editor: Yoav Freund

Abstract

The statistical perspective on boosting algorithms focuses on optimization, drawing parallels with maximum likelihood estimation for logistic regression. In this paper we present empirical evidence that raises questions about this view. Although the statistical perspective provides a theoretical framework within which it is possible to derive theorems and create new algorithms in general contexts, we show that there remain many unanswered important questions. Furthermore, we provide examples that reveal crucial flaws in the many practical suggestions and new methods that are derived from the statistical view. We perform carefully designed experiments using simple simulation models to illustrate some of these flaws and their practical consequences.

Keywords: boosting algorithms, LogitBoost, AdaBoost

1. Introduction

As the AdaBoost algorithm of Freund and Schapire (1996) gained popularity in the computer science community because of its surprising success with classification, the statistics community focused its efforts on understanding how and why the algorithm worked. Friedman, Hastie and Tibshirani in 2000 made great strides toward understanding the AdaBoost algorithm by establishing a statistical point of view. Among the many ideas in the Friedman, Hastie and Tibshirani *Annals of Statistics* paper, the authors identified a stagewise optimization in AdaBoost, and they related it to the maximization of the likelihood function in logistic regression. Much work has followed from this paper: extensions of the algorithm to the regression setting (e.g., Buhlmann and Yu, 2003), modification of the loss function (e.g., Hastie et al., 2001), and work on regularization methods for the original AdaBoost algorithm and variants (e.g., Lugosi and Vayatis, 2004). This broad statistical view of boosting is fairly mainstream in the statistics community. In fact, the statistics community has taken to attaching the *boosting* label to *any* classification or regression algorithm that incorporates a stagewise optimization.

Despite the enormous impact of the Friedman, Hastie and Tibshirani paper, there are still questions about the success of AdaBoost that are left unanswered by this statistical view of boosting. Chief among these is the apparent resistance to overfitting observed for the algorithm in countless examples from both simulated and real data sets. This disconnect was noted in some of the dis-

cussions published along with the original 2000 *Annals of Statistics* paper. For instance, Freund and Schapire (2000) note that, "one of the main properties of boosting that has made it interesting to statisticians and others is its relative (but not complete) immunity to overfitting," and write that the paper by Friedman, Hastie and Tibshirani "does not address this issue." Also Breiman (2000) writes, "a crucial property of AdaBoost is that it almost never overfits the data no matter how many iterations it is run," and states "unless I am missing something, there is no explanation in the paper."

Various arguments are given in response to the question of why boosting seems to not overfit. A view popular in computer science attributes the lack of overfitting to boosting's ability to achieve a large *margin* separating the two classes, as discussed by Schapire et al. (1998). A number of different opinions exist in the statistics community. Many statisticians simply argue that boosting does in fact overfit and construct examples to prove it (e.g., Ridgeway, 2000). While single examples certainly disprove claims that boosting *never* overfits, they do nothing to help us understand why boosting resists overfitting and performs very well for the large collection of examples that raised the question in the first place. Others argue that boosting will eventually overfit in most all cases if run for enough iterations, but that the number of iterations needed can be quite large since the overfitting is quite slow. Such a notion is difficult to disprove through real examples since any finite number of iterations may not be enough. Furthermore, it is difficult to prove limiting results for an infinite number of iterations without substantially over-simplifying the algorithm. Some evidence supporting the argument that boosting will eventually overfit can be found in Grove and Schuurmans (1998) which has examples for which boosting overfits when run for a very large number of iterations. Another argument often used is that boosting's success is judged with respect to 0/1 misclassification loss, which is a loss function that is not very sensitive to overfitting (e.g., Friedman et al., 2000b). More detailed explanations attribute the lack of overfitting to the stagewise nature of the algorithm (e.g., Buja, 2000). Along this same line, it has also been observed that the repeated iterations of the algorithm give rise to a self-averaging property (e.g., Breiman, 2000). This self-averaging works to reduce overfitting by reducing variance in ways similar to bagging (Breiman, 1996) and Random Forests (Breiman, 2001).

Whatever the explanation for boosting's resistance to overfitting in so many real and important examples, the statistical view of boosting as an optimization does little to account for this. In fact the statistical framework as proposed by Friedman, Hastie and Tibshirani does exactly the opposite; it suggests that *overfitting should be a major concern*. Still, in the final analysis, we do not imply that the statistical view is wrong. Indeed, we agree with Buja (2000) who writes, "There is no single *true* interpretation of anything; interpretation is a vehicle in the service of human comprehension. The value of an interpretation is in enabling others to fruitfully think about an idea." Certainly the paper of Friedman, Hastie and Tibshirani and other related work is quite valuable in this regard. However, any view or theoretical understanding generally gives rise to practical suggestions for implementation. Due to the disconnect between the statistical view and reality, many of these resulting practical suggestions are misguided and empirical performance suffers accordingly. In this paper we focus on illustrating this phenomenon through simulation experiments.

It is important to note that although this paper deals with "the statistical view of boosting", it is an overgeneralization to imply there is only one single view of boosting in the statistical community. All statisticians are not of a single mindset, and much literature has been produced subsequent to the Friedman, Hastie and Tibshirani *Annals of Statistics* paper. Much of what we categorize as the statistical view of boosting can be found in that original paper, but other ideas, especially those in Sections 3.9, 3.10, 4.9 and 4.10, are attributable to other researchers and subsequent publications in

the statistics community. For this reason, we are careful to provide references and direct quotations throughout this paper.

The following section describes the general setting for classification and the AdaBoost algorithm. Sections 3 and 4 consider a collection of practical suggestions, commonly held beliefs and modifications to the AdaBoost algorithm based on the statistical view. For each one, a simulation providing contradictory evidence is included. Section 5 mentions a slightly different set of simulations to consider, and finally Section 6 offers practical advice in light of the evidence presented in this paper as well as some concluding remarks.

2. The Classification Problem and Boosting

In this section we will begin by describing the general problem of classification in statistics and machine learning. Next we will describe the AdaBoost algorithm and give details of our implementation.

2.1 Classification

The problem of classification is an instance of what is known as *supervised learning* in machine learning. We are given training data $x_1,...,x_n$ and $y_1,...,y_n$ where each x_i is a $d-$dimensional vector of predictors $(x_i^{(1)},...,x_i^{(d)})$ and $y_i \in \{-1,+1\}$ is the associated observed class label. To justify generalization, it is usually assumed that the training data are *iid* samples of random variables (X,Y) having some unknown distribution. The goal is to learn a rule $\hat{C}(x)$ that assigns a class label in $\{-1,+1\}$ to any new observation x. The performance of this rule is usually measured with respect to misclassification error, or the rate at which new observations drawn from the same population are incorrectly labelled. Formally we can define the misclassification error for a classification rule $\hat{C}(x)$ as $\mathrm{P}(\hat{C}(X) \neq Y)$.

For any given data set misclassification error can be estimated by reserving a fraction of the available data for test data and then computing the percent of incorrect classifications resulting from the classifier trained on the remainder of the data. Various cross-validation techniques improve upon this scheme by averaging over different sets of test data. In this paper we will consider only examples of simulated data so that the joint distribution of X and Y is known. This will enable us to estimate misclassification error as accurately as desired by simply repeatedly simulating training and test data sets and averaging the misclassification errors from the test sets.

2.2 Boosting

AdaBoost (Freund and Schapire, 1996) is one of the first and the most popular boosting algorithms for classification. The algorithm is as follows. First let $F_0(x_i) = 0$ for all x_i and initialize weights $w_i = 1/n$ for $i = 1,...,n$. Then repeat the following for m from 1 to M:

- Fit the classifier g_m to the training data using weights w_i where g_m maps each x_i to -1 or 1.
- Compute the weighted error rate $\varepsilon_m \equiv \sum_{i=1}^{n} w_i \mathrm{I}[y_i \neq g_m(x_i)]$ and half its log-odds, $\alpha_m \equiv \frac{1}{2}\log\frac{1-\varepsilon_m}{\varepsilon_m}$.
- Let $F_m = F_{m-1} + \alpha_m g_m$.
- Replace the weights w_i with $w_i \equiv w_i e^{-\alpha_m g_m(x_i) y_i}$ and then renormalize by replacing each w_i by $w_i/(\sum w_i)$.

The final classifier is 1 if $F_M > 0$ and -1 otherwise. The popularity of this algorithm is due to a vast amount of empirical evidence demonstrating that the algorithm yields very small misclassification error relative to competing methods. Further, the performance is remarkably insensitive to the choice of the total number of iterations M. Usually any sufficiently large value of M works well. For the simulations in this paper we will take $M = 1000$, with the single exception of the simulation in Section 4.7 where it is instructive to consider $M = 5000$.

Many variations of the AdaBoost algorithm now exist. We will visit some of these in Sections 3 and 4 and compare their performance to the original AdaBoost algorithm. Further, these variations as well as AdaBoost itself are very flexible in the sense that the class of classifiers from which each g_m is selected can be quite general. However, the superior performance of AdaBoost is generally in the context of classification trees. For this reason we will use classification trees in our experiments. Specifically, the trees will be fit using the "rpart" function in the "R" statistical software package (http://www.r-project.org/). The R code for all the experiments run in this paper is available on the web page http://www.davemease.com/contraryevidence.

3. Experiments Which Contradict the Statistical View of Boosting

In this section we describe the results of several experiments based on simulations from the model introduced below. Each experiment is meant to illustrate particular inconsistencies between that which is suggested by the statistical view of boosting and what is actually observed in practice.

For the experiments we will consider in this section we will simulate data from the model

$$\mathrm{P}(Y = 1|x) = q + (1-2q)\ \mathrm{I}\left[\sum_{j=1}^{J} x^{(j)} > J/2\right].$$

We will take X to be distributed *iid* uniform on the d-dimensional unit cube $[0,1]^d$. The constants n, d, J and q will be set at different values depending on the experiment. Note that q is the Bayes error and $J \leq d$ is the number of effective dimensions. Recall n is the number of observations used to train the classifier. The unconditional probabilities for each of the two classes are always equal since $\mathrm{P}(Y = 1) = \mathrm{P}(Y = 0) = 1/2$. The only exceptions to this are experiments in which we take $J = 0$ for which the sum (and thus the indicator) is taken to be always zero. In these cases the model reduces to the "pure noise" model $\mathrm{P}(Y = 1|x) \equiv q$ for all x.

3.1 Should Stumps Be Used for Additive Bayes Decision Rules?

Additive models are very popular in many situations. Consider the case in which the Bayes decision rule is additive in the space of the predictors $x^{(1)},...,x^{(d)}$. By this we mean that the Bayes decision rule can be written as the sign of $\sum_{i=1}^{d} h_i(x^{(i)})$ for some functions $h_1,...,h_d$. This is true, for example, for our simulation model. The classification rule produced by AdaBoost is itself necessarily additive in the classifiers g_m. Thus when the g_m are functions of only single predictors the AdaBoost classification rule is additive in the predictor space. For this reason it has been suggested that one should use stumps (2-node trees) if one believes the optimal Bayes rule is approximately additive, since stumps are trees which only involve single predictors and thus yield an additive model in the predictor space for AdaBoost. It is believed that using trees of a larger size will lead to overfitting because it introduces higher-level interactions. This argument is made explicit in Hastie et al. (2001) on pages 323-324 and in Friedman et al. (2000a) on pages 360-361.

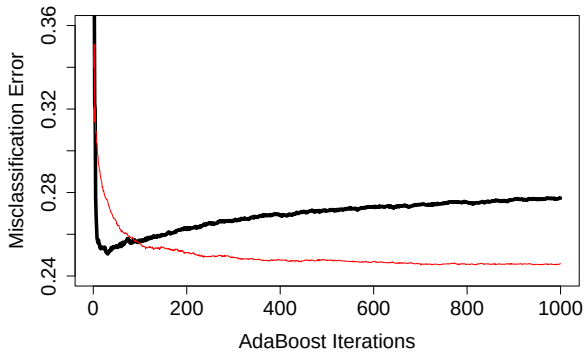


Figure 1: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule

Despite the logic of this argument which is based on the idea that one should use an additive model when fitting an additive function, it can be observed that often, in fact, using larger trees is more effective than using stumps even when the Bayes rule is additive. The reason has to do with the fact that boosting with larger trees actually often overfits *less* than boosting with smaller trees in practice since the larger trees are more orthogonal and a self-averaging process prevents overfitting. We do not endeavor to make this argument rigorous here, but we will provide a compelling example.

For our example we will use our model with a Bayes error rate of $q = 0.1$, a training sample size of $n = 200$ and $d = 20$ dimensions of which $J = 5$ are active. Figure 1 displays the misclassification error of AdaBoost based on hold out samples of size 1000 (also drawn *iid* on $[0,1]^d$) as a function of the iterations. The results are averaged over 100 repetitions of the simulation. While AdaBoost with stumps (thick, black curve) leads to overfitting very early on, AdaBoost with 8-node trees (thin, red curve) does not suffer from overfitting and leads to smaller misclassification error. In fact, the misclassification error by 1000 iterations was smaller for the 8-node trees in 96 of the 100 simulations. The average (paired) difference in misclassification error was 0.031 with a standard error of $0.018/\sqrt{100} = 0.0018$. Also note that both algorithms here perform considerably worse than the Bayes error rate of $q = 0.1$.

The R code for this experiment as well as all others in this paper can be found at http://www.davemease.com/contraryevidence. We encourage the reader to appreciate the reproducibility of the qualitative result by running the code for various values of the parameters q, n, d and J.

It is worth further commenting on the fact that in this simulation AdaBoost with stumps leads to overfitting while AdaBoost with the larger 8-node trees does not, at least by 1000 iterations. This is of special interest since many of the examples other researchers provide to show AdaBoost can in fact overfit often use very small trees such as stumps as the base learner. Some such examples of overfitting can be found in Friedman et al. (2000a), Jiang (2000) and Ridgeway (2000) as well as Leo Breiman's 2002 Wald Lectures on Machine Learning.[1] The belief is that if stumps overfit then so will larger trees since the larger trees are more complex. (Clearly the example presented

1. Breiman's lecture notes can be found at http://www.stat.berkeley.edu/users/breiman/wald2002-1.pdf.

in this section shows that this is not the case.) To illustrate this viewpoint consider the quote from Jiang (2001) who writes, "all these base systems, even the ones as simple as the 'stumps', will unavoidably lead to suboptimal predictions when boosted forever." Additionally, such examples in which overfitting is observed also often deal with extremely low-dimensional cases such as $d = 2$ or even $d = 1$. By experimenting with the simulation code provided along with this paper one can confirm that in general AdaBoost is much more likely to suffer from overfitting in trivial low-dimensional examples as opposed to high-dimensional situations where it is more often used.

3.2 Should Smaller Trees Be Used When the Bayes Error is Larger?

Similar arguments to those in the previous section suggest that it is necessary to use smaller trees for AdaBoost when the Bayes error is larger. The reasoning is that when the Bayes error is larger, the larger trees lead to a more complex model which is more susceptible to overfitting noise. However, in practice we can often observe the opposite to be true. The higher Bayes error rate actually can favor the larger trees. This counterintuitive result may be explained by the self-averaging which occurs during the boosting iterations as discussed by Krieger et al. (2001). Conversely, the smaller trees often work well for lower Bayes error rates, provided they are rich enough to capture the complexity of the signal.

We illustrate this phenomenon by re-running the experiment in the previous section, this time using $q = 0$, which implies the Bayes error is zero. The average misclassification error over the 100 hold out samples is displayed in the top panel of Figure 2. It can now be observed that AdaBoost with stumps performs better than AdaBoost with 8-node trees. In fact, this was the case in 81 out of the 100 simulations (as opposed to only 4 of the 100 for $q = 0.1$ from before). The mean difference in misclassification error after 1000 iterations was 0.009 with a standard error of $0.011/\sqrt{100} = 0.0011$. The bottom panel of Figure 2 confirms that AdaBoost with stumps outperforms AdaBoost with 8-node tress only for very small values of q with this simulation model.

3.3 Should LogitBoost Be Used Instead of AdaBoost for Noisy Data?

The LogitBoost algorithm was introduced by Friedman et al. (2000a). The algorithm is similar to AdaBoost, with the main difference being that LogitBoost performs stagewise minimization of the negative binomial log likelihood while AdaBoost performs stagewise minimization of the exponential loss. By virtue of using the binomial log likelihood instead of the exponential loss, the LogitBoost algorithm was believed to be more "gentle" and consequently likely to perform better than AdaBoost for classification problems in which the Bayes error is substantially larger than zero. For instance, on page 309 Hastie et al. (2001) write, "it is therefore far more robust in noisy settings where the Bayes error rate is not close to zero, and especially in situations where there is misspecification of the class labels in the training data."

Despite such claims, we often observe the opposite behavior. That is, when the Bayes error is not zero, LogitBoost often overfits while AdaBoost does not. As an example, we consider the performance of AdaBoost and LogitBoost on the simulation from Section 3.1 in which the Bayes error was $q = 0.1$. The base learners used are 8-node trees. Figure 3 displays the performance averaged over 100 hold out samples. It is clear that LogitBoost (blue, thick) begins to overfit after about 200 iterations while AdaBoost (red, thin) continues to improve. After 1000 iterations the mean difference was 0.031 with a standard error of $0.017/\sqrt{100}$=0.0017. The misclassification error for LogitBoost at 1000 iterations was larger than that of AdaBoost in all but 4 of the 100 simulations.

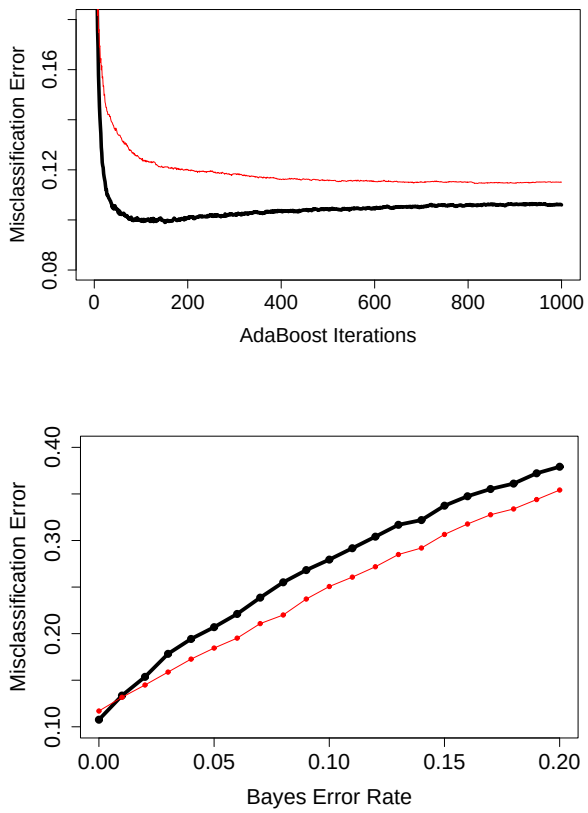


Figure 2: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule. Top Panel: Misclassification Error for Zero Bayes Error as a Function of the Iterations. Bottom Panel: Misclassification Error at 1000 Iterations as a Function of the Bayes Error Rate q.

Other examples of this phenomenon of LogitBoost overfitting noisy data when AdaBoost does not can be found in Mease et al. (2007).

The R code used for LogitBoost was written by Marcel Dettling and Peter Buhlmann and can be found at http://stat.ethz.ch/∼dettling/boosting.html. Two small modifications were made to the code in order to fit 8-node trees, as the original code was written for stumps.

It should be noted that LogitBoost differs from AdaBoost not only in the loss function which it minimizes, but also in the Newton style minimization that it employs to carry out the minimization. For this reason it would be of interest to examine the performance of the algorithm in Collins et al. (2000) which minimizes the negative binomial log likelihood in a manner more analogous to AdaBoost. We do not consider that algorithm in this paper since our focus is mainly on the work of Friedman et al. (2000a) and the implications in the statistical community.

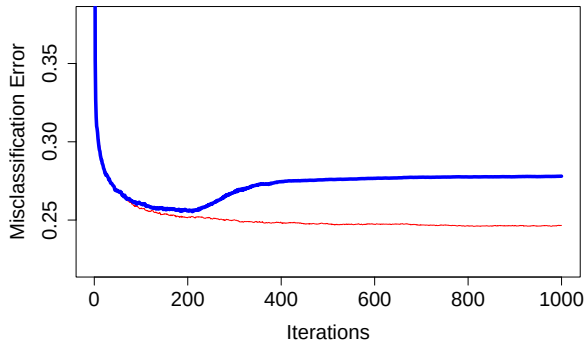


Figure 3: Comparison of AdaBoost (Red, Thin) and LogitBoost (Blue, Thick) with 8-Node Trees

3.4 Should Early Stopping Be Used to Prevent Overfitting?

In order to prevent overfitting, one popular regularization technique is to stop boosting algorithms after a very small number of iterations, such as 10 or 100. The statistics community has put a lot of emphasis on early stopping as evidenced by the large number of papers on this topic. For example, the paper "Boosting with Early Stopping: Convergence and Consistency" by Zhang and Yu (2005) tells readers that "boosting forever can overfit the data" and that "therefore in order to achieve consistency, it is necessary to stop the boosting procedure early." Standard implementations of boosting such as the popular gbm package for R by Ridgeway (2005) implement data-derived early stopping rules.

The reasoning behind early stopping is that after enough iterations have occurred so that the complexity of the algorithm is equal to the complexity of the underlying true signal, then any additional iterations will lead to overfitting and consequently larger misclassification error. However, in practice we can often observe that additional iterations beyond the number necessary to match the complexity of the underlying true signal actually reduce the overfitting that has already occurred rather than causing additional overfitting. This is likely due to the self-averaging property of AdaBoost to which we eluded earlier.

To illustrate this we use a somewhat absurd example. We take $J = 0$ in our simulation model, so that there is no signal at all, only noise. We have $\mathrm{P}(Y = 1|x) \equiv q$ so that Y does not depend on x in any way. We take a larger sample size of $n = 5000$ this time, and also use larger $2^8 = 256$-node trees. The experiment is again averaged over 100 repetitions, each time drawing the $n = 5000$ x values from $[0,1]^d$ with $d = 20$. The 100 hold out samples are also drawn from $[0,1]^{20}$ each time. The Bayes error rate is $q = 0.2$.

Since there is no signal to be learned, we can observe directly the effect of AdaBoost's iterations on the noise. We see in Figure 4 that early on there is some overfitting, but this quickly goes away and the misclassification error decreases and appears to asymptote very near the Bayes error rate of $q = 0.2$. In fact, the final average after 1000 iterations (to three decimals accuracy) is 0.200 with a standard error of $0.013/\sqrt{100}$=0.0013. Even more interesting, the misclassification error after 1000 iterations is actually less than that after the first iteration (i.e., the misclassification error for a single 2^8-node tree). The mean difference between the misclassification error after one iteration and that after 1000 iterations was 0.012 with a standard error of $0.005/\sqrt{100}$=0.0005. The difference was

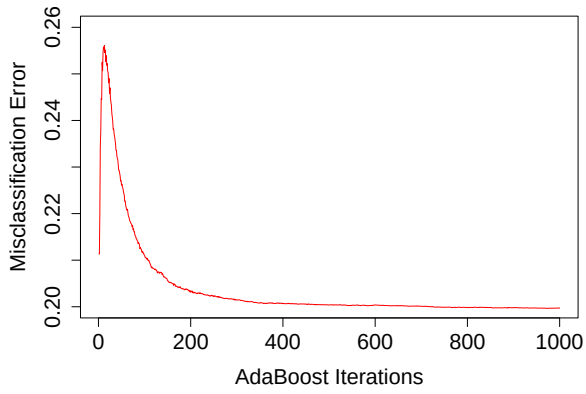


Figure 4: AdaBoost on 20% Pure Noise

positive in 99 of the 100 repetitions. Thus we see that not only does AdaBoost resist overfitting the noise, it actually fits a classification rule that is less overfit than its own 2^8-node tree base classifier.

3.5 Should Regularization Be Based on the Loss Function?

Since the statistical view of boosting centers on the stagewise minimization of a certain loss function on the training data, a common suggestion is that regularization should be based on the behavior of that loss function on a hold out or cross-validation sample. For example, the implementation of the AdaBoost algorithm in the gbm package (Ridgeway, 2005) uses the exponential loss $\sum_{i=1}^{n} e^{-y_i F_m(x_i)}$ to estimate the optimal stopping time. Indeed, if early stopping is to be used as regularization, the statistical view would suggest stopping when this exponential loss function begins to increase on a hold out sample. However, in practice the misclassification error often has little to do with the behavior of the exponential loss on a hold out sample. To illustrate this, we return to the experiment in Section 3.1. If we examine the exponential loss on hold out samples for AdaBoost with the 8-node trees, it can be seen that this loss function is exponentially increasing throughout the 1000 iterations. This is illustrated in Figure 5 which shows the linear behavior of the log of the exponential loss for a single repetition from this experiment on a hold out sample of size 1000. Thus, early stopping regularization based on the loss function would suggest stopping after just one iteration, when in fact Figure 1 shows we do best to run the 8-node trees for the full 1000 iterations. This behavior has also been noted for LogitBoost as well (with respect to the negative log likelihood loss) in Mease et al. (2007) and in Dettling and Buhlmann (2003). In the latter reference the authors estimated a stopping parameter for the number of iterations using cross-validation but observed that they "could not exploit significant advantages of estimated stopping parameters" over allowing the algorithm to run for the full number of iterations (100 in their case).

3.6 Should the Collection of Basis Functions Be Restricted to Prevent Overfitting?

Another popular misconception about boosting is that one needs to restrict the class of trees in order to prevent overfitting. The idea is that if AdaBoost is allowed to use *all* 8-node trees for instance, then the function class becomes too rich giving the algorithm too much flexibility which

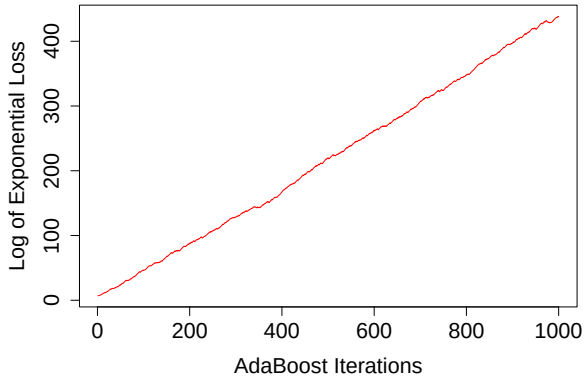


Figure 5: The Log of the Exponential Loss for AdaBoost on a Hold Out Sample

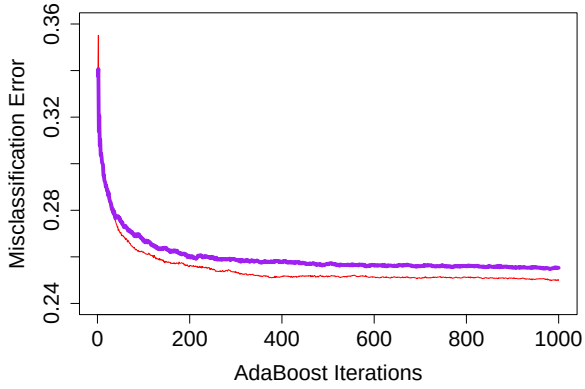


Figure 6: Comparison of AdaBoost with 8-Node Trees (Red, Thin) to AdaBoost with 8-Node Trees Restricted to Have at Least 15 Observations in Each Terminal Node (Purple, Thick)

leads to overfitting. This line of thinking gives rise to various methods for restricting or regularizing the individual trees themselves as a method of regularizing the AdaBoost algorithm. For instance, the implementation of AdaBoost in the gbm code (Ridgeway, 2005) has a parameter called "n.minobsinnode" which is literally the minimum number of observations in the terminal nodes of the trees. The default of this value is not 1, but 10.

In spite of this belief, it can be observed that the practice of limiting the number of observations in the terminal nodes will often degrade the performance of AdaBoost. It is unclear why this happens; however, we note that related tree ensemble algorithms such as PERT (Cutler and Zhao, 2001) have demonstrated success by growing the trees until only a single observation remains in each terminal node.

As an example of this performance degradation, we again revisit the simulation in Section 3.1 and compare the (unrestricted) 8-node trees used there to 8-node trees restricted to have at least 15

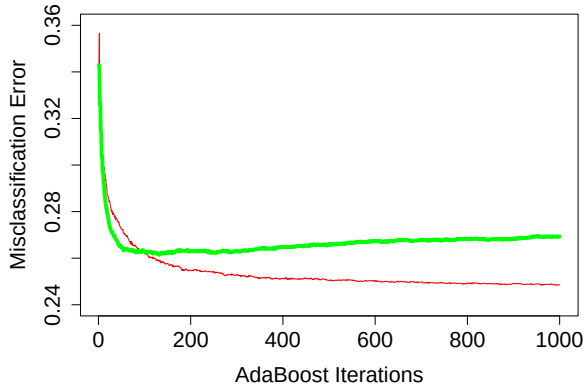


Figure 7: Comparison of AdaBoost (Red, Thin) and AdaBoost with Shrinkage (Green, Thick)

observations in each terminal node. (This is done in R by using the option "minbucket=15" in the "rpart.control" syntax.) Figure 6 shows the results with the unrestricted 8-node trees given by the red (thin) curve and the restricted 8-node trees given by the purple (thick) curve. The degradation in performance is evident, although not extremely large. The mean difference in misclassification error at 1000 iterations was 0.005 with a standard error of $0.010/\sqrt{100}$=0.001. AdaBoost with unrestricted 8-node trees gave a lower misclassification error in 67 of the 100 repetitions.

3.7 Should Shrinkage Be Used to Prevent Overfitting?

Shrinkage is yet another form of regularization that is often used for boosting algorithms. In the context of AdaBoost, shrinkage corresponds to replacing the α_m in the update formula $F_m = F_{m-1} + \alpha_m g_m$ by $\nu\alpha_m$ where ν is any positive constant less than one. The value $\nu = 0.1$ is popular. In the statistical view of boosting, shrinkage is thought to be extremely important. It is believed to not only reduce overfitting but also to increase the maximum accuracy (i.e., the minimum misclassification error) over the iterations. For instance, Friedman et al. (2000b) write, "the evidence so far indicates that the smaller the value of ν, the higher the overall accuracy, as long as there are enough iterations."

Despite such claims, it can be observed that shrinkage often does not improve performance and instead can actually cause AdaBoost to overfit in situations where it otherwise would not. To understand why this happens one needs to appreciate that it is the suboptimal nature of the stagewise fitting of AdaBoost that helps it to resist overfitting. Using shrinkage can destroy this resistance. For an example, we again revisit the simulation in Section 3.1 using the 8-node trees. In Figure 7 the red (thin) curve corresponds to the misclassification error for the 8-node trees just as in Section 3.1 and the green (thick) curve now shows the effect of using a shrinkage value of $\nu = 0.1$. It is clear that the shrinkage causes overfitting in this simulation. By 1000 iterations shrinkage gave a larger misclassification error in 95 of the 100 repetitions. The mean difference in misclassification error at 1000 iterations was 0.021 with a standard error of $0.012/\sqrt{100}$=0.0012.

3.8 Is Boosting Estimating Probabilities?

The idea that boosting produces probability estimates follows directly from the statistical view through the stagewise minimization of the loss function. Specifically, the exponential loss $\sum_{i=1}^{n} e^{-y_i F_m(x_i)}$, which is minimized at each stage by AdaBoost, achieves its minimum when the function $F_m(x)$ relates to the true conditional class probabilities $p(x) \equiv \mathrm{P}(Y=1|x)$ by the formula $F_m(x) = \frac{1}{2}\log\frac{p(x)}{1-p(x)}$. This leads to the estimator of $p(x)$ after m iterations given by

$$\hat{p}_m(x) = 1/(1+e^{-2F_m(x)}).$$

This relationship between the score function F_m in AdaBoost and conditional class probabilities is given explicitly in Friedman et al. (2000a). An analogous formula is also given for obtaining probability estimates from LogitBoost. Standard implementations of boosting such as Dettling and Buhlmann's LogitBoost code at http://stat.ethz.ch/∼dettling/boosting.html as well as the gbm LogitBoost code by Ridgeway (2005) output conditional class probabilities estimates directly.

Despite the belief that boosting is estimating probabilities, the estimator $\hat{p}_m(x)$ given above is often extremely overfit in many cases in which the classification rule from AdaBoost shows no signs of overfitting and performs quite well. An example is given by the experiment in Section 3.1. In Figure 1 we saw that the classification rule using 8-node trees performed well and did not overfit even by 1000 iterations. Conversely, the probability estimates are severely overfit early on. This is evidenced by the plot of the exponential loss in Figure 5. In this context the exponential loss can be thought of as an estimate of a *probability scoring rule* which quantifies the average disagreement between a true probability p and an estimate $\hat{p}$ using only binary data (Buja et al., 2006). For the exponential loss the scoring rule is $p\sqrt{(1-\hat{p})/\hat{p}} + (1-p)\sqrt{\hat{p}/(1-\hat{p})}$. The fact that the plot in Figure 5 is increasing shows that the probabilities become worse with each iteration as judged by this scoring rule. Similar behavior can be seen using other scoring rules such as the squared loss $(p-\hat{p})^2$ and the log loss $-p\log\hat{p} - (1-p)\log(1-\hat{p})$ as shown in Mease et al. (2007). This reference also shows the same behavior for the probability estimates from LogitBoost, despite the fact that efficient probability estimation is the main motivation for the LogitBoost algorithm.

The reason for the overfitting of these probability estimators is that as more and more iterations are added to achieve a good classification rule, the value of $|F_m|$ at any point is increasing quickly. The classification rule only depends on the sign of F_m and thus is not affected by this. However, this increasing tendency of $|F_m|$ impacts the probability estimates by causing them to quickly diverge to 0 and 1. Figure 8 shows the probability estimates $\hat{p}_m(x_i) = 1/(1+e^{-2F_m(x_i)})$ for AdaBoost from a single repetition of the experiment in Section 3.1 using 8-node trees on a hold out sample of size 1000. The top histogram corresponds to $m = 10$ iterations and the bottom histogram shows $m = 1000$ iterations. The histograms each have 100 equal width bins. It can be seen that after only 10 iterations almost all of the probability estimates are greater than 0.99 or less than 0.01, and even more so by 1000 iterations. This indicates a poor fit since we know all of the true probabilities are either 0.1 or 0.9.

Other researchers have also noted this type of overfitting with boosting and have used it as an argument in favor of regularization techniques. For instance, it is possible that using a regularization technique such as shrinkage or the restriction to stumps as the base learners in this situation could produce better probability estimates. However, from what we have seen of some regularization techniques in this paper, we know that regularization techniques often severely degenerate the classification performance of the algorithm. Furthermore, some are not effective at all in many

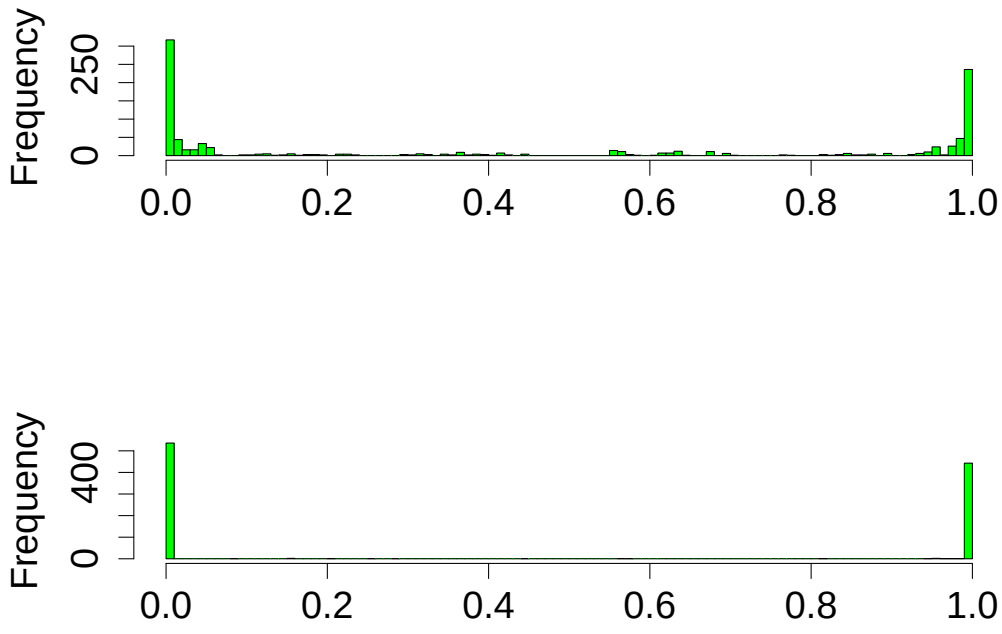


Figure 8: Probability Estimates From AdaBoost at $m = 10$ Iterations (Top) and $m = 1000$ Iterations (Bottom)

situations. For instance, early stopping, one of the most popular regularization techniques, is of little help when the probabilities overfit from the outset as in Figure 5. For a technique that achieves conditional probability estimation using AdaBoost without modification or regularization the reader should see Mease et al. (2007).

3.9 Is Boosting Similar to the One Nearest Neighbor Classifier?

In all the experiments considered in this paper, AdaBoost achieves zero misclassification error on the training data. This characteristic is quite typical of AdaBoost and has led some researchers to draw parallels to the (one) nearest neighbor classifier, a classifier which necessarily also yields zero misclassification error on the training data. This characteristic has also been suggested as a reason why AdaBoost will overfit when the Bayes error is not zero.

The belief in a similarity between boosting and the nearest neighbor classifier was not expressed in the original paper of Friedman et al. (2000a), but rather has been expressed more recently in the statistics literature on boosting by authors such as Wenxin Jiang in papers such as Jiang (2000), Jiang (2001) and Jiang (2002). In Jiang (2000), the equivalence between AdaBoost and the nearest neighbor classifier is established only for the case of $d = 1$ dimension. In the $d = 1$ case, the equivalence is merely a consequence of fitting the training data perfectly (and following Jiang's

convention of using midpoints of the training data for the classification tree splits). However, as we will see from the experiment in this section, the behavior of AdaBoost even in $d = 2$ dimensions is radically different from the nearest neighbor rule.

Despite this difference, Jiang goes on to suggest that the performance of AdaBoost in higher dimensions might be similar to the case of $d = 1$ dimension. For instance in "Is Regularization Unnecessary for Boosting?" Jiang (2001) writes, "it is, however, plausible to conjecture that even in the case of higher dimensional data running AdaBoost forever can still lead to a suboptimal prediction which does not perform much better than the nearest neighbor rule." Further, Jiang (2002) writes, "the fit will be perfect for almost all sample realizations and agree with the nearest neighbor rule at all the data points as well as in some of their neighborhoods" and that "the limiting prediction presumably cannot perform much better than the nearest neighbor rule."

To understand why equivalent behavior on the training data (or "data points" using Jiang's terminology above) does not imply similar performance for classification rules for $d > 1$, it is important to remember that in the case of continuous data the training data has measure zero. Thus the behavior on the training data says very little about the performance with respect to the population. This is well illustrated by the pure noise example from Section 3.4. For any point in the training data for which the observed class differs from the class given by the Bayes rule, both AdaBoost and nearest neighbor will classify this point as the observed class and thus disagree with the Bayes rule. However, the volume of the affected neighborhood surrounding that point can be arbitrarily small with AdaBoost, but will necessarily be close to $1/n$ of the total volume with nearest neighbor.

To help the reader visualize this, we consider a $d = 2$-dimensional version of the pure noise example from Section 3.4. We again use a Bayes error rate of $q = 0.2$ but now take only $n = 200$ points spread out evenly according to a Latin hypercube design. The left plot in Figure 9 shows the resulting classification of AdaBoost using 8-node trees after 1000 iterations and the right plot shows the rule for nearest neighbor. The training points with $Y = -1$ are colored black and those with $Y = +1$ are colored yellow. Regions classified as -1 are colored purple and those classified as $+1$ are colored light blue. Since the Bayes rule is to classify the entire area as -1, we can measure the overfitting of the rules by the fraction of the total area colored as light blue. The nearest neighbor classifier has 20% of the region colored as light blue (as expected), while AdaBoost has only 16%. The two classifiers agree "at all the [training] data points as well as in some of their neighborhoods" as stated by Jiang, but the "some" here is relatively small.

In higher dimensions this effect is even more pronounced. For the $d = 20$-dimensional example from Section 3.4 the area (volume) of the light blue region would be essentially zero for AdaBoost (as evidenced by its misclassification error rate matching almost exactly that of the Bayes error), while for nearest neighbor it remains at 20% as expected. Thus we see that the nearest neighbor classifier differs from the Bayes rule for 20% of the points in both the training data and the population while AdaBoost differs from the Bayes rule for 20% of the points in the training data but virtually none in the population.

The differences between the nearest neighbor classifier and AdaBoost are obvious in the other experiments in this paper as well. For instance, for the experiment in Section 3.1 the nearest neighbor classifier had an average misclassification error rate of 0.376 versus 0.246 for AdaBoost with the 8-node trees.

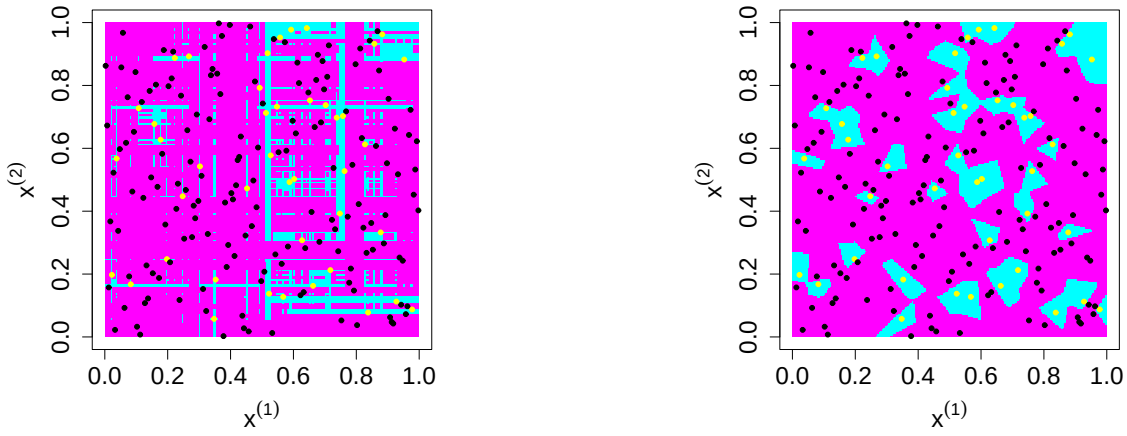


Figure 9: Comparison of AdaBoost (Left) and Nearest Neighbor (Right) on 20% Pure Noise

3.10 Is Boosting Consistent?

An important question to ask about any estimator is whether or not it is consistent. A consistent estimator is defined to be any estimator for which the estimated quantity converges in probability to the true quantity. In our context, to ask if AdaBoost is a consistent estimator is to ask if its classification rule converges in probability to the Bayes rule. If it is consistent, then with a sufficiently large training sample size n its misclassification error will come arbitrarily close to the Bayes error.

The belief in the statistics community is that AdaBoost is not consistent unless regularization is employed. The main argument given is that if AdaBoost is left unregularized it will eventually fit all the data thus making consistency impossible as with the nearest neighbor classifier. Consequently, all work on the consistency of boosting deals with regularized techniques. While we have noted in Section 3.9 that it is characteristic of AdaBoost to achieve zero misclassification error on the training data, we have also discussed the fact that this in no way determines its performance in general, as the training data has measure zero in the case of continuous data. In fact in Section 3.4 we observed that with a sample size of $n = 5000$ AdaBoost with 2^8-node trees achieved the Bayes error rate to three decimals on a 20% pure noise example despite fitting all the training data.

In this section we consider a simulation with this same sample size and again 2^8-node trees but we now include a signal in addition to the noise. We take $J = 1$ and use $d = 5$ dimensions and fix the Bayes error rate at $q = 0.1$. The resulting misclassification error rate averaged over 100 repetitions each with a hold out sample of size 1000 is shown in Figure 10. As before, AdaBoost fits all the training data early on, but the misclassification error after 1000 iterations averages only 0.105 with a standard error of $0.010/\sqrt{100}$=0.001. This is quite close to the Bayes error rate $q = 0.1$ and can be observed to come even closer by increasing the sample size. It should also be noted that this error rate is much below the limit of $2q(1-q) = 0.18$ that holds for the nearest neighbor classifier in this case.

The belief that unregularized AdaBoost can not be consistent is promoted by Wenxin Jiang's work mentioned in Section 3.9 connecting the performance of AdaBoost and the nearest neighbor classifier. His result for $d = 1$ rules out consistency in that case since the nearest neighbor rule is

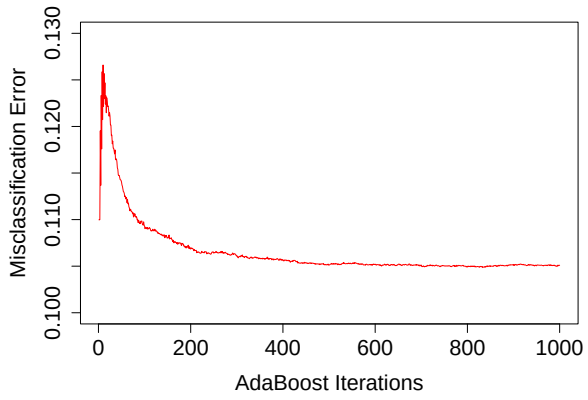


Figure 10: Performance of AdaBoost for a Simulation with a Bayes Error of 0.1

not consistent, but nothing is established for $d > 1$ with regard to AdaBoost. Jiang (2002) admits this when he writes, "what about boosting forever with a higher dimensional random continuous predictor x with $\dim(x) > 1$? We do not have theoretical results on this so far."

4. More Experiments Which Contradict the Statistical View of Boosting

In this section we revisit the experiments from Section 3 using a different simulation model. The purpose here is to show that the results are reproducible and do not depend on a particular simulation model. We also encourage readers to experiment with other simulation models by modifying the code provided on the web page.

The simulations in this section will use the model

$$P(Y=1|x) = \begin{cases} q & x^{(1)} \in [0,0.1) \cup [0.2,0.3) \cup [0.4,0.5) \cup [0.6,0.7) \cup [0.8,0.9) \\ 1-q & x^{(1)} \in [0.1,0.2) \cup [0.3,0.4) \cup [0.5,0.6) \cup [0.7,0.8) \cup [0.9,1]. \end{cases}$$

We will rerun each experiment from Section 3 using this model. Throughout this section we will use $d = 20$ dimensions and take X to be distributed *iid* uniform on the 20-dimensional unit cube $[0,1]^{20}$. For each experiment we will use twice the sample size of the analogous experiment in Section 3 and the same Bayes error q. The single exception will be the experiment in Section 4.9 in which we use a Bayes error of $q = 0.1$ and $d = 2$ dimensions for visualization purposes.

Note that while the experiments in Section 3 had $J \leq d$ effective dimensions, the experiments in this section will all have only one effective dimension as a result of this simulation model. The plots in Figure 19 are useful for visualizing this model in $d = 2$ dimensions.

4.1 Should Stumps Be Used for Additive Bayes Decision Rules?

As in Section 3.1 we use a Bayes error rate of $q = 0.1$ and $d = 20$ dimensions. We use the new simulation model with a training sample size of $n = 400$. Figure 11 displays the misclassification error of AdaBoost based on hold out samples of size 1000 (also drawn *iid* on $[0,1]^{20}$) as a function of the iterations. The results are again averaged over 100 repetitions of the simulation.

As in Section 3.1, Adaboost with 8-node trees (thin, red curve) does not show any signs of overfitting while AdaBoost with stumps (thick, black curve) leads to overfitting. The overfitting

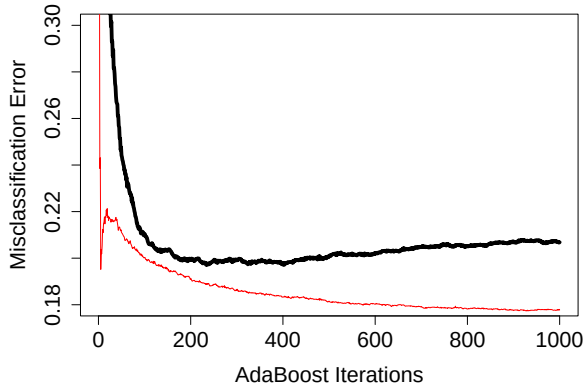


Figure 11: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule

is evident in this experiment after about 400 iterations. Furthermore, AdaBoost with 8-node trees outperforms AdaBoost with stumps throughout the entire 1000 iterations. The misclassification error by 1000 iterations was smaller for the 8-node trees in 93 of the 100 simulations. The average (paired) difference in misclassification error after 1000 iterations was 0.029 with a standard error of $0.018/\sqrt{100} = 0.0018$. As before, since the simulation model used here has an additive Bayes decision rule, this evidence is directly at odds with the recommendation in Hastie et al. (2001) and Friedman et al. (2000a) that stumps are preferable for additive Bayes decision rules.

4.2 Should Smaller Trees Be Used When the Bayes Error is Larger?

As in Section 3.2, we observe that when we decrease the Bayes error rate from $q = 0.1$ to $q = 0$, the 8-node trees no longer have an advantage over the stumps. Figure 12 displays the results of the simulation in Section 4.1 using a Bayes error rate of $q = 0$. We see that the advantage of the 8-node trees has completely disappeared, and now the 8-node trees and stumps are indistinguishable. By 1000 iterations the misclassification errors for both are identical in all of the 100 repetitions.

Thus we see that the advantage of the larger trees in Section 4.1 is a result of the non-zero Bayes error, again suggesting that larger trees are in some way better at handling noisy data. This directly contradicts the conventional wisdom that boosting with larger trees is more likely to overfit on noisy data than boosting with smaller trees.

4.3 Should LogitBoost Be Used Instead of AdaBoost for Noisy Data?

We now rerun the experiment in Section 4.1 using AdaBoost and LogitBoost both with 8-node trees. Figure 13 displays the results with AdaBoost in red (thin) and LogitBoost in blue (thick). While LogitBoost performs better early on, it eventually suffers from overfitting near 400 iterations while AdaBoost shows no overfitting. Furthermore, the misclassification error for AdaBoost after 1000 iterations is (slightly) lower than the minimum misclassification error achieved by LogitBoost. After 1000 iterations the mean difference in misclassification error between LogitBoost and AdaBoost

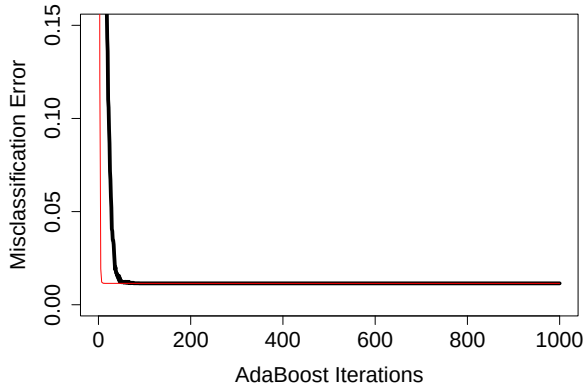


Figure 12: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for an Additive Bayes Rule with Zero Bayes Error

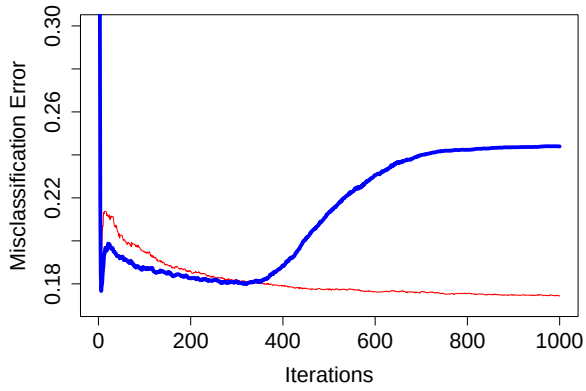


Figure 13: Comparison of AdaBoost (Red, Thin) and LogitBoost (Blue, Thick) with 8-Node Trees

was 0.069 with a standard error of $0.021/\sqrt{100}$=0.0021. The misclassification error for LogitBoost at 1000 iterations was larger than that of AdaBoost in all of the 100 repetitions.

Thus we again see that although LogitBoost was invented to perform better than AdaBoost for data with non-zero Bayes error, LogitBoost actually overfits the data while AdaBoost does not.

4.4 Should Early Stopping Be Used to Prevent Overfitting?

In this section we repeat the simulation from Section 3.4 using the new simulation model. Just as in Section 3.4 we use large $2^8 = 256$-node trees, a Bayes error rate of $q = 0.2$ and $d = 20$ dimensions. We now take twice the training sample size of Section 3.4 so that we have $n = 10,000$ points.

Figure 14 shows the resulting misclassification error averaged over 100 repetitions for hold out samples of size 1000. Although there is overfitting early on, the best performance is again achieved

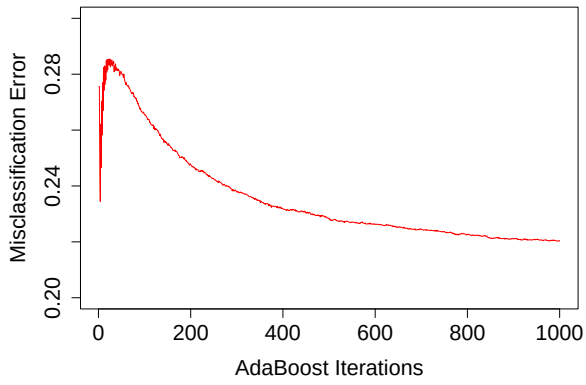


Figure 14: AdaBoost with 20% Bayes Error Using 256-Node Trees

by running the algorithm for the full 1000 iterations. We note that conventional early stopping rules here would be especially harmful since they would stop the algorithm after only a few iterations when the overfitting first takes place. Consequently any such early stopping rule would miss the optimal rule of running for the full 1000 iterations.

It should also be noted that the $2^8 = 256$-node trees used here are much richer than needed to fit the simple one-dimensional Bayes decision rule for this simulation model. Despite this, the misclassification error after 1000 iterations was lower than the misclassification error after the first iteration in all 100 of the reptitions. Thus it is again the self-averaging property of boosting that improves the performance as more and more iterations are run. Early stopping in this example would destroy the benefits of this property.

4.5 Should Regularization Be Based on the Loss Function?

As discussed in Section 3.5, regularization techniques for boosting such as early stopping are often based on minimizing a loss function such as the exponential loss in the case of AdaBoost. However, the performance of AdaBoost with regard to misclassification loss often has very little to do with the exponential loss function in practice.

In this section we examine the exponential loss for the experiment in Section 4.1 using 8-node trees. Figure 15 shows the increasing linear behavior for the log of the exponential loss for a single repetition of this experiment with a hold out sample of size 1000. Thus, just as in Section 3.5, the exponential loss increases exponentially as more iterations are run, while the misclassification error continues to decrease. Choosing regularization to minimize the exponential loss is again not useful for minimizing the misclassification error.

4.6 Should the Collection of Basis Functions Be Restricted to Prevent Overfitting?

In Section 3.6 we saw that restricting the number of observations in the terminal nodes of the trees to be at least 15 degraded the performance of AdaBoost, despite the common belief that such restrictions should be beneficial. In this section we rerun the experiment in Section 4.1 but again consider this same restriction.

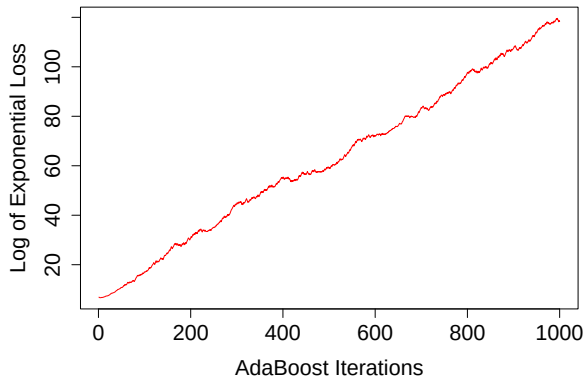


Figure 15: The Log of the Exponential Loss for AdaBoost on a Hold Out Sample

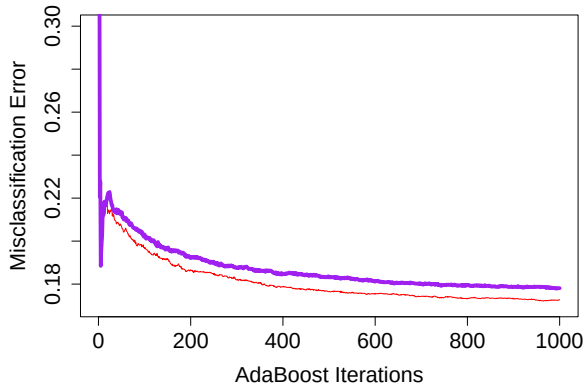


Figure 16: Comparison of AdaBoost with 8-Node Trees (Red, Thin) to AdaBoost with 8-Node Trees Restricted to Have at Least 15 Observations in the Terminal Nodes (Purple, Thick)

Figure 16 shows the results with the unrestricted 8-node trees given by the red (thin) curve and the 8-node trees restricted to have at least 15 observations in the terminal nodes given by the purple (thick) curve. As in Section 3.6, degradation in performance is evident. The mean difference in misclassification error at 1000 iterations was 0.005 with a standard error of $0.010/\sqrt{100}$=0.001. AdaBoost with unrestricted 8-node trees gave a lower misclassification error at 1000 iterations in 65 of the 100 repetitions for this simulation model.

4.7 Should Shrinkage Be Used to Prevent Overfitting?

In Section 3.7 we saw that shrinkage actually caused AdaBoost to overfit in a situation where it otherwise would not have, in spite of the popular belief that shrinkage prevents overfitting. In this section we rerun the experiment in Section 4.1 with 8-node trees again using a shrinkage value of

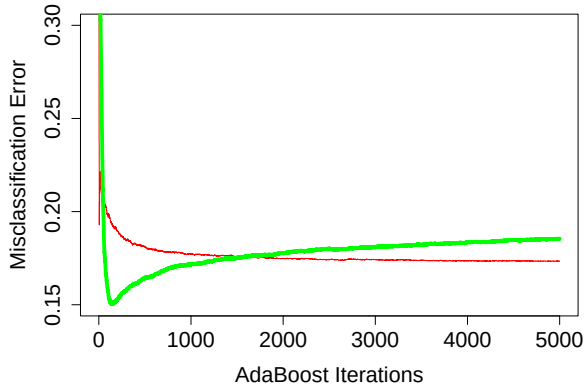


Figure 17: Comparison of AdaBoost (Red, Thin) and AdaBoost with Shrinkage (Green, Thick)

$\nu = 0.1$. Figure 17 shows the results with the red (thin) curve corresponding to no shrinkage and the green (thick) curve showing the results for shrinkage. The plot shows that again shrinkage causes overfitting.

It is interesting to note that in this simulation, unlike the simulation in Section 3.7, shrinkage has the beneficial effect of producing a lower misclassification error very early on in the process, despite causing the eventual overfitting. This suggests that a stopping rule which could accurately estimate the optimal number of iterations combined with shrinkage may prove very effective for this particular simulation. As a result of the good performance early on, the shrinkage actually gives a lower misclassification error at our chosen stopping point of 1000 iterations than without the shrinkage. However, if we run for enough iterations (the plot shows 5000 iterations) the overfitting caused by the shrinkage eventually overwhelms this advantage. By 5000 iterations the shrinkage leads to a larger misclassification error in 87 of the 100 repetitions. The mean difference in misclassification error at 5000 iterations was 0.012 with a standard error of $0.012/\sqrt{100}$=0.0012.

4.8 Is Boosting Estimating Probabilities?

In Section 3.8 we saw that the probability estimates suggested by Friedman et al. (2000a) for AdaBoost diverge quickly to 0 and 1 and consequently perform very poorly even for cases where the AdaBoost classification rule performs well. In this section we examine the probability estimates for a single repetition of the experiment in Section 4.1 on a hold out sample of size 1000.

The two histograms in Figure 18 show the resulting probability estimates for $m = 10$ iterations and $m = 1000$ iterations respectively using 8-node trees. Both histograms have 100 equal width bins. At 10 iterations the estimates have not yet diverged, but by 1000 iterations almost all of the probability estimates are greater than 0.99 or less than 0.01, just as we saw in Section 3.8. As before, this indicates a poor fit since with this simulation model all of the true probabilities are either 0.1 or 0.9.

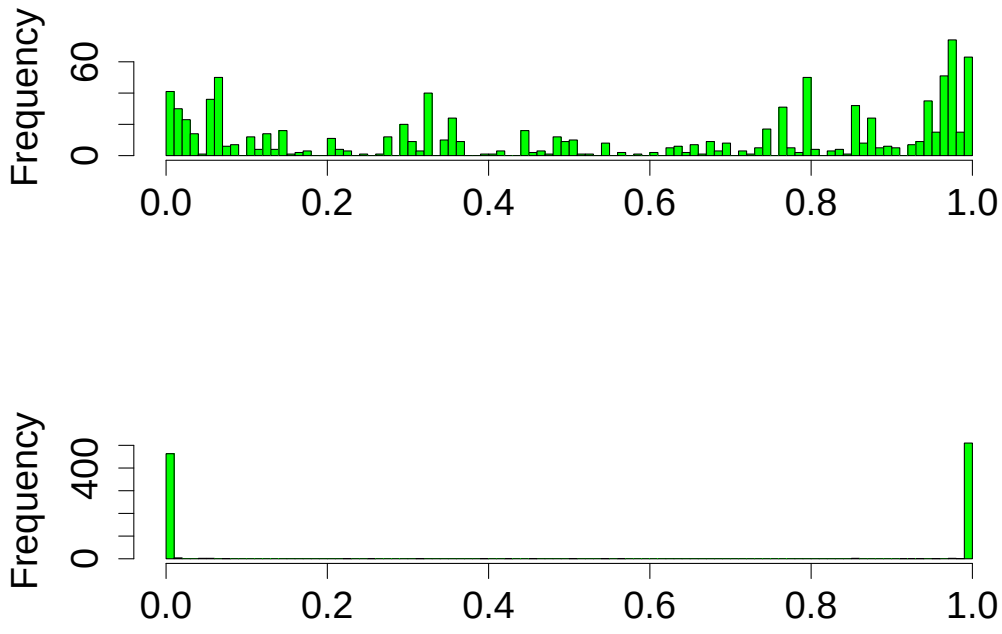


Figure 18: The Probability Estimates From AdaBoost at $m = 10$ Iterations (Top) and $m = 1000$ Iterations (Bottom)

4.9 Is Boosting Similar to the One Nearest Neighbor Classifier?

In Section 3.9 we saw that despite the fact that boosting agrees with the nearest neighbor classifier on all the training data, its performance elsewhere is quite different for $d > 1$ dimensions. For AdaBoost, areas surrounding points in the training data for which the observed class differs from that of the Bayes rule are classified according to the Bayes rule more often than they would be using the nearest neighbor rule.

We illustrate this again using $d = 2$ dimensions for visualization purposes. We use a Bayes error rate of $q = 0.1$ and take $n = 400$ points spread out evenly according to a Latin hypercube design. The plot on the left in Figure 19 shows the resulting classification rule of AdaBoost with 8-node trees at 1000 iterations for a single repetition using the new simulation model. The plot on the right shows the nearest neighbor rule. Both plots use the same color scheme as Figure 9. For the nearest neighbor rule, 21% of the points in the hold out sample disagree with the Bayes rule. This number is only 6% for AdaBoost, despite the fact that both classifiers classify every point in the training data according to the observed class label.

The difference between AdaBoost and the nearest neighbor rule is also well illustrated by other experiments in Section 4. For instance, in Section 4.1 the misclassification error for the nearest neighbor classifier was 0.499 but only 0.178 for AdaBoost with the 8-node trees.

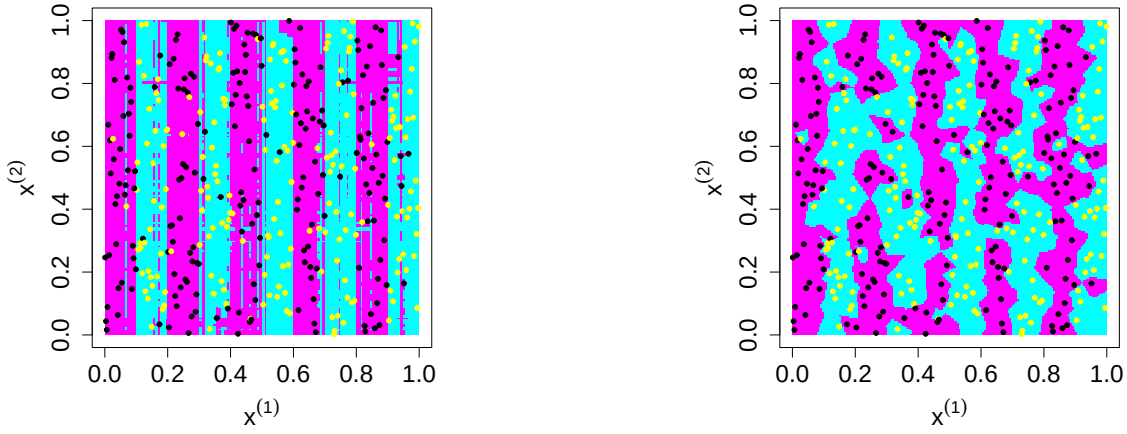


Figure 19: Comparison of AdaBoost (Left) and Nearest Neighbor (Right) with 10% Bayes Error

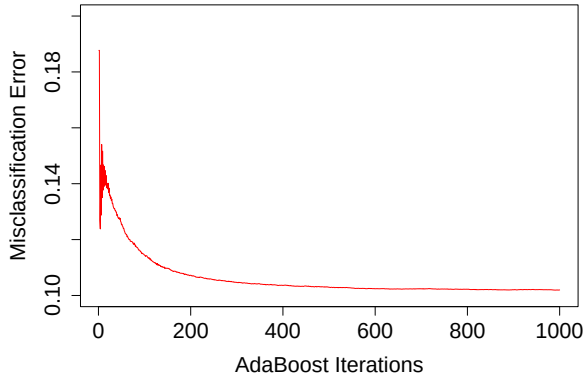


Figure 20: Performance of AdaBoost for a Simulation with a Bayes Error of 0.1

4.10 Is Boosting Consistent?

In Section 3.10 we illustrated that with a large sample size n, the misclassification error for AdaBoost can come quite close to the Bayes error rate, despite the fact that AdaBoost fits the training data perfectly. We illustrate this again in this section. As in Section 3.10, we use 2^8-node trees and a Bayes error rate of $q = 0.1$ but now take $n = 10,000$ and use the new simulation model.

Figure 20 shows the misclassification error averaged over 100 repetitions using hold out samples of size 1000. The mean misclassification error after 1000 iterations was 0.102 with a standard error of $0.009/\sqrt{100}$=0.0009. As we saw in Section 3.10, this is extremely close to the Bayes error rate and much less than the nearest neighbor bound of $2q(1-q) = 0.18$. We encourage readers to rerun the simulation with larger n to make the misclassification error even closer to the Bayes error.

5. Additional Experiments Which Contradict the Statistical View of Boosting

As mentioned at the beginning of Section 4, we encourage the reader to try simulation models other than those considered in this paper by using the R code provided on the web page http://www.davemease.com/contraryevidence. The simulation model can be specified by changing only three lines of this code in most cases. We have only considered two simulation models in this paper due to space constraints.

One criticism of the two simulation models considered in this paper is that both have a discontinuous (piecewise constant) conditional class probability function $p(x) \equiv \mathrm{P}(Y = 1|x)$. An argument can be made that both AdaBoost and LogitBoost can not provide a good fit to these models because of the discontinuities. To investigate this, we examined additional experiments from the simulation model specified by

$$p(x) = 1/(1 + e^{k(\sum_{j=1}^{J} x^{(j)} - J/2)})$$

where J is the number of effective dimensions as in Section 3 and k is a constant which determines the Bayes error rate. We note that this model has the same Bayes decision boundary as the model in Section 3 but now has a smooth conditional class probability function without any discontinuities. The results for this model are not included in the paper but are qualitatively extremely similar to those in Section 3. We encourage the reader to investigate this further.

6. Concluding Remarks and Practical Suggestions

By way of the simulations in Sections 3 and 4 we have seen that there are many problems with the statistical view of boosting and practical suggestions arising from that view. We do not endeavor to explain in this paper why these inconsistencies exist, nor do we offer a more complete view of boosting. Simply put, the goal of this paper has been to call into question this view of boosting that has come to dominate in the statistics community. The hope is that by doing so we have opened the door for future research toward a more thorough understanding of this powerful classification technique.

The statistical view of boosting focuses only on one aspect of the algorithm - the optimization. A more comprehensive view of boosting should also consider the stagewise nature of the algorithm as well as the empirical variance reduction that can be observed on hold out samples as with the experiments in this paper. Much insight on such ideas can be gained from reading work by the late Leo Breiman (e.g., Breiman, 2000, 2001) who subsequently abandoned interest in boosting and went on to work on his own classification technique known as Random Forests. The Random Forests algorithm achieves variance reduction directly through averaging as opposed to AdaBoost for which the variance reduction seems to happen accidently.

While we do not offer much in the way of an explanation for the behavior of AdaBoost in this paper, we will conclude with some practical advice in light of the evidence presented. First of all, AdaBoost remains one of, if not *the*, most successful boosting algorithms. One should not assume that newer, regularized and modified versions of boosting are necessarily better. We encourage readers to try standard AdaBoost along with these newer algorithms. If AdaBoost is not available as an option in your preferred software package, it is only a few lines of code to write yourself. Secondly, if classification is your goal, the best way to judge the effectiveness of boosting is by monitoring the misclassification error on hold out (or cross-validation) samples. We have seen that other loss functions are not necessarily indicative of the performance of boosting's classification

rule. Finally, much of the evidence we have presented is indeed counter-intuitive. For this reason, a practitioner needs to keep an open mind when experimenting with AdaBoost. For example, if stumps are causing overfitting, be willing to try larger trees. Intuition may suggest the larger trees will overfit even more, but we have seen that is not necessarily true.

Acknowledgments

D. Mease's research was supported by an NSF-DMS post-doctoral fellowship. The authors are grateful to Andreas Buja and Abba Krieger for their help and guidance.

References

L. Breiman. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:374–377, 2000.

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

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

P. Buhlmann and B. Yu. Boosting with the L_2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.

A. Buja. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:387–391, 2000.

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications. 2006.

M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. In *Computational Learing Theory*, pages 158–169, 2000.

A. Cutler and G. Zhao. Pert: Perfect random tree ensembles. *Computing Science and Statistics*, 33: 490–497, 2001.

M. Dettling and P. Buhlmann. Boosting for tumor classification with gene expression data. *Bioinformatics*, 19:1061–1069, 2003.

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

Y. Freund and R. E. Schapire. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:391–393, 2000.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:337–374, 2000a.

J. Friedman, T. Hastie, and R. Tibshirani. Rejoiner for additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:400–407, 2000b.

A. J. Grove and D. Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 692–699, 1998.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

W. Jiang. Does boosting overfit: Views from an exact solution. *Technical Report 00-03, Department of Statistics, Northwestern University*, 2000.

W. Jiang. Is regularization unnecessary for boosting? In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 57–64, 2001.

W. Jiang. On weak base hypotheses and their implications for boosting regression and classification. *Annals of Statistics*, 30:51–73, 2002.

A. Krieger, C. Long, and A. J. Wyner. Boosting noisy data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 274–281, 2001.

G. Lugosi and N. Vayatis. On the bayes-risk consistency of regularized boosting methods. *Annals of Statistics*, 32:30–55, 2004.

D. Mease, A. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8:409–439, 2007.

G. Ridgeway. Discussion of additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:393–400, 2000.

G. Ridgeway. Generalized boosted models: A guide to the gbm package. 2005.

R. E. Schapire, Y. Freund, P. L. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26:1651–1686, 1998.

T. Zhang and B. Yu. Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, 33:1538–1579, 2005.

Journal of Machine Learning Research 9 (2008) 157-164 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Kristin P. Bennett BENNEK@RPI.EDU
Department of Mathematical Sciences
Rensselear Polytechnic Institute
Troy, NY 12180, USA

Editor: Yoav Freund

1. Introduction

Mease and Wyner (MW) argue experimentally that the statistical view of boosting does not account for its success and that following the now conventional wisdom arising from this view in Friedman et al. (2000) does not necessarily lead to choices in the boosting algorithm that improve generalization. The authors did an excellent job of defining a set of experiments in which small changes in the boosting algorithm (such as changing the hypothesis space, loss function, and shrinkage) produce significant changes in generalization that were unintuitive given the statistical view of AdaBoost (Freund and Schapire, 1996) expressed in Friedman et al. (2000).

The authors state "The statistical view focuses only on one aspect of the algorithm - the optimization." But one can argue just the opposite, that some of the problems and surprises come from not enough of the optimization perspective instead of too much. Analyzing AdaBoost's performance as an optimization algorithm in terms of convergence rates and optimality conditions (measured on the training data) can be quite revealing. First, we observe that the experiments in MW make dramatic changes in the convergence rates of AdaBoost and that these convergence rates are closely associated with the margin of the classifier. AdaBoost may avoid overfitting for two completely different reasons. Sometimes the algorithm is converging so slowly that stopping at a large number of iterations is still early stopping. At other times, AdaBoost converges relatively quickly and is in essence "overtrained" way past reasonable measures of the optimality conditions. In this case the classifier has converged and is no longer changing much, so the classifier does not overfit. Indeed, some overtraining appears to help improve the classifier slightly. Second, we observe that AdaBoost cannot be trained forever. For the separable case, overtraining AdaBoost and LogitBoost will eventually produce numeric problems that can produce artifacts in the generalization error. In Experiments 3.3 and 4.3 in MW, LogitBoost was overtrained to the point of failure. The so called overfitting observed for LogitBoost was really an algorithmic issue that is quite fixable. If LogitBoost is stopped appropriately or another stepsize strategy is used, the results for LogitBoost are as good as or better than those for AdaBoost. More discussion of these results can be found below.

2. A Mathematical Programmer's View of AdaBoost

AdaBoost optimizes a linear combination of weak hypotheses with respect to the exponential loss. AdaBoost is a coordinate descent (CD) algorithm, that iteratively optimizes the problem with respect to one hypothesis at a time using column generation (Bennett et al., 2000). The weak learner seeks the hypothesis that maximizes the inner product with the function gradient (Mason et al., 2000). The convergence properties of such coordinate descent algorithms have been extensively studied in the mathematical programming community and a full analysis of relevant CD results and their extension to the boosting case can be found in Rätsch (2001).

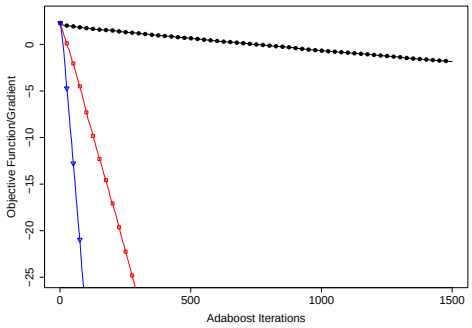
Thus from the mathematical programming perspective, we know AdaBoost inherits both the beneficial and potentially problematic properties of CD. We know from both the CD and original AdaBoost theoretical results that the AdaBoost *objective* converges linearly to the optimal objective. The simplicity of CD and its suitability for column generation make coordinate descent an attractive algorithm, but in practice coordinate descent is not widely used because it can be very slow and it has a tendency to cycle. CD guarantees that the objective function converges to the minimum but there is no guarantee that optimal hypothesis coefficients are attained, and cycling is possible. The AdaBoost loss function is particularly problematic since the exponential function is not strongly convex and the Hessian is rank deficient when the size of the hypothesis space exceeds the number of points. Overall, mathematical programming tells us that we can expect the AdaBoost objective value to converge linearly and the convergence rate to be slow, especially when cycling occurs. The paper on the dynamics of AdaBoost (Rudin et al., 2004) investigates this cycling behavior.

The MW experiments focus on the degenerate case in which the optimal objective value of the underlying exponential optimization problem is zero. LogitBoost and AdaBoost are functions of the form $min_{\alpha} J(f)\ s.t.\ f = Ha$ where H is the hypothesis space matrix containing all possible weak learners for that data set. In every case, there exists some linear combination of weak learners that classifies the points with no error, and therefore the objective can be driven to zero. The AdaBoost exponential loss function is $\sum \exp(-y_i f_i)$. The function space gradient is $\frac{\partial J(f)}{\partial f_i} = -y_i exp(-y_i f_i)$. Note the 1-norm of the function space gradient is the same as the objective, $\left\| \frac{\partial J(f)}{\partial f} \right\|_1 = \sum \exp(-y_i f(x_i))$ for two-class classification. The optimality condition is that the gradient with respect to α is zero, $\frac{\partial J(Ha)}{\partial \alpha} = H' \frac{\partial J(f)}{\partial f} = 0$. In theory, to check this gradient we need to know the weak learners for the full hypothesis space, H. But, for cases where the misclassification error is driven to 0, it is sufficient to monitor the gradient in function space. Fortunately, the norm of the function space gradient provides an upper bound on the norm of the true gradient since $\left\| H' \frac{\partial J(f)}{\partial f} \right\| \leq C \left\| \frac{\partial J(f)}{\partial f} \right\|$ for some fixed $C > 0$.

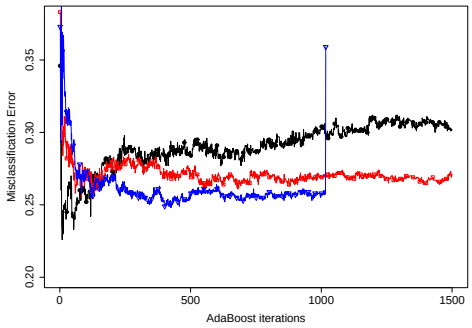
From a mathematical programming perspective, we are optimizing a degenerate, poorly-scaled problem for which the optimal objective value of 0 can only be achieved in the limit using a slower algorithm prone to cycling that may become numerically unstable. Clearly, convergence of the algorithm should be monitored closely. Yet, in most machine learning boosting papers, the focus is on generalization for a fixed number of iterations and rarely on optimization performance.

3. Convergence Rate of AdaBoost

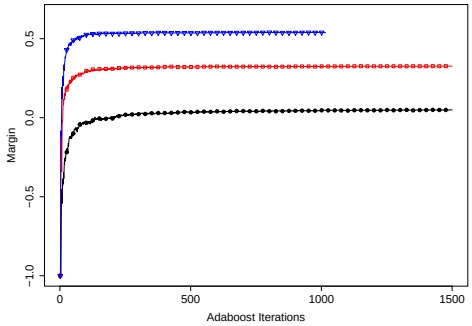
Let's examine the convergence rate and optimality conditions of AdaBoost in the MW experiments. Figure 1 contains three plots, one each for the log base 10 of the objective (or equivalently the 1-



(a) Objective/Gradient (log10)



(b) Test Error

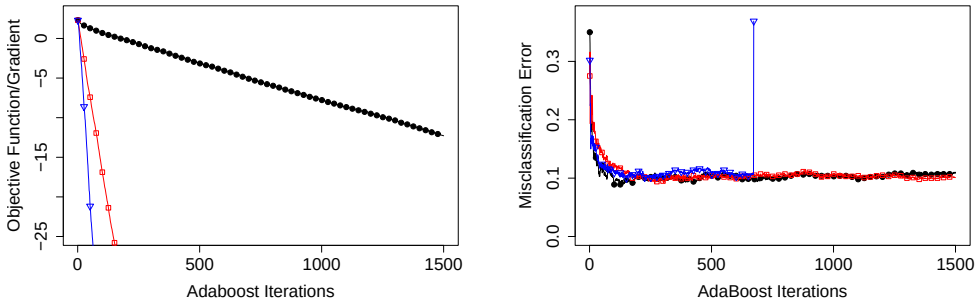


(c) Margin

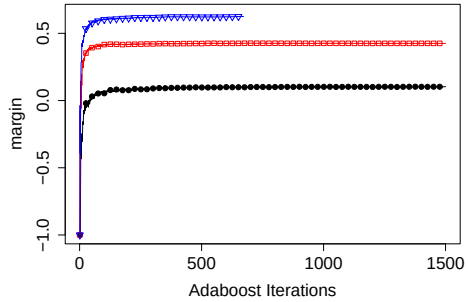
Figure 1: 1 trial of Experiment 3.1 (10% Bayes Error) for AdaBoost + stumps (black, dots), AdaBoost + 8-node trees (red, squares), and AdaBoost + 16-node trees (blue, triangles).

norm of the gradient), the testing error, and the training margin ($\frac{\min_i(y_i f_i)}{\sum_m \alpha_m}$) for 1500 iterations for the first trial of the experiment with 10% Bayes error in section 3.1 of MW. The graphs contain results for AdaBoost with stumps (black, dots), AdaBoost with 8-node trees (red, squares) and AdaBoost with 16-node trees (blue, triangles). Observe that the loss function and gradient are driven to zero for all three hypothesis spaces and that the remarkably different convergence rates are inversely proportional to the size of the trees being boosted. The results for AdaBoost with 16-node trees end at 1017 iterations because the objective becomes less than 10^{-322}, so a divide-by-zero error occurs. In general, AdaBoost with bigger trees achieves bigger margins and obtains better generalization. AdaBoost with stumps converges incredibly slowly and arguably should be run for more than 1500 iterations if early stopping is not desired.

Figure 2 contains the same three graphs for the first trial for the experiment with no Bayes Error in section 3.2. The objective/gradient and margin graphs are qualitatively similar for experiments 3.1 and 3.2. Note that the 16 node tree Adaboost algorithm underflows at 673 iterations. The testing error graph for experiment 3.2 is quite different. The performance for AdaBoost with stumps is much improved and now competitive or better than AdaBoost with 8 or 16 node trees. Here the margin results do not predict which type of boosted trees will generalize best. MW found at least one simple example in which margins don't work well.



a) Objective/Gradient (log10) (b) Test Error



(c) Margin

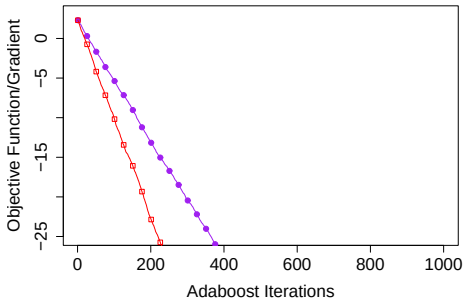
Figure 2: 1 trial of Experiment 3.2 (0 Bayes Error) for AdaBoost + stumps (black, dots), AdaBoost + 8-node trees (red, squares), and AdaBoost + 16-node trees (blue, triangles).

Figure 3 shows the results for AdaBoost with 8-node trees and AdaBoost with 8-node trees restricted to 15 nodes for the first trial of experiment 3.6. Here we see that restricting the trees slows convergence, decreases margins, and increases error.

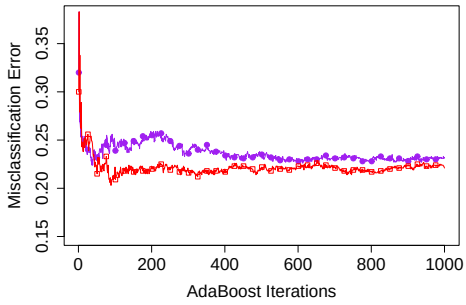
Figure 4 shows the results for AdaBoost with 8-node trees with no shrinkage (red, squares), .1 shrinkage (purple, dots), and .5 shrinkage (blue, triangles) for the first trial of experiment 3.3 (10 % Bayes error). Here we see that shrinkage can speed up or slow down the convergence rates. For .1 shrinkage compared to no shrinkage, the convergence rate was slower, the margin smaller, and the test error larger. For .5 shrinkage, the convergence rate was faster and the margin was larger than for the .1 shrinkage case. If the .5 shrinkage algorithm is terminated at using reasonable stopping criteria, the performance is quite comparable with the no shrinkage case, and improved over the .1 shrinkage case.

We present the following conjectures based on observations of this and other MW experiments for the separable case and leave fuller investigation to later work.

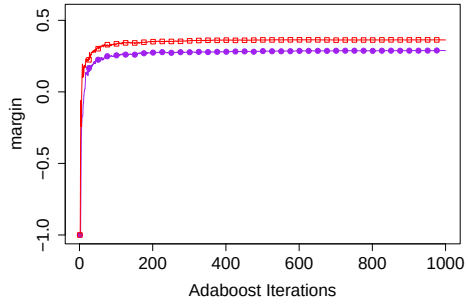
- *The convergence rate of AdaBoost is dependent on the space spanned by the weak learner and larger hypothesis spaces converge faster.* The weak learners produced by stumps are a subset of those from the 8-node decision tree which are in turn a subset of those produced by the 16-node decision tree. The bigger the decision tree, the better the weak learner can match



(a) Objective/Gradient (log10)



(b) Test Error

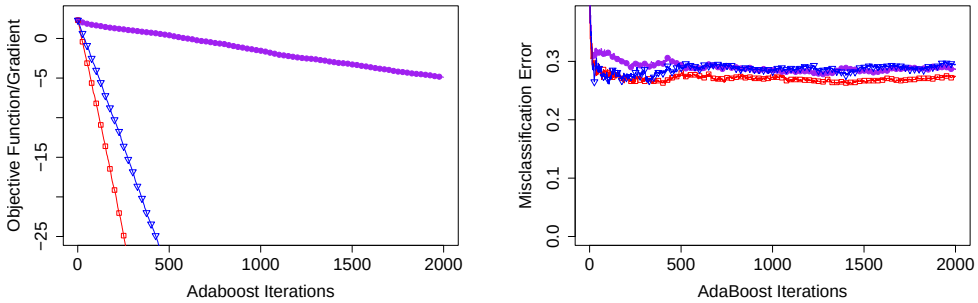


(c) Margin

Figure 3: 1 trial of Experiment 3.6 (10% Bayes Error) for AdaBoost + 8-node trees (red, squares), and AdaBoost + 8-node trees restricted to at least 15 observations in terminal nodes (purple, dots).

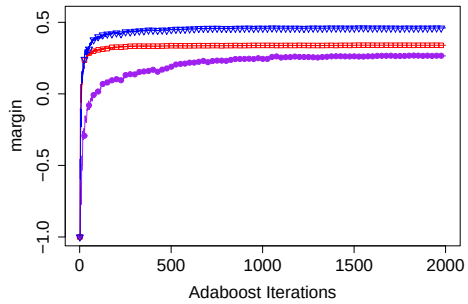
the gradient at each iteration (as reflected by weighted misclassification error). So AdaBoost can obtain a better decrease in the objective value. This conjecture is also supported by the fact that in Figure 3's run of experiment 3.6, decreasing the hypothesis space by restricting the terminal node size, also reduced the convergence rate.

- *For a fixed separable problem, faster convergence rates of AdaBoost can result in larger margins.* AdaBoost is known to approximately optimize the margin as measured by the 1-norm (Rosset et al., 2004; Schapire et al., 1998). The objective decreases the numerator of the margin and the iterations increase the denominator, so getting a smaller objective quicker creates a better margin. Bigger hypothesis spaces allow bigger steps resulting in larger margins. This finding is also supported by the fact that when shrinkage is used to change the convergence rate, the resulting margins changed as well (see Figure 4). For problems with no training error, we expect larger margins to translate to better generalization rates. But MW's experiments 3.2 and 4.2 show that this is not always the case. Figure 2 shows the margin for 1 run of Experiment 3.2. So MW are quite right in their conclusion that there is more to the



(a) Objective/Gradient (log10)

(b) Test Error



(c) Margin

Figure 4: 1 trial of Experiment 3.7 (10% Bayes Error) for AdaBoost + 8-node trees (red, squares), AdaBoost + 8-node trees with .5 shrinkage (blue, triangles) and AdaBoost + 8-node trees with .1 shrinkage (purple, dots).

generalization of AdaBoost then just optimizing the loss. Adding consideration of the margin is not enough either.

- *For slowly converging problems, AdaBoost will frequently be regularized by early stopping.* In experiments 3.1 and 3.2, AdaBoost with stumps is overfitting and the early stopping in MW at 1000 iterations helps the generalization error. For this specific experiment, the slow convergence is a result of cycling. For the first trial in experiments 3.1 and 3.2, AdaBoost with stumps only generated 158 and 156 distinct weak learners in 1000 iterations respectively. The weak learners generated by AdaBoost with 8-node trees and 16-node trees were distinct except for 2. By cycling through relatively few weak learners, AdaBoost with stumps strongly weights a few trees. This appears to be bad for generalization in experiment 3.1 and good for generalization for the no noise case in experiment 3.2.

- *For more rapidly converging problems, AdaBoost will converge and enter an overtraining phase.* For the larger tree cases, the objective and margins converge rapidly. Typically one would halt an optimization algorithm when the gradient became near 0. In the MW experiments, AdaBoost with 8-node trees is overtrained past the point where one would normally

halt an optimization algorithm based on gradient criteria (Gill et al.). AdaBoost doesn't overfit in this overtraining phase because it has converged and only very small changes are being made. Perhaps the overtraining phase contributes to the robustness of AdaBoost, since AdaBoost is performing the self-averaging discussed in MW and acting more like bagging. In the MW experiments, AdaBoost achieves better generalization when trained to an extraordinarily high degree of accuracy, a fact contrary to the usual loose convergence criteria used in support vector machines (Bennett and Parrado-Hernández, 2006). But care must be taken to halt the boosting algorithm before the overtraining produces numeric problems due to finite precision problems. As shown in Figure 1, AdaBoost with 16-node trees underflows at 1017 iterations for the 10% Bayes error case and at 673 iterations for the 0 error case. AdaBoost with 8-node trees also underflows eventually as well.

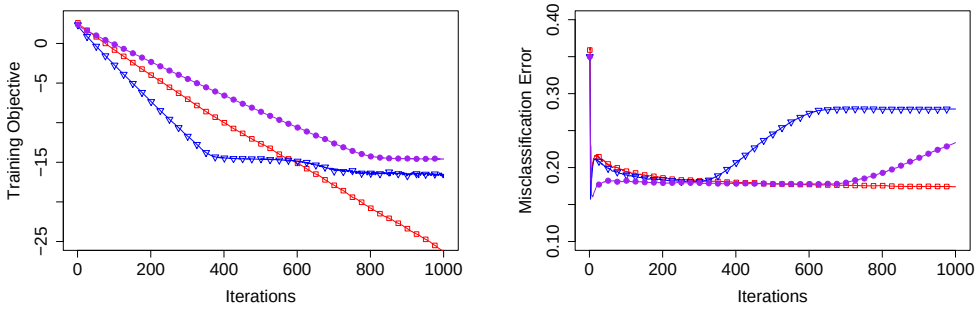
4. LogitBoost versus AdaBoost

Experiments 3.4 and 4.4 compare LogitBoost and AdaBoost and conclude LogitBoost overfits. Tracking the convergence of LogitBoost shows that this is not quite the case. We show our results repeating experiment 4.4 exactly as in the paper for AdaBoost and LogitBoost. Recall LogitBoost differs from AdaBoost in two ways. First, it uses the logistic loss instead of the exponential loss and second, it uses a Newton step instead of an exact step size. The Newton step for logistic loss works out to be 1/2 at each iteration. AdaBoost's stepsize is adaptive. The CD convergence results do not apply directly to LogitBoost as implemented in the paper because of the Newton step.

Figure 4 shows the average objective and misclassification results for 100 trials with 8-node trees. Note that at about 375 iterations, LogitBoost fails to obtain a decrease in the objective because the Newton step is too large when the objective is very small. From that point, the testing error declines. LogitBoost with shrinkage converges more slowly, so it can go more iterations before the step size fails. Once the objective becomes too small, the stepsize fails and the generalization performance of LogitBoost decreases remarkably. The LogitBoost objective is still small and continues to decrease slightly, but the self-averaging properties observed in AdaBoost in the overtraining phase are lost. Note that up until it missteps, LogitBoost is very competitive with AdaBoost. If LogitBoost and AdaBoost were halted at the same high degree of accuracy (e.g., 10^{-8}), there is no evidence of overfitting.

5. Conclusion

MW are correct is saying that optimization provides only part of the picture because optimization tells us nothing about generalization. Mathematical programming theory tells us that more well-posed boosting problems with well-conditioned loss functions (like the hinge loss) and explicit regularization in the objective should produce boosting algorithms with better behavior from an optimization perspective. But AdaBoost's ill-conditioning appears to be one of the secrets of its success. More investigation is needed comparing Adaboost with its regularized counterparts. Certainly machine learning researchers should mind their optimization theory and track the convergence of their algorithms. Optimality conditions should be used to halt and compare boosting algorithms instead of fixed iteration limits.



(a) Objective (log10) (b) Test Error

Figure 5: Average of 100 trials of Experiment 4.3 (10% Bayes Error) for AdaBoost (red, squares), LogitBoost (blue, triangles), and LogitBoost with .5 shrinkage (purple, circles) for 8-node trees.

References

K.P. Bennett and E. Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 65–72, 2000.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, 148:156, 1996.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting (With discussion and a rejoinder by the authors). *Ann. Statist*, 28(2):337–407, 2000.

P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London and New York.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. *Advances in Neural Information Processing Systems*, 12:512–518, 2000.

G. Rätsch. *Robust Boosting via Convex Optimization Theory and Applications*. PhD thesis, Universitat Potsdam, 2001.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

C. Rudin, I. Daubechies, and R.E. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

Journal of Machine Learning Research 9 (2008) 165-170 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Andreas Buja BUJA.AT.WHARTON@GMAIL.COM
Statistics Department
The Wharton School, University of Pennsylvania
Philadelphia, PA 19104-6340

Werner Stuetzle WXS@STAT.WASHINGTON.EDU
Statistics Department
University of Washington
Seattle, WA 98195-4322

Editor: Yoav Freund

We thank the authors for writing a thought-provoking piece that may ruffle the feathers of recent orthodoxies in boosting. We also thank JMLR for publishing this article! Since the late 1990s, boosting has undergone the equivalent of a simultaneous X-ray, fMRI and PET exam, and the common view these days is that boosting is a kind of model fitting. As such, it is subjected to assumptions that are common in non-parametric statistics, such as: limiting the complexity of the base learner, building up complexity gradually by optimization, and preventing overfitting by early stopping or by regularizing the criterion with a complexity penalty. The theories backing this up use VC dimensions and other measures to show that, if the complexity of fits grows sufficiently slowly, asymptotic guarantees can be given. Into this orthodox scene Mease and Wyner throw one of the most original mind bogglers we have seen in a long time: "if stumps are causing overfitting, be willing to try larger trees." In other words, if boosting a low-complexity base learner leads to overfit, try a higher-complexity base learner; boosting it might just not overfit. Empirical evidence backs up the claim.

Is this counterintuitive wisdom so surprising? Yes, if seen from the point of view of orthodoxy, but less so when reviving some older memories. We may remind ourselves how boosting's fame arose in statistics when the late Leo Breiman stated in a discussed 1998 Annals of Statistics article (based on a 1996 report) that boosting algorithms are "the most accurate ... off-the-shelf classifiers on a wide variety of data sets." We should further remind ourselves what this praise was based on: boosting of the full CART algorithm by Breiman himself, and boosting of the full C4.5 algorithm by others. In other words, the base learners were anything but 'weak' in the sense of today's orthodoxy, where 'weak' means 'low complexity, low variance, and generally high bias.' (Few people today use PAC theory's untenable notion of weak learner, which was gently demolished by Breiman in the appendix of this same article.) Breiman's (1998b, p. 802) 02) major conclusion at the time was: "The main effect of both bagging and [boosting] is to reduce variance." It appears, therefore, that his notion of 'weak learner' was one of 'high complexity, high variance, and low bias'! This was before the low-variance orthodoxy set in and erased the memories of the early boosting experiences.

Unfortunately, soon thereafter Breiman saw his own assumptions thrown into question when he learned from Schapire et al.'s (1998) work that excellent results could also be achieved by boosting

stumps. This experience was later reinforced when Friedman et al. (2000) introduced the interpretation of boosting as model fitting: the base learner now had to be weak in the sense of low variance. Ever since, theoretical attempts at 'explaining boosting' have relied on low complexity of the base learner and controlling complexity of the final classifier to assure good generalization properties. These 'explanations,' however, have never been able to explain why boosting is relatively immune to overfitting, even when not stopped and not regularized and used with a high complexity base learner.

Mease and Wyner's achievement is to pull the messy truth out from under the rug of the low-variance orthodoxy. They do so with the equivalent of boy scout tools, some simple but telling simulations, which reinforce the idea that our reasonings about early stopping, regularization, low variance of the base learner, and the specifics of the surrogate loss function, are not or not the only essence of boosting. To explain why this is so, Mease and Wyner do not give us hard theory, but they point in a direction, essentially by recovering memories that predate the low-variance orthodoxy: "self-averaging" for variance reduction, which is the principle behind bagging and random forests.

While variance reduction is an aspect that has been ignored by the low-variance orthodoxy, the orthodoxy's implicit dogma, that boosting can reduce bias, is also true. As asserted and documented empirically a decade ago by Schapire et al. (1998, Section 5.3), boosting can do both. Depending on the data and the base learner, the effect that dominates may be bias reduction *or* variance reduction. In this regard Schapire et al.'s (1998) simulation results as summarized in their Table 1 (p. 1673) are illuminating, and had we taken them seriously sooner, we would be less surprised by Mease and Wyner's messages. Arguing against Breiman (1998b), Schapire et al. used the table to make the now orthodox point that boosting can reduce bias. An unprejudiced look shows, however, that the winner in all four scenarios is boosting C4.5, not boosting stumps, and when C4.5 is the base learner the overwhelming story is indeed variance reduction. With this information, the Mease-Wyner mind boggler is a touch less mind boggling indeed: From the combined evidence of Breiman (1998b) and Schapire et al. (1998), we should expect that boosting high-variance base learners generally outperforms boosting low-variance base learners. For the practitioner the recommendation should be to boost CART or C4.5. In theoretical terms, one should let most of the bias removal be done by the base learner and take advantage of boosting's variance removal; at the same time, boosting may further reduce the base learner's bias by another notch if that is possible.

Where does this leave us in terms of theory? The implications of Mease and Wyner's unorthodoxies stand: Complexity controlling theories of bias removal are off the mark; they are not incorrect but misleading, and they ignore a whole other dimension that matters hugely for the practice of boosting. What we need is a theory that explains bias and variance reduction in a single framework. We do not even know of a unified general theory of variance reduction, although some interesting work has been done in the area of bagging (Bühlmann and Yu, 2002) and random forests (Amit et al., 2001). The real jackpot, however, would be a theory that explains how and when boosting reduces bias *and* variance.

Meanwhile we are left with some tantalizing clues, above all Breiman's hunch (1999, p. 3): "AdaBoost has no random elements .... But just as a deterministic random number generator can give a good imitation of randomness, my belief is that in its later stages AdaBoost is emulating a random forest." If born out, this conjecture would have theoretical and practical implications. For one, it would mean that the initial stages of boosting may remove bias, whereas the later stages remove variance. According to Breiman (1998b, p. 803), boosting a high-variance base learner does not yield convergence but exhibits "back and forth rocking" of the weights, and "This variability

may be an essential ingredient of successful [boosting] algorithms." Breiman implies that at some point boosting iterations turn into a pseudo-random process whose behavior may resemble more the purely random iterations of bagging than those of a minimization process. This random process may be able to achieve the self-averaging effect of variance reduction that is so prominent when boosting high-variance base learners. If this view is correct, one may have to rethink the role of the surrogate loss function that is minimized by boosting. Its main role is to produce structured weights in the iterations, but with noisy errors, these weights may for practical purposes be as much random as they are systematic. This insight jibes with Wyner's (2002) malicious experiments in which he doubled the step size of discrete AdaBoost with C4.5, thereby assuring that the exponential loss never decreased and in fact provably remained at a constant level; his empirical results indicated that on average this SOR (successively over-relaxed) form of boosting performs as well as regular AdaBoost. These results may be taken as evidence that the minimization aspect is of little importance for a high-variance base learner; of greater importance may be a pseudo-random aspect of the reweighting scheme that achieves variance reduction similar to bagging, just more successfully due to a sort of adaptivity in the reweighting that improves over the purely random resampling of bagging.

If the pseudo-random aspect of boosting is critical for high-variance base learners, one may draw consequences and implement boosting with proper pseudo-random processes. So did Breiman. He didn't attempt a theory of boosting for high-variance base learners, and instead he put his intuitions to use in further proposals such as in his work on "half & half bagging" (Breiman, 1998a), apparently with success. Another example that benefited from Breiman's inspiration was Friedman's (2002) "stochastic gradient boosting" which inhibits convergence of boosting by computing gradient steps from random subsamples drawn without replacement. Friedman (ibid., p. 9) observes improvements over deterministic boosting in a majority of situations, above all for small samples and "high capacity" (high variance) base learners. He admits that "the reason why this randomization produces improvement is not clear," but suggests "that variance reduction is an important ingredient." Friedman goes on to suggest that stochastic AdaBoost with sampling from the weights rather than reweighting may have similar variance-reducing effects. In early boosting approaches such sampling (with replacement) was performed to match the given sample size, but Friedman suggests that further variance reduction could be gained by choosing smaller resamples.

An implication of Breiman's hunch is that the real difference between LogitBoost and AdaBoost is not so much due to the differences in loss functions as to the minimization method, at least when the base learner has relatively high variance, or generally in the late stages of boosting. AdaBoost can be interpreted as constrained gradient descent on the exponential loss, whereas LogitBoost is Newton descent on the logistic loss (Friedman et al., 2000). The two minimization schemes produce very different reweighting schemes, and they work off different working responses during the iterations. We are currently ignorant about whether LogitBoost develops pseudo-random behavior late in the iterations, similar to AdaBoost. If it does, the cause may be traced to the base learner, and the phenomenon may be robust to the specifics not only of loss functions but of algorithms as well.

Another implication of Breiman's hunch is that boosting does both, reduce bias and variance, in the same problem, but each primarily at different stages of the boosting iterations. If it is true that variance reduction occurs during later iterations, then this should go a long way to explain boosting's relative immunity to overfitting. By comparison, conventional fitting mechanisms only know how to do one thing: follow the data ever more closely, thereby continually reduce bias and continually

accumulate variance. According to orthodoxy, therefore, the art is to find the proper balance, and to this end auxiliary devices such as early stopping, regularization penalties and cross-validation come into play. Boosting seems to be different, but we do not have the theory yet to prove it.

All that we said so far is based on out-of-sample classification error. A peculiarity of classification error is that it is not the criterion being minimized in-sample because of its discontinuous nature. The role of minimizing a smooth surrogate loss function is to trace a path that leads to low classification error, but the surrogate loss is not of interest in itself. Yet, for the variance reducing properties of the resulting classifier, the surrogate loss is of interest. First of all, the surrogate loss should keep decreasing because for example discrete AdaBoost is constrained gradient descent with line search (Friedman et al., 2000). This explains why in terms of the surrogate loss, the fitted class probability estimates end up vastly overfitting the data, confirming the orthodox view in terms of the surrogate loss. Yet, two phenomena are also observed: in terms of out-of-sample classification error, no overfitting is taking place, and, according to Breiman, no convergence of the weights is taking place. The bouncing of the weights would indicate that, in spite of a well-behaved convex loss function, the descent directions chosen by the base learner become erratic. Such behavior would be plausible if the base learner is of the high-variance type, but the specifics of why the variance component of out-of-sample classification error is improved is not explained. It is quite clear, though, that explaining boosting's variance reduction would be a greater achievement than explaining its bias reduction. Bias can be largely taken care of by the base learner, variance can't.

If the peculiarities we observe in boosting are due to the use of two loss functions, one may ask whether any lessons learned carry over to other parts of statistics. The "statistical view" has indeed produced generalizations of boosting to other areas, such as regression: Bühlmann and Yu's L_2-boosting (2003), Friedman's gradient boosting in their deterministic and stochastic forms (2001; 2002), and boosting of exponential and survival models by Ridgeway (1999). In these single-loss function contexts, the paradoxical phenomena should no longer be visible, as they aren't for boosting if judged in terms of the surrogate loss function. Yet, Friedman's stochastic gradient boosting shows that adding an element of variance reduction with randomization may just be what the doctor ordered in most statistical model fitting contexts even with a single loss function. We should therefore aim for a variance reduction theory for all of statistics, reaching beyond classification.

Another question that may be raised for binary, or categorical response data in general, is whether classification error is as desirable a loss function as suggested by the attention it has received. Classification error is a bottom line number that may be appropriate in industrial contexts where real large scale engineering problems are solved, for example, in document retrieval. One might characterize these contexts as "the machine learner's black box problems." There do exist other contexts, though, and one might characterize them as "the problems of the interpreting statistician." When interpretation is the problem, attaining the last percent of classification accuracy is not the goal. Instead, one hopes to develop a functional form that reasonably fits the data but also "speaks," that is, lends itself to statements about what variables are associated with the categorical response. Fitting good conditional class probabilities takes on greater importance because associations and effects can then be measured in terms of differences in the logits of class 1 (for example) for a unit or percentage difference in the predictor variables. Interpretability is a problem for nonparametric model fits such as boosted trees. The decomposition of complex fits into interpretable components, for example with an ANOVA decomposition as suggested by Friedman et al. (2000), takes on considerable importance. In the end, one may want to produce a few telling plots explaining functional form and a few numbers summarizing the strengths of various associations. When

fitting models for conditional class probabilities, the surrogate loss becomes the primary loss function because it can be interpreted as a loss function for fitted class probabilities. It is one of the achievements of Friedman et al. (2000) to have shown that this is true for exponential loss as much as for logistic loss, even though there is a misperception, as pointed out by Mease and Wyner, that LogitBoost was specifically designed to recover class probabilities that AdaBoost couldn't. Exponential loss does similar things as logistic loss in Friedman et al.'s analysis, and they provide the appropriate link functions for both. All this is relevant only if minimization of the "surrogate/now-primary loss" is prevented from overfitting, with cross-validated early stopping, penalization, or variance-reducing randomization, and it comes at the cost of diminished classification performance, one of Mease and Wyner's points.

Diminished classification performance when estimating class probabilities is easily explained; it is due to a compromise that class probability estimation has to strike. It effectively attempts good classification simultaneously at *all* misclassification cost ratios. (Note that this ratio is assumed to be one in most of the boosting literature.) This statement can be made precise in a technical sense: unbiased loss functions for class probabilities, so-called "proper scoring rules," are weighted mixtures of cost-weighted misclassification losses Buja et al. (2005). After mapping exponential and logistic losses to probability scales with their associated inverse link functions, they turn into proper scoring rules and therefore exhibit the mixture structure just described. It follows that both loss functions attempt compromises across classification problems with non-equal misclassification costs. Both loss functions give inordinate attention to extreme cost ratios, but exponential loss even more so than logistic loss. At any rate, the nature of the compromise is such that no cost ratio, in particular not equal costs, is served optimally if the exponential or logistic losses are tuned to high out-of-sample performance. By comparison, overfitting these losses in-sample seems to provide benefits in terms of classification error. However, once we change our priorities from black-box performance to interpretation, and hence from classification to class probability estimation, we may prefer tuning surrogate loss and accept the increased classification error.

Anticipating an objection by Mease and Wyner, we should disclose that one of us (Buja) collaborated with them on an article that is relevant here (Mease et al., 2007). In this work we traveled the opposite of the usual direction by composing class probability estimates from layered classification regions, estimated at a grid of misclassification cost ratios. Presumably such class probability estimation inherits superior performance from boosting in classification. When interpretation is the goal, however, a simple functional form that "speaks" might be more desirable than the increased performance of the layered estimates we provide in our joint proposal. The problem is that our proposal inherits the interpretative disadvantages of boosted classification regions, which tend to be jagged around the edges and pockmarked with holes—not a credible feature when it comes to interpretation.

We started this discussion joining Mease and Wyner in their argument against today's boosting orthodoxy. We ended by questioning the single-minded reliance on classification error as the only yard stick of performance. Still, Mease and Wyner's call should be heard because the orthodoxy misattributes the causes of boosting's success and makes invalid recommendations.

References

Y. Amit, G. Blanchard, and K. Wilder. Multiple randomized classifiers: Mrcl. Technical report, University of Chicago, 2001.

L. Breiman. Half & half bagging and hard boundary points. Technical Report 534, Statistics Dept., Univ. of California, Berkeley, 1998a.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–849, 1998b.

L. Breiman. Random forests—random features. Technical Report 567, Statistics Dept., Univ. of California, Berkeley, 1999.

P. Bühlmann and B. Yu. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

Peter Bühlmann and Bin Yu. Boosting with the L_2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, The Wharton School, University of Pennsylvania, 2005.

J. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

J. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378, 2002.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

D. Mease, A. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8:409–439, 2007.

G. Ridgeway. The state of boosting. *Computing Science and Statistics*, 31:172–181, 1999.

R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5):1651–1686, 1998.

A. Wyner. Boosting and the exponential loss. In *Proceedings of the Ninth Annual Conference on AI and Statistics*, 2002.

Journal of Machine Learning Research 9 (2008) 171-174 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Yoav Freund YFREUND@UCSD.EDU
Department of Computer Science and Engineering
University of California
San Diego, CA 92093

Robert E. Schapire SCHAPIRE@PRINCETON.EDU
Princeton University
Department of Computer Science
35 Olden Street
Princeton, NJ 08540

Editor: Yoav Freund

For such a simple algorithm, it is fascinating and remarkable what a rich diversity of interpretations, views, perspectives and explanations have emerged of AdaBoost. Originally, AdaBoost was proposed as a "boosting" algorithm in the technical sense of the word: given access to "weak" classifiers, just slightly better in performance than random guessing, and given sufficient data, a true boosting algorithm can provably produce a combined classifier with nearly perfect accuracy (Freund and Schapire, 1997). AdaBoost has this property, but it also has been shown to be deeply connected with a surprising range of other topics, such as game theory, on-line learning, linear programming, logistic regression and maximum entropy (Breiman, 1999; Collins et al., 2002; Demiriz et al., 2002; Freund and Schapire, 1996, 1997; Kivinen and Warmuth, 1999; Lebanon and Lafferty, 2002). As we discuss further below, AdaBoost can been seen as a method for maximizing the "margins" or confidences of the predictions made by its generated classifier (Schapire et al., 1998). The current paper by Mease and Wyner, of course, focuses on another perspective, the so-called statistical view of boosting. This interpretation, particularly as expounded by Friedman et al. (2000), focuses on the algorithm as a stagewise procedure for minimizing the exponential loss function, which is related to the loss minimized in logistic regression, and whose minimization can be viewed, in a certain sense, as providing estimates of the conditional probability of the label.

Taken together, these myriad interpretations of AdaBoost form a robust theory of the algorithm that provides understanding from an extraordinary range of points of view in which each perspective tells us something unique about the algorithm. The statistical view, for instance, has been of tremendous value, allowing for the practical conversion of AdaBoost's predictions into conditional probabilities, as well as the algorithm's generalization and extension to many other loss functions and learning problems.

Still, each perspective has its weaknesses, which are important to identify to keep our theory in touch with reality. The current paper is superb in exposing empirical phenomena that are apparently difficult to understand according to the statistical view. From a theoretical perspective, the statistical interpretation has other weaknesses. As discussed by Mease and Wyner, this interpretation does not explain AdaBoost's observed tendency not to overfit, particularly in the absence of regularization

or early stopping. It also says little about how well AdaBoost will generalize when provided with a finite data set, nor how its ability to generalize is dependent on the complexity or simplicity of the base classifiers, an issue that arises in the experiments comparing decision stumps and decision trees in this role.

Much of the difficulty arises from the fact that AdaBoost is a *classification* algorithm (at least as it is used and studied in the current paper). This means that AdaBoost's purpose is to find a rule h that, given X, predicts one of the labels $h(X)$, and that attempts to achieve minimal probability of an incorrect classification (in which $h(X)$ disagrees with the true label Y). This is quite different from the problem of estimating the conditional probability $P(Y|X)$. An accurate estimate of this conditional probability is a sufficient, but certainly not a necessary, condition for minimizing the classification error. A weaker requirement that is still sufficient is to estimate the set of inputs for which $P(Y = +1|X) > 1/2$. In most cases, this requirement is much weaker than the requirement of getting good estimates of conditional probabilities. For example, if $P(Y = +1|X) = 0.49$ then our estimate of the conditional probability need be accurate to within 1%, while if $P(Y = +1|X) = 0.2$ the accuracy we need is only 30%.

This simple observation demonstrates a crucial shortcoming in the statistical interpretation of Adaboost, and undermines many of its apparent consequences, including the following:

- *Adaboost can be interpreted as a method for maximizing conditional likelihood.* If the goal is not to estimate the conditional probability, there is no reason to maximize likelihood.

- *A question of central importance is whether Adaboost is asymptotically consistent.* When evaluating probability estimators, it is standard procedure to start by verifying that the estimator is unbiased. Once the estimator is confirmed to be unbiased, the next question is the rate at which its variance decreases with the size of the sample. Again, as the learning problem in the case of classification is a weaker one, it is not clear that this is the relevant sequence of questions that a theoretician should ask.

- *Decision stumps should be used as base classifiers when the input variables are independent* This argument is based on the assumption that the goal is to estimate probabilities.

The view of AdaBoost as a method for minimizing exponential loss, though in some ways quite useful, can also lead us very much astray, as pointed out to some degree by Mease and Wyner. Taken to an extreme, this view suggests that any method for minimizing exponential loss will be equally effective, and is likely to be much better if designed with speed and this explicit goal in mind. However, this is quite false. Indeed, any real-valued classifier F which classifies the training examples perfectly, so that $y_i F(x_i) > 0$ for each training example (x_i, y_i), can be modified to minimize the exponential loss $\sum_i e^{-y_i F(x_i)}$ simply by multiplying F by an arbitrarily large positive constant. This scaling of F of course has no impact on the classifications that it makes. Thus, in the common case in which an exponential loss of zero is possible, minimization of this loss means nothing more than that the computed classifier F has a classification error of zero on the training set. The minimization of this particular loss tells us nothing more, and leaves us as open to overfitting as any other method whose only purpose is minimization of the training error.

This means that, in order to understand AdaBoost, which does indeed minimize exponential loss, we need to go well beyond this narrow view. In particular, we need to consider the *dynamics* of AdaBoost—not just *what* it is minimizing, but *how* it goes about doing it.

Like other interpretations of AdaBoost, although the statistical view has its weaknesses, it also has its strengths, as noted above. Still, to fully understand AdaBoost, particularly in the face of such deficiencies, it seems unavoidable that we consider a range of explanations and modes of understanding. Where the statistical view may be lacking, the margins explanation in particular can often shed considerable light.

Briefly, the *margin* of a labeled example with respect to a classifier is a real number that intuitively measures the confidence of the classifier in its prediction on that example. More precisely, in the notation of Mease and Wyner, the margin on labeled example (x,y) is defined to be $yF_M(x)/\sum_m \alpha_m$. Equivalently, viewing the prediction of AdaBoost's combined classifier as a weighted majority vote of the base classifiers, the margin is the weighted fraction of base classifiers voting for the correct label minus the weighted fraction voting for the incorrect label.

The margins theory (Schapire et al., 1998) provides a complete analysis of AdaBoost in two parts: First, AdaBoost's generalization error can be bounded in terms of the distribution of margins of training examples, as well as the number of training examples and the complexity of the base classifiers. And second, it can be proved that AdaBoost's dynamics have a strong tendency to increase the margins of the training examples in a manner that depends on the accuracy of the base classifiers on the distributions on which they are trained.

This theory is quite useful for understanding AdaBoost in many ways (despite a few shortcomings of its own—see, for instance, Breiman (1999) as well as the recent work of Reyzin and Schapire (2006)). For starters, the theory, in which performance depends on margins rather than the number of rounds of boosting, predicts the same lack of overfitting commonly observed in practice. The theory provides non-asymptotic bounds which, although usually too loose for practical purposes, nevertheless illuminate qualitatively how the generalization error depends on the number of training examples, the margins, and the accuracy and complexity of the base classifiers. Finally, the theory is concerned directly with classification accuracy, rather than the algorithm's ability to estimate conditional probabilities, which is in fact entirely irrelevant to the theory.

Moreover, some of the phenomena observed by Mease and Wyner do not appear so mysterious when viewed in terms of the margins theory. For instance, the experiments in Section 3.1 show AdaBoost overfitting with stumps but not decision trees. In terms of margins, decision trees have higher complexity, which tends to hurt generalization, but also tend to produce much larger margins, which tend to improve generalization, an effect that can easily be strong enough to compensate for the increased complexity. Moreover, according to the theory, these larger margins tend to provide immunity against overfitting, and indeed, overfitting is expected exactly in the case that we are using base classifiers producing small margins, such as decision stumps. This is just what is observed in Figure 1.

In sum, the various theories of boosting, including the margins theory and the statistical view, are all imperfect but are largely complementary, each with its strengths and weaknesses, and each providing another piece of the AdaBoost puzzle. It is when they are taken together that we have the most complete picture of the algorithm, and the best chances of understanding, generalizing and improving it.

References

Leo Breiman. Prediction games and arcing classifiers. *Neural Computation*, 11(7):1493–1517, 1999.

Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1/2/3), 2002.

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1/2/3):225–254, 2002.

Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332, 1996.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, April 2000.

Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 134–144, 1999.

Guy Lebanon and John Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*, 2002.

Lev Reyzin and Robert E. Schapire. How boosting the margin can also boost classifier complexity. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

Journal of Machine Learning Research 9 (2008) 175-180 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Jerome Friedman JHF@STANFORD.EDU
Trevor Hastie HASTIE@STANFORD.EDU
Robert Tibshirani TIBS@STANFORD.EDU
Department of Statistics
Stanford University
Stanford, CA 94305

Editor: Yoav Freund

1. Introduction

This is an interesting and thought-provoking paper. We especially appreciate the fact that the authors have supplied R code for their examples, as this allows the reader to understand and assess their ideas. The paper inspired us to re-visit many of these issues underlying boosting methods. However in the end we do not believe that the examples provided in the paper contradict our statistical view, although other views may well prove informative.

2. Our Statistical View of Boosting

Friedman et al. (2000) and our book (Hastie et al., 2001) argue that boosting methods have three important properties that contribute to their success:

1. they fit an additive model in a flexible set of basis functions
2. they use a suitable loss function for the fitting process
3. they regularize by forward stagewise fitting; with shrinkage this mimics an L_1 (lasso) penalty on the weights.

In many cases the paper ascribes consequences of this statistical view that are not the case. For example, it does not follow that smaller trees are necessarily better than larger ones for noisier problems (Sections 3.2 and 4.2), that the basis should necessarily be restricted as described in Sections 3.6 and 4.6, or that regularization should be based on the loss function used for fitting (Sections 3.5 and 4.5). To the extent possible model selection should be based on the ultimate loss associated with the application. Also, there is no requirement that test error have a unique minimum as a function of the number of included terms (Sections 3.4 and 4.4). However, to the extent that these are commonly held beliefs, the paper provides a valuable service by pointing out that they need not hold in all applications.

There is no direct relation between the application of shrinkage and overfitting (Sections 3.7 and 4.7). Heavy shrinkage emulates L_1 regularization, whereas its absence corresponds to stagewise

fitting approximating L_0 regularization. There is nothing in the statistical view that requires L_1 to be superior to L_0 in every application, although this is often the case. The best regularizer depends on the problem: namely the nature of the true target function, the particular basis used, signal-to-noise ratio, and sample size.

Finally, there is nothing in our statistical interpretation suggesting that boosting is similar to one nearest neighbor classification (Sections 3.9 and 4.9).

None-the-less, the paper does provide some interesting examples that appear to contradict the statistical interpretation. However these examples may have been carefully chosen, and the effects seems to vanish under various perturbations of the problem.

3. Can the "Wrong" Basis Work Better than the Right One?

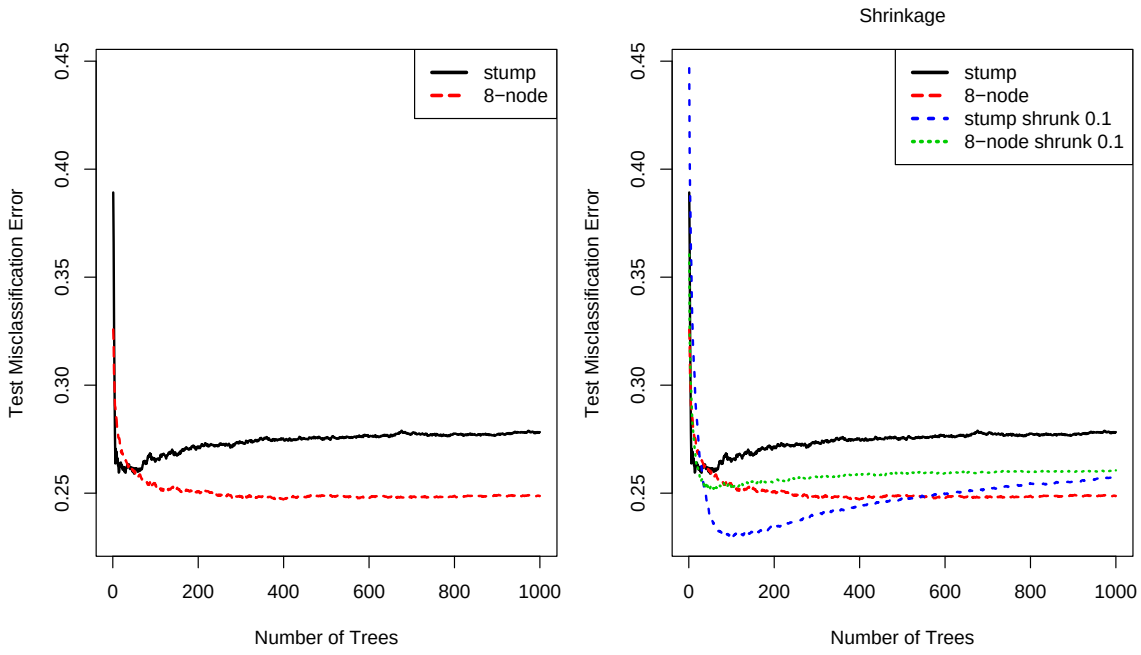


Figure 1: Average test misclassification error for 20 replications of Mease and Wyner's example used in their Figure 1. We used the package GBM in R, with the "adaboost" option. The left panel shows that 8-node trees outperform stumps. The right panel shows that stumps with shrinkage win handily.

The left panel of Figure 1 shows a version of the paper's Figure 1. We see that boosting with 8 node trees seems to outperform stumps, despite the fact that the generative model is additive in the predictor variables. However the right panel shows what happens to both stumps and 8 nodes trees when shrinkage is applied. Here shrinkage helps in both cases, and we see that stumps with shrinkage work the best of all.

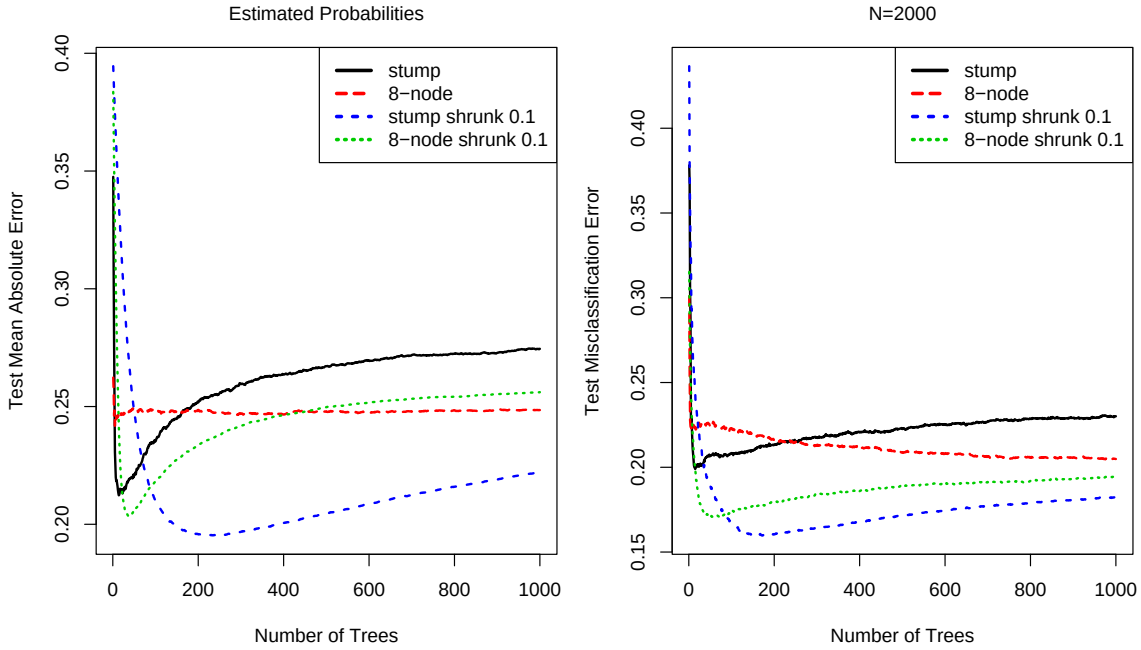


Figure 2: Left panel: average absolute deviations of the fitted probabilities from the true probabilities for the same simulations as in Figure 1. Right panel: average test misclassification error for the same simulations as in Figure 1, except using 2000 rather than 200 training examples.

We are not sure why unshrunken 8 node trees outperform unshrunken stumps in this example. As in the paper, we speculate that the extra splits in the 8 node tree might act as a type of regularizer, and hence they help avoid the overfitting displayed by unshrunken stumps in this example. All but the first split will tend to be noisy attempts at the other variables, which when averaged will have a "bagging" effect.

However this explanation becomes less convincing and indeed the effect itself seems to fade when we look more deeply. Figure 2 [left panel] shows the average absolute error in the estimated probabilities, while Figure 2[right panel] shows what happens when we increase the sample size to 2000. In Figure 3[left panel] we use the Bernoulli loss rather than exponential of Adaboost, and Figure 3[right panel] shows results for the regression version of this problem. In every case, the effect noted by the authors goes away and both the correct bases and shrinkage help performance. We repeated these runs on the second simulation example of Section 4, and the results were similar. Thus the effect illustrated by the authors is hard to explain, and seems to hold only for misclassification error. It depends on a very carefully chosen set of circumstances. Most importantly, we have to remember the big picture. Looking at the right panel of Figure 1, which method would anyone choose? Clearly, shrunken stumps work best here, just as might be expected from the statistical view.

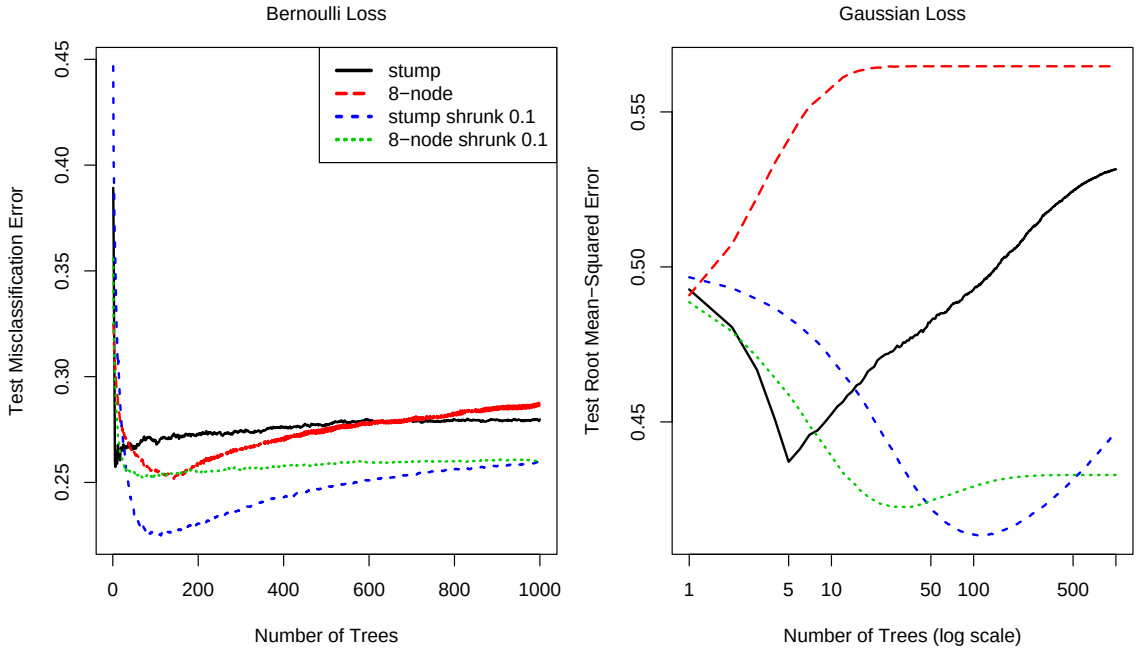


Figure 3: Left panel: test misclassification error when boosting with Bernoulli loss for the same simulations as in Figure 1. Right panel: root mean-squared test error when boosting with squared-error loss for the same simulations as in Figure 1 (legend as in left panel).

Figure 4 shows the fitted probabilities over the 20 runs, separately for each class, when using 250 shrunken stumps. Here 250 was chosen since it corresponds to the minimum in Figure 2[left panel]. This is an appropriate tradeoff curve if we are interested in probabilities; test deviance would also be fine. We see that the estimates are biased toward 0.5, which is expected when regularization is used. Hence they are underfit, rather than overfit.

A similar argument can be made concerning the paper's Figure 3. Yes, AdaBoost works better than Logitboost in this example. But using the statistical view of boosting, we have moved on and developed better methods like gradient boosting (Friedman, 2001) that typically outperform both of these methods.

Hastie et al. (2007) add further support to (3) of the statistical interpretation of boosting: they show that the incremental forward stagewise procedure used in boosting (with shrinkage) optimizes a criterion similar to but smoother than the L_1 penalized loss.

4. Conclusion

No theory, at least initially, can fully explain every observed phenomenon. Everything about regularized regression is not yet fully understood. There is still considerable ongoing research in the literature concerning the interplay between the target function, basis used, and regularization method. Hopefully, some of the apparent anomalies illustrated in this paper will eventually be explained with

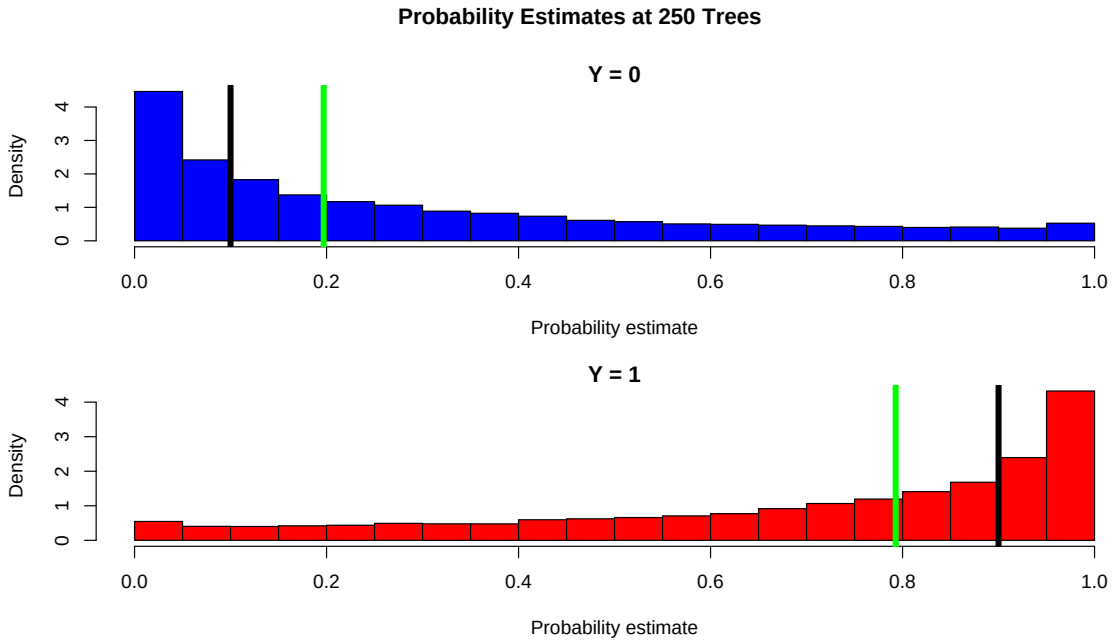


Figure 4: Fitted probabilities shown for the two classes, at 250 shrunken stumps. The vertical black bars are the target probabilities for this problem, and the green bars are the median of the estimates in each class.

a more thorough understanding of these issues. The paper provides a service in reminding us that there is still work remaining.

Although we would not begin to suggest that our statistical view of boosting has anywhere near the substance or importance of the Darwin's theory of evolution, the latter provides a useful analogy. The proponents of Intelligent Design point out that the theory of evolution does not seem to explain certain observed biological phenomena. And therefore they argue that evolution must be wrong despite the fact that it *does* explain an overwhelming majority of observed phenomena, and without offering an alternative *testable* theory.

We are sure that the authors will mount counter-arguments to our remarks, and due to the (standard format) of this discussion, they will have the last word. We look forward to *constructive* counter-arguments and alternative explanations for the success of boosting methods that can be used to extend their application and produce methods that perform better in practice (as in the right panel of Figure 1).

References

J. Friedman. Greedy function approximation: The gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting (with discussion). *Annals of Statistics*, 28:337–407, 2000.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning; Data Mining, Inference and Prediction*. Springer Verlag, New York, 2001.

T. Hastie, J. Taylor, R. Tibshirani, and G. Walther. Forward stagewise regression and the monotone lasso. *Electronic Journal of Statistics*, 1:1–29 (electronic), 2007. DOI: 10.1214/07-EJS004.

Journal of Machine Learning Research 9 (2008) 181-186 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008**: And Yet It Overfits**

Peter J. Bickel BICKEL@STAT.BERKELEY.EDU
Department of Statistics
University of California
Berkeley, CA 94720-3860, USA

Ya'acov Ritov YAACOV.RITOV@GMAIL.COM
Department of Statistics
The Hebrew University of Jerusalem
91905 Jerusalem, Israel

Editor: Yoav Freund

Galileo: God help us, I'm not half as sharp as those gentlemen in the philosophy department. I'm stupid, I understand absolutely nothing, so I'm compelled to fill the gaps in my knowledge... Sir, my branch of knowledge is still avid to know. The greatest problems still find us with nothing but hypotheses to go on. Yet we keep asking ourselves for proofs. Brecht (1980)

This is a "sock it to them" paper—though much less so than the previous versions one of us have seen.

The authors argue in the paper that AdaBoost (without early stopping) is one of, if not the, most successful boosting algorithms, and they present this paper as a disproof of what the , rather amorphous community of statistical practitioners, represented by Friedman, Hastie and Tibshirani have:

(i) Pointed out as remediable flaws of the original Freund-Schapire boosting algorithm;

(ii) Given as remedies.

Evidently, that community should be able to respond on its own. We in fact, agree with some of the hypotheses Mease and Wyner's limited simulations lead them to, whether these are or are not embraced by statistical practitioners. But others we find dubious and unproven. Let us stress the positive first.

1. They argue that boosting does not behave like nearest neighbor for $d > 1$. Not only do we agree with this but would conjecture even further without any proof:

2. That, for reasonable sequences of d dimensional distributions, the random classification rules induced by the stationary measures corresponding to boosting forever, should in a suitable sense as $n, d \to \infty$ concentrate near the Bayes rule. However, an example below shows that the improvement from $d = 1$ to $d = 2$ can be slight.

3. We don't believe that boosting is consistent, in the sense of section 3.10, for any d, but indeed there is no disproof for $d > 1$.

4. We agree that a sharp explanation why, for classification, boosting *may* not overfit—that is, continues to reduce the probability of misclassification long past the point where all training sample observations are correctly classified has not been provided in the statistical (or the machine learning) literature.

5. We agree that using more complex basis functions may actually improve performance. This was analyzed theoretically for L_2 boosting by Bühlmann and Yu (2003).

6. We agree that there is a need for a convincing argument for basing an early-stopping algorithm of a classifier on a loss function that is not the classification loss. *A-priori* we do not expect that stopping on any criterion, other than minimum classification error will work in general, even if the classifier itself is based on minimizing this indirectly relevant criterion. However, it certainly can be that a good early stopping algorithm will be based on estimate of the loss with respect to something other than from the classification error. It was proven to be so, for example, with L_2-boost.

Since we have never been persuaded on theoretical grounds of the superiority of other "boosts", logit or L_2, over AdaBoost , we leave this battle to others.

Where we really part company with the authors of this "against the heretics" paper is on the issue of the desirability of early stopping.

Galileo tries to explain his young student, Andrea, the structure of the Copernican system, to make it so simple that Andrea will be able to explain it to his mother. He rotates him on a chair, and tells him that an apple in the center is the earth. However, Andrea is smart enough to understand that so far and not more can be deduced from examples.
Andrea: Examples always work if you're clever. Only I can't lug my mother round in a chair like you did me. So you see it's a rotten example really. And suppose your apple is the earth like you say? Nothing follows. Brecht (1980)

Everybody can produce examples. The authors gave two examples. Other commentators will bring their own. Here are ours. We consider $X \in \mathbb{R}^2$ uniform on 5 concentric circles. The classes were randomly assigned according to $P(Y=1|x) = \text{logit}(4\text{sgn}\,(\xi)\sqrt{(}\xi))$ where $\xi = \|x\| - 2|x_1|$ and $\text{logit}(\zeta) = e^{\zeta}/(1+e^{\zeta})$. The training sample includes 500 i.i.d. observations. 200 more observations were used for early stopping. I.e., stopping when the mean empirical classification error on these 200 observations was minimal. Finally another set of 1500 observations was used to evaluate the mean classification error of the AdaBoost procedure as a function of the number of observations. See Figure 1 for a plot of typical set of Xs and $P(Y=1|X))$. The weak classifier we used was a standard classification tree with 8-terminal nodes. The simulations were run with slightly different set-ups a few tens of times. There was no case that contradicted the results of the single experiment we will present next (but see later). Our example is small. Since the goal of the discussed paper is to suggest policy on the basis of two examples, even one (we believe) reasonable counterexample should give some pause.

In Figure 2(a) we show the classification error as a function of the number of iterations. The horizontal lines are, from the top down: The risk of the closest neighbor, the risk of AdaBoost with early stopping, and the Bayes Risk. It is clear in this example that AdaBoost starts quite nicely. The primitive early stopping technique we employed is enough to give a decent performance. However, the performance of AdaBoost starts to degenerate after some tens of iterations. After 800 hundred

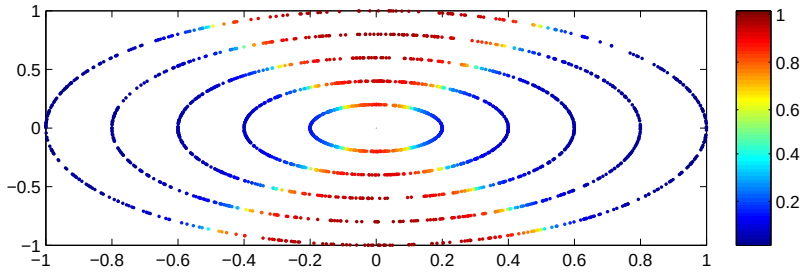


Figure 1: $P(Y=1|X)$ as function of location.

iterations it is only slightly better than the nearest neighbor classifier. Within the context of this example the "does not over-fit" property, can be understood at best as "it is simply a slow algorithm".

Figure 2(b) shows the root mean square error of AdaBoost implicit estimate of the probability as function of the number of iterations. It is clear that it degenerate much faster than AdaBoost performance. However, the implicit probability estimate is fair, as long as the classifier is in its prime. So, the authors' doubt whether boosting estimate probabilities cannot be based on this example.

However, we should mention that the apparent failure of the boosting algorithm to estimate probabilities is somewhat misleading. In Figure 2(c) we plot $F_m - F^0$, where F^0 is the ideal value, as a function of $P(Y=1|x)$ after $m=200$ iterations. $P(Y=1|x)$ is used here as proxy for the distance of the point from the $P(Y=1|x)=0.5$ boundary. What can be seen is that most of the error in the estimation of the $P(Y=1|x)$ comes from the easy to classify points. So, the boosting algorithm fails to estimate the probability where it does not really matter.

There are facts in life. One cannot invoke the church teaching or Aristotle's books in face of empirical facts. Galileo get annoyed by the insistence of the philosopher and the mathematician on using irrelevant arguments.

Galileo: My reasons! When a single glance at the stars themselves and my own notes make the phenomenon evident? Sir, your disputation becoming absurd. Brecht (1980)

The authors used a smart device to present the discrepancy between the one nearest neighbor classifier and AdaBoost. Namely, they compare their performance under a null hypothesis, where $P(Y=1|x)$ does not depend on x (this is true for Section 3.9 but not for 4.9, where for some unknown reasons something else was done). We did the same. The distribution of X and the sample sizes are as above, while $P(Y=1|x)\equiv 0.2$. The results are presented in Figure 3. The graphs are similar to Figure 2(a-b), in description and in essence. Boosting without early stopping is not like 1-NN, but it is not much better, at least for this tiny example.

We did observe a further interesting phenomenon when we added irrelevant explanatory variables. That is, 8 independent variables $X_3,\ldots,X_{10}$ were added, while the distribution of (X_1,X_2,Y) remains as above. The result was surprising. Adding these irrelevant variables improved the performance of "boosting-for-ever" to the level of the early stopping algorithm. This gives some credence to our conjecture 2.

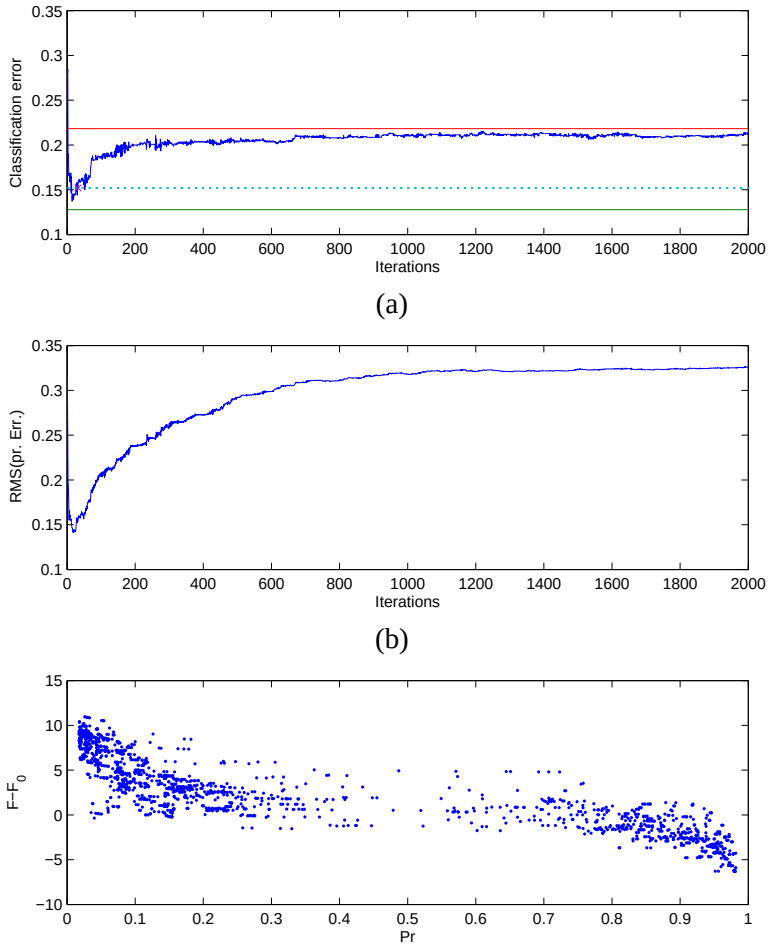
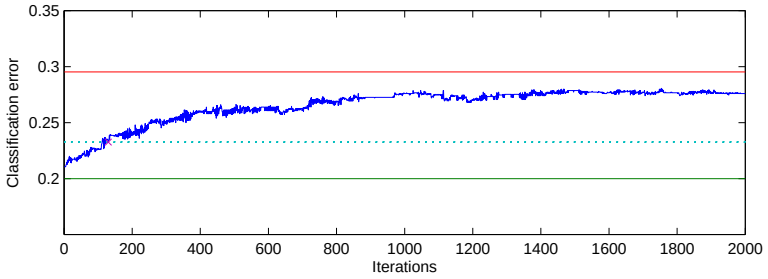
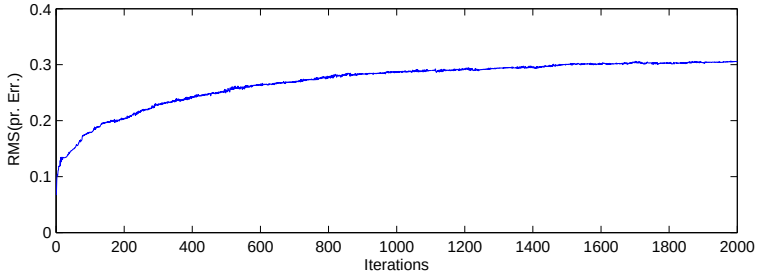


Figure 2: AdaBoost, with and without early stopping.



(a)



(b)

Figure 3: AdaBoost under the null hypothesis.

But, this is not the end of the story. We changed the the distribution a little. We left $P(Y|X_1,\ldots,X_{10}) = P(Y|X_1,X_2)$ as above. $X_3,\ldots,X_{10}$ are still independent of each other and independent of (X_1,X_2). However the distribution of (X_1,X_2) was changed to be discrete with 24 well isolated atoms. One can conceive the distribution as that of 24 columns with $500/24$ on the average observations. The Ys are i.i.d. given the column. AdaBoost with early stopping handled this situation very well and stopped after very few (e.g., 2) iterations. It essentially isolated the columns and left them intake. Boosting-for-ever stabilized nicely after very few iterations. However, as could be expected, it did break the columns. The result was that the misclassification error of boosting-for-ever was almost in the middle between the early stopping algorithm, and the strictly inferior 1-NN estimator.

A further point. The phenomenon of not overfitting for a long time is certainly interesting to investigate, but why this should be a virtue of a procedure is unclear, since it merely increases computation time at an often (perhaps usually) negligible improvement over stopping early.

AdaBoost is a mystery, but we, the weak, can solve only one toy problem at a time.
Galileo: Why try to be so clever now, that we at last have a chance of being less stupid? Brecht (1980)

AdaBoost was crowned by Leo Breiman as the best off-the-shelf classifier. It has some mysterious properties, particularly, sometimes continuing to improve off-sample performance even after completely collapsing on the data. It behaves better, sometimes, with 8-nodes trees, some other times with 256-node trees, and other times, mainly when examined by some statisticians, stumps are superior. It presents some mathematical challenges, which should be carefully investigated. However, many examples appearing in the literature are either very artificial, or the investigators

don't have a gold standard like the Bayes risk. We hope that this technique will grow out of its status of something like an art, to a scientifically justified method. But this is only a hope.

Acknowledgments

This work was supported in part by NSF grant DMS-0605236 and an ISF grant.

References

Bertolt Brecht. *Life of Galileo*. Arcade Publishing, New York, 1980. Translated by John Willet.

Peter Bühlmann and Bin Yu. Boosting with the l2 loss: Regression and classification. *Journal of the American Statistical Association*, 98:324–339, 2003.

Journal of Machine Learning Research 9 (2008) 187-194 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Peter Bühlmann BUHLMANN@STAT.MATH.ETHZ.CH
Seminar für Statistik, LEO C17
ETH Zurich
CH-8092 Zurich
Switzerland

Bin Yu BINYU@STAT.BERKELEY.EDU
Department of Statistics
367 Evans Hall #3860
Berkeley, CA 94720
USA

Editor: Yoav Freund

We would like to thank the authors for their provocative view on boosting. Their view is built upon some "contrary" evidence based on a particular simulation model. In our discussion, we argue that the structure of the simulation model explains many aspects of the "contrary" evidence. We touch upon the issue of shrinkage or small step-sizes, and we conclude that the "statistical view" provides constructive insights for applying boosting in a highly successful way.

The gradient and "statistical" point of view The gradient point of view of AdaBoost is, in our opinion, a great leap forward for understanding AdaBoost and deriving new variants of boosting now meaning much more than just AdaBoost. This view, which seems to be called the "statistical view" by Mease and Wyner (MW), has been pioneered by Breiman (1998, 1999), Friedman et al. (2000), Mason et al. (2000) Rätsch et al. (2001) and is not just a product of the statistics community. The gradient view of boosting allows transferring of the boosting methodology to many other contexts than just classification, see for example Meir and Rätsch (2003) or Bühlmann and Hothorn (2007) for an overview. We should also emphasize that the gradient view has never promised to explain everything about AdaBoost. Hence we are puzzled by the negative picture of this view painted in the paper under discussion: it differs greatly for most part from our experience and understanding of the statistical research on boosting. In particular, the MW paper seems to ignore simulation, real data and theoretical evidence about overfitting and early stopping (cf. Bartlett and Traskin 2007 regarding asymptotic theory for AdaBoost). We will discuss these issues in more details below.

The relevance of MW's counter-examples The evidences in MW are simulated "counter-examples". It is questionable that they are representative of situations encountered in practice. More importantly, with one exception, evidence of differences shown contradicting the so-called "statistical view" are 1 or 2 % in error rate. One wonders how important or meaningful these differences are in practice, even though they might be statistically significant. In any real world situation, the model used is for sure wrong and the approximation error of the model to the real situation could easily swallow these small differences in performance.

Furthermore, all the evaluation metrics in the MW paper are on statistical performance without any consideration of the computation involved or the meaning of the model derived. For large data problems, computation is an indispensable player and needs to be in the picture.

Additive decision boundary but non-additive logit-probabilities MW's model (in Section 3) is additive for the decision boundary. In terms of conditional probabilities $p(x) = \mathbb{P}[Y = 1|X = x]$ on the logit-scale, $\text{logit}(p(x))$ is *not* an additive function in the (feature) components of x.

Since the population minimizer of (gradient) AdaBoost or also of LogitBoost equals

$$F_{\text{pop}}(x) = 0.5 \cdot \text{logit}(p(x)) = 0.5 \cdot \log\left(\frac{p(x)}{1-p(x)}\right),$$

a (boosting) estimate will be good if it involves an effective parameterization. We believe that this is a central insight, which has been pioneered by Breiman (1998, 1999), Friedman et al. (2000) and which has been further developed by more recent asymptotic results on boosting. In the MW model, $F_{\text{pop}}(x)$ is non-additive in x while boosting with stumps yields an estimate $\hat{f}(x)$ which is additive in x. We think that this is the main reason why some of the figures in MW lead to "contrary" evidence: with our model, as illustrated below, the comparison of stumps versus larger trees for weak learners is always in favor of stumps, that is, stumps yield better performance and larger trees are more heavily overfitting which is the opposite finding to Figures 1, 2, and 11 in MW. MW's model in Section 4 involves only a single component of x and hence it is additive also on the logit-scale for the probabilities. But our own model described below does not confirm MW's statement that their findings "do not depend on a particular simulation model".

Other issues in MW concerning "contrary" evidence cannot be easily explained by the nature of the model.

Figure 3 intends to show that LogitBoost is worse than AdaBoost. The MW finding might seem relevant at 1000 iterations. But one doesn't need to go that far for both methods by early stopping. 100 or so iterations seems enough for stumps and 400 for 8-node trees. The performance difference is then less than 1%. Thus, having some computation savings in mind, early stopped LogitBoost is preferable.

Figure 4 tries to make the point that early stopping could hurt to lose about 1% performance when the total Bayes error is 20% and there is no structure to be learned. However, the 1000 iteration model undoubt-fully gives the wrong impression that something is there, while the early stopped model gives the correct impression that not much is to be learned. Hence we think early stopping is not hurting here. In addition, the starting value of boosting matters but this issue is ignored in standard AdaBoost. A (gradient) boosting algorithm should be started with $F_{init} = F_0 \equiv 0.5\log(\hat{p}/(1-\hat{p}))$ where $\hat{p}$ is the empirical frequency of $Y = 1$, cf. Bühlmann and Hothorn (2007). That is, boosting would (try to) improve upon the MLE from the "pure noise" model. Then, it is expected - and we checked this using gradient LogitBoost on the unbalanced example corresponding to Figure 4 in MW - that boosting will overfit from the beginning because the underlying structure is pure noise. The same idea could be applied to AdaBoost as well: in contrast, standard AdaBoost and MW start with the naive value $F_{init} = F_0 \equiv 0$.

Shrinkage and small step-sizes: another dimension for regularization MW makes some claims about additional shrinking using small step-sizes. As we understand Friedman (2001), he *never* intended to say that a shrinkage factor would avoid overfitting. Instead, he argued that introducing a shrinkage factor may improve the performance. Later, Efron et al. (2004) made the connection,

in the setting of linear models, that boosting with an infinitesimally small shrinkage step is equivalent to the Lasso under some conditions, and for general situations, Zhao and Yu (2007) showed that appropriate backward steps need to be added to boosting to get Lasso. This intriguing connection shows again that the shrinkage factor cannot eliminate overfitting. All what it achieves is a different, usually more powerful solution path (with a new regularization dimension through the step-size) than without shrinkage.

Our own findings with an additive model for logit-probabilities Now we devise our own simulation model to clarify some issues regarding overfitting, choice of weak learner and the estimation of probabilities via boosting. Arguably, as emphasized above, examples should not be overinterpreted. However, in view of many reported findings similar to what we show here, we feel that our examples are rather "representative" and we are reporting major instead of slight differences.

Our model is in the spirit of MW but on the logit-scale:

$$\begin{aligned}&\text{logit}(p(x)) = 8\sum_{j=1}^{5}(x_j - 0.5)\\ &Y \sim \text{Bernoulli}(p(x)),\end{aligned}$$

where $p(x) = \mathbb{P}[Y|X = x]$. This model has Bayes error rate approximately equal to 0.1 (as in the MW paper). We use this model as it is additive on the *logit-scale* for the probabilities since the population minimizer of (gradient) AdaBoost and (gradient) LogitBoost is $0.5\text{logit}(p(x))$. We use $n = 100$, $d = 20$ (i.e., 15 ineffective features), x as in MW and we show the results for one representative example with test set of size 2000. We skipped the repetition step over many realizations from the model: again, we think that one realization is representative and it mimics somewhat better the situation of analyzing one real data set.

We consider the misclassification test error, the surrogate loss test error (e.g., the test set average of $\exp(-y\hat{f})$ for AdaBoost) and the absolute error for probabilities

$$\frac{1}{2000}\sum_{i=1}^{2000}|\hat{p}(X_i) - p(X_i)|,$$

where averaging is over the test set.

All our computations have been done with MW's code for AdaBoost and the `R`-package `mboost` from Bühlmann and Hothorn (2007): we used stumps and larger trees as weak learners. By the way, MW's code is *not* implementing 8 node trees but trees which have on average about 6-8 terminal nodes (during the boosting iterations for this model). The results are displayed in Figures 1- 3. A comparison is also made to the naive estimator with $\hat{p}(x) \equiv 0.5$.

From this very limited experiment we find all facts that we view as important and typical for boosting:

1. Overfitting can be a severe issue when considering the test surrogate loss or for estimating conditional probabilities. In fact, overfitting is seen clearly for all three methods, that is gradient AdaBoost, LogitBoost and AdaBoost. In addition, the misclassification loss is much more insensitive with respect to overfitting. This has been pointed out very clearly in Bühlmann and Yu (2000) and in the rejoinder of Friedman et al. (2000).

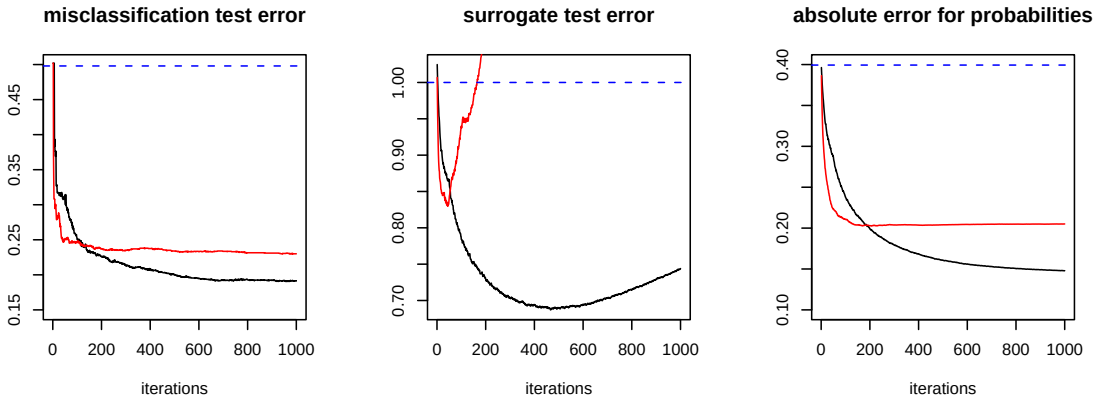


Figure 1: Gradient boosting with exponential loss (gradient AdaBoost). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator.

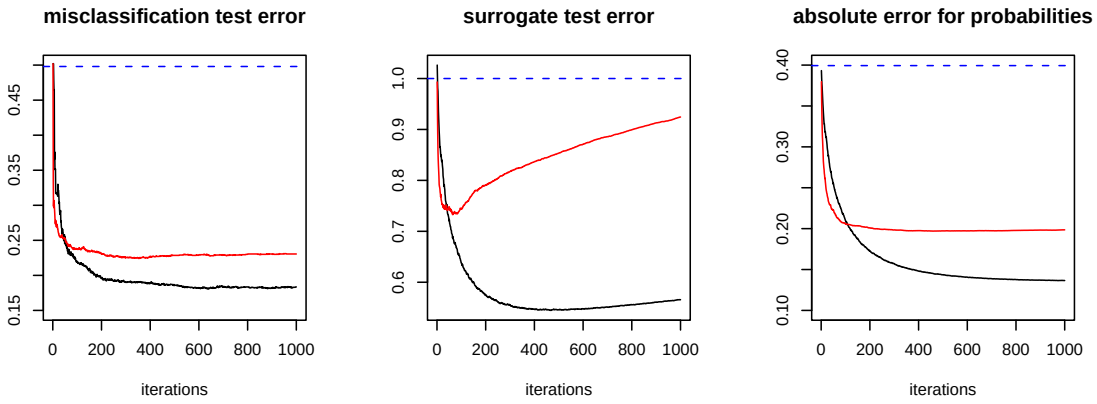


Figure 2: Gradient boosting with Binomial log-likelihood (gradient LogitBoost). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator.

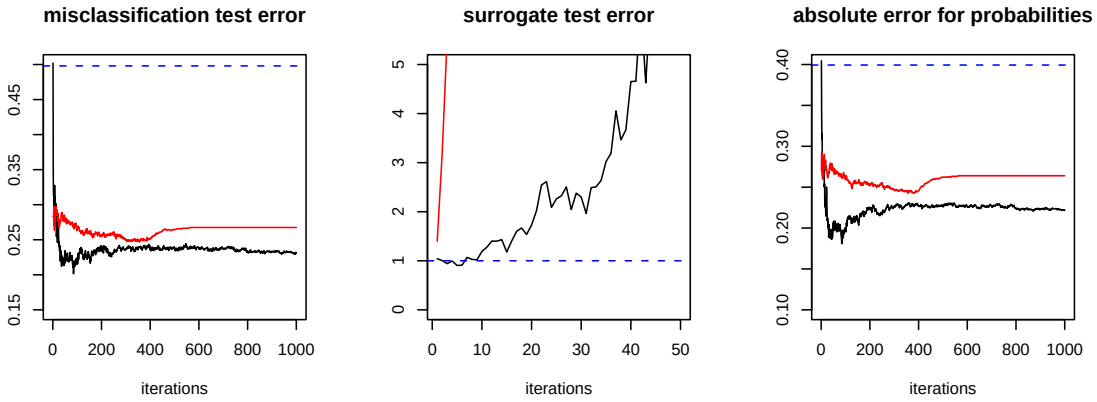


Figure 3: AdaBoost (as in MW). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator. More details are described in point 4 of our summary of findings.

2. Estimating conditional probabilities is quite reasonable when stopping early: as in point 1 above, we see very clearly that early stopping is absolutely crucial for all three methods. And LogitBoost with early stopping gives the best misclassification error and best probability estimate among the three.

3. Regarding the weak learner, larger trees are worse than stumps for our model where the conditional probability function is additive on the logit scale. The "statistical view" reveals the model behind AdaBoost and LogitBoost: we have to consider the logit-scale (the MW model is *not* additive in terms of the logit of conditional probabilities; note that for the decision boundary the scale doesn't matter while it does play a role for conditional probabilities).

   Larger trees do overfit more heavily for probability estimation or with respect to surrogate test loss. For non-additive models (for probabilities on the logit-scale), the overfitting will kick in later for large trees as the the underlying model requires a more complex fit to balance approximation ("bias") and stochastic error ("variance").

4. Somewhat more in line with the MW paper, the original AdaBoost has less a tendency to overfit than the gradient boosting version. The reason why AdaBoost with the larger tree in Figure 3 is staying constant after a while is due to the fact that the algorithm gets "stuck": it alternates back and forth and hence, the amount of overfitting is limited. At this stage of alternating behavior the estimated conditional class probabilities are very much concentrated around either zero or one (not shown but similar to Fig. 18 in MW), that is, overfitting has kicked in severely. We are not convinced that this "getting stuck" property of the algorithm is desirable, despite the consequence that a bound on overfitting is then obviously in action. The surrogate loss function in AdaBoost explodes much earlier (w.r.t. boosting iterations) and one needs to implement an upper bound in the program in order to avoid NA values (MW's code needs some small modification here!).

Our general understanding about Boosting and it's success Instead of going through all issues in MW, we choose instead to repeat several general understandings about boosting which were incorrectly questioned by the paper under discussion:

A. Overfitting does matter, and it is a function of the both the "bias" and "variance". Large trees do not overfit heavily in terms of classification error because:
(i) the misclassification loss is very insensitive to overfitting (see Bühlmann and Yu 2000 and the rejoinder of Friedman et al. 2000);
(ii) larger trees are not as "complex" as the number of nodes in them indicates since they are fitted in a greedy fashion (e.g., 8-node trees fitted by boosting are not 4 times as complex as stumps with two nodes).

   Most probably, the difference between plain vanilla AdaBoost and a gradient version of AdaBoost (as in MW) will *not* play a crucial role in terms of overfitting behavior; but gradient-based boosting seems somewhat more exposed to overfitting while AdaBoost can get stuck which naturally limits the amount of overfitting (on a single data-set).

B. Early stopping, particularly for probability estimation, is very important (because of overfitting) and brings computational savings. The supporting theory is given in, for example, Zhang and Yu (2005), Bühlmann (2006), Bartlett and Traskin (2007) and Bissantz et al. (2007).

C. Estimating probability via boosting is often quite reasonable. It is essential though to tune the boosting algorithm appropriately: a good choice is to do early stopping with respect to the log-likelihood test score (see next point regarding surrogate and evaluating loss).

D. It is important to distinguish between surrogate loss (implementing loss) and loss (evaluating loss) function. For example, there is no surprise that it can happen with AdaBoost that the training misclassification error is zero while the test set misclassification still decreases.

The usage of boosting as we have advocated in our works, and this is very much in line with Friedman et al. (2000) and their subsequent works, has proven to be very competitive and successful in applications. Gao et al. (2006) describe a successful application of boosting to a language transliteration problem. Lutz (2006) has won the performance prediction challenge of the world congress in computational intelligence in 2006 (WCCI 2006): he was using early-stopped LogitBoost with stumps. Part of his success is probably due to careful choice of choosing the stopping iteration: according to personal communication (he has been a former PhD student of the first author of this discussion), he stopped before reaching the minimal value of a cross-validation scheme. In summary, he did not take any of the findings from MW into account (he didn't know the paper at that time, of course). Maybe his success is more convincing evidence that LogitBoost with (i) its "natural" loss function for a binary classification problem, and using (ii) early stopping, (iii) simple weak learners and (iv) a small step size (i.e., shrinkage factor) often works surprisingly well. Other references about successful applications of gradient-based boosting can be found in Bühlmann and Hothorn (2007) which includes the `R` package `mboost` (standing for model-based boosting) for numerous application areas ranging from classification, regression, generalized regression to survival analysis.

References

P. L. Bartlett and M. Traskin. AdaBoost is consistent. *Journal of Machine Learning Research*, 8: 2347–2368, 2007.

N. Bissantz, T. Hohage, A. Munk, and F. Ruymgaart. Convergence rates of general regularization methods for statistical inverse problems and applications. *SIAM Journal on Numerical Analysis*, 45:2610–2636, 2007.

L. Breiman. Arcing classifiers (with discussion). *The Annals of Statistics*, 26:801–849, 1998.

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

P. Bühlmann. Boosting for high-dimensional linear models. *The Annals of Statistics*, 34:559–583, 2006.

P. Bühlmann and T. Hothorn. Boosting algorithms: regularization, prediction and model fitting (with discussion). *Statistical Science*, 22, 2007.

P. Bühlmann and B. Yu. Discussion of "Additive logistic regression: a statistical view" (J. Friedman, T. Hastie and R. Tibshirani, auths.). *The Annals of Statistics*, 28:377–386, 2000.

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

J.H. Friedman. Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29:1189–1232, 2001.

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

J. Gao, H. Suzuki, and B. Yu. Approximation Lasso methods for language modeling. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 225–232, 2006.

R.W. Lutz. LogitBoost with trees applied to the WCCI 2006 performance prediction challenge datasets. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2006.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, 2000.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Computer Science, pages 119–184. Springer, 2003.

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

T. Zhang and B. Yu. Boosting with early stopping: convergence and consistency. *The Annals of Statistics*, 33:1538–1579, 2005.

P. Zhao and B. Yu. Stagewise Lasso. *Journal of Machine Learning Research*, 8:2701–2726, 2007.

Journal of Machine Learning Research 9 (2008) 195-201 Published 2/08

Evidence Contrary to the Statistical View of Boosting: A Rejoinder to Responses

David Mease MEASE_D@COB.SJSU.EDU
Department of Marketing and Decision Sciences
College of Business, San Jose State University
San Jose, CA 95192-0069, USA

Abraham Wyner AJW@WHARTON.UPENN.EDU
Department of Statistics
Wharton School, University of Pennsylvania
Philadelphia, PA, 19104-6340, USA

Editor: Yoav Freund

1. Introduction

We thank the discussants for their comments on our paper. We also thank the editors for arranging the discussions. Many interesting points have been raised by the discussants. We can not respond to everything, but we do include a section addressing the main points of each discussant. Following these, we provide a final section in which we give some general concluding remarks.

Many of the discussants comment on the overfitting of boosting. Different authors will have different ideas of what the term overfitting means in the context of boosting, but for clarification throughout this rejoinder we will define overfitting as a positive slope for a specified loss metric as a function of the iterations. Specifically, the loss metric we focus on is misclassification error, although we understand that some of the discussants are concerned about other loss functions which quantify probability estimation accuracy rather than classification accuracy. The importance of focusing on misclassification error is underscored in the discussion by Freund and Schapire who remind us that AdaBoost is an algorithm for carrying out classification, not probability estimation.

2. Rejoinder for Kristin P. Bennett

Bennett provides a useful perspective on the situation by studying the convergence of boosting algorithms from the optimization point of view. We agree that this aspect of the problem is too often overlooked by researchers.

In studying the convergence of the algorithms, Bennett touches on a number of important considerations. For example, she mentions cycling in the context of stumps and notes that the cycling results in boosting using only a small number of unique trees. The number of unique trees is rarely noted by researchers in empirical investigations. Bennett's studies also lead her to concur that boosting algorithms sometimes benefit from a bagging type of self-averaging during what she calls the "overtraining" stage. ("Overtraining" as defined by Bennett should not be confused with "over*fitting*" as we have defined it). Another point on which we strongly agree with Bennett is that

boosting's resistance to overfitting can occur for different reasons in different contexts. We believe that this is one reason why researchers have difficulty in coming to agreement regarding various explanations for boosting.

In addition to studying convergence, Bennett also looks at the margin of the classification rules. We mentioned margin theory in our paper only briefly since our focus was on the statistical view of boosting, and margin theory is generally separate from this view. It is our hope, however, that by finding holes in the accepted statistical view we are encouraging researchers to approach the problem from different perspectives to help explain the phenomena left unexplained by the statistical view. Unfortunately, in this case Bennett points out that examination of the margin still fails to explain the results of the experiment in Section 3.2.

We believe that studying boosting as an algorithm (rather than as a statistical model) in the way Bennett has done can be quite helpful in understanding some of its remaining mysteries. We are glad that our examples have inspired this type of investigation.

3. Rejoinder for Andreas Buja and Werner Stuetzle

One of the main points argued by Buja and Stuetzle is that most of the current literature on boosting does not explain its variance reduction. They argue that a complete view of boosting should explain both its ability to reduce bias and variance. We certainly agree with this point. In fact, some of the examples in our paper such as those in Sections 3.4 and 4.4 illustrate this quite well. It is interesting that Buja and Stuetzle site references from the early research on boosting which argue in this same direction. It is a shame that more attention has not been given to boosting's ability to reduce variance in addition to bias in more recent research. We hope that our paper helps to rejuvenate research on this aspect of boosting.

Buja and Stuetzle go on to argue that there is often a need for more than a black box classifier which produces small misclassification error. Some applications call for interpretable and diagnostic models and/or conditional class probability estimates. This is certainly true, and we agree that research that extends boosting in these directions is quite welcome. However, in carrying out this research it is important to be honest about situations in which the theory does not explain the performance of traditional boosting algorithms for classification. One of our purposes in writing this paper was to promote this honesty.

4. Rejoinder for Yoav Freund and Robert E. Schapire

Freund and Schapire focus their discussion on margin theory. It is quite interesting that margin theory has not been embraced much at all by the statistical community. In fact, Freund and Schapire's paper "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods" appeared less than two years before the Friedman, Hastie and Tibshirani paper "Additive Logistic Regression: A Statistical View Of Boosting" in the same journal (*Annals of Statistics*). Despite this, the statistics community has largely ignored it in favor of the more familiar theory in the latter paper.

Freund and Schapire make a case for margin theory by arguing that this theory explains the results of the experiment in Section 3.1 while obviously the statistical theory does not. However, it should be noted that Bennett believes margin theory still does not explain the results for Section 3.2. Despite this, we still believe that margin theory is worth pursuing, especially given the large number of inconsistencies between the statistical view and the reality of the simulations presented

in our paper. As we have mentioned, we wrote the paper with the goal of promoting alternative explanations for boosting other than the statistical view, which leaves much unexplained. Margin theory is one such alternative explanation.

Additionally, Freund and Schapire make a number of other points on which we agree and wish to underscore. They argue that in the statistics research on boosting too much importance is placed on class probability estimation over class estimation, the use of stumps over larger trees and theoretical questions of consistency and asymptotic variance over more relevant theoretical questions.

5. Rejoinder for Jerome Friedman, Trevor Hastie and Robert Tibshirani

Friedman, Hastie and Tibshirani argue that many of the ten statements made in our paper should not be ascribed to the statistical view as laid out in Friedman et al. (2000). This is understandable, and in the case of the similarity with nearest neighbor algorithms, for example, we have even noted this in our original paper. However, other statements could be argued to follow fairly directly, whether that is the intent of the authors or the fault of the reader. At the very least, we believe it is fair to say the statistical view offers very little to help explain the non-intuitive nature of the results in our paper.

Of all ten statements, the most direct relationship to the work of Friedman, Hastie and Tibshirani is the idea in Sections 3.1 and 4.1 that stumps should be used for additive Bayes rules. We believe the authors would agree that this is the strongest connection to their work, which is why they focused the majority of their discussion on the experiment in Section 3.1. We also think it is refreshing that they admit that they "are not sure why" the results of this experiment are such, but they do produce some graphs to try to understand this better. Some of their graphs show the performance of the probability estimates. It is argued in the discussion by Freund and Schapire that probability estimation for boosting has very little to do with boosting's classification performance. We agree and for this reason we will not comment on the graphs for probability estimation. However, the graphs showing misclassification error are of interest and we discuss these below.

Friedman, Hastie and Tibshirani's Figure 1 shows that using shrinkage in our original experiment causes stumps to "win handily". We note, however, that the shrinkage causes overfitting, so it also becomes necessary to stop the boosting process before 600 iterations to realize any advantage over the 8-node trees. In practice the optimal stopping time is not known, and a fair comparison would require incorporating the uncertainty in the estimation of this value.

The right panel of their Figure 2 shows that with a larger training sample size of $n = 2000$ the overfitting caused by shrinkage is not as severe and the stumps maintain their advantage over the full 1000 iterations. However, by 1000 iterations the overfitting for the stumps has caused the gap with the 8-node trees to close considerably and extrapolation would suggest that additional iterations would result in this gap becoming even smaller or disappearing altogether. Meanwhile, the 8-node trees again show no signs of overfitting.

We believe the more interesting research question with regard to Friedman, Hastie and Tibshirani's Figures 1 and 2 is not which algorithm performs best for a certain stopping time and sample size, but rather why all algorithms display overfitting with regard to misclassification error except the 8-node trees without shrinkage. The authors state that the larger trees can have a bagging effect, and we certainly agree with this. We feel that understanding this effect better (as well as understanding why shrinkage can destroy this effect) is essential to gaining a better understanding of boosting.

The left panel of Friedman, Hastie and Tibshirani's Figure 3 shows the effect of using Bernoulli loss rather than exponential loss. With Bernoulli loss all algorithms show overfitting. This is another curiosity not explained by the statistical view of boosting. In fact, the paper by Friedman et al. (2000) seems to suggest the opposite should be expected.

In their final paragraph Friedman, Hastie and Tibshirani welcome "*constructive* counter-arguments and alternative explanations for the success of boosting methods." We will make a couple of comments here in response to this. First, we believe that explanations such as the variance reducing bagging effect of boosting fall into this category. Our paper is certainly not the first to suggest this notion, nor do we offer a theoretical explanation for the phenomenon, but our paper does stress the importance of not overlooking this effect and thus we hope promotes constructive research on this topic. Secondly, we feel that researchers currently embrace the statistical view too strongly, and for this reason it is difficult for researchers to offer any alternative explanations without first tearing down the current beliefs to some degree. We base this statement on our own experience. For instance, an early version of Mease et al. (2007) was rejected for publication by a different outlet. That paper offers a method for estimating probabilities using AdaBoost. Two of the three referees rejected the paper arguing that in order to estimate probabilities using boosting it would be sufficient to use LogitBoost in place of AdaBoost.

6. Rejoinder for P. J. Bickel and Ya'acov Ritov

Bickel and Ritov begin by identifying some points we made in our paper with which they agree. The amount of agreement is substantial. The disagreement is focused largely on what is generally regarded to be the most mysterious property of the AdaBoost algorithm: its ability to reduce the (test) error rate long after the training data has been fit without error. Other classifiers such as CART, neural nets, LDA and logistic regression perform optimally only with some sort of appropriate early stopping to prevent over parameterization and overfitting of the data. It is understandable that Bickel and Ritov's negative remarks focus on this particular issue, as it is the feature of AdaBoost most at odds with the statistical view.

The example provided by Bickel and Ritov is a model for which early stopping is essential to achieve optimal classification performance. If AdaBoost is not stopped after 10 or 20 iterations in their example, it will overfit the data and the generalization error will increase steadily. Their example is not the first of its kind, but rather is typical of the simulations used to provide empirical support for the statistical view. We addressed this point directly in our paper:

> *Such examples in which overtting is observed often deal with extremely low-dimensional cases such as $d = 2$ or even $d = 1$. By experimenting with the simulation code provided along with this paper, one can confirm that in general AdaBoost is much more likely to suffer from overtting in trivial low-dimensional examples as opposed to high-dimensional situations where it is more often used.*

The example provided by Bickel and Ritov is of this same spirit. It is a $d = 2$ dimensional model that lives on $d = 1$ dimensional circular manifolds. Since it is well known that AdaBoost will converge to the nearest neighbor classifier in one dimension, the results for this simulation are not unexpected. Along these same lines, Bickel and Ritov provide further evidence for our claim that overfitting is largely a symptom of trivial low dimensional examples by observing that when

they add 8 additional dimensions with no signal the overfitting disappears. The authors refer to this phenomenon as "surprising", but we would argue that this should also be expected.

Bickel and Ritov note interesting changes when they discretize some of the predictors. We did not discuss this in our paper, but the difference in behavior for discrete data and continuous data is tremendous. We learned first hand about this in Mease et al. (2007) when we artificially introduced discreteness by adding replicates. Because of this difference, we focused our current paper on the case of continuous data. The case of discrete data relates more to what Friedman et al. (2000) called the *population version* of boosting. The statistical view of boosting is largely based on this population version of boosting, and thus the statistical view becomes more relevant for discrete data.

Bickel and Ritov also bring up probability estimation in their discussion. Probability estimation is a popular topic among statisticians with regard to boosting, but again we will not go into any detail here because we agree with Freund and Schapire that probability estimation is not the central topic, but rather classification. It is curious, however, that Bickel and Ritov state that "most of the error in the estimation" of the probabilities "comes from the easy to classify points". The truth of this statement depends on how one measures the error in probability estimates. For example, if one computes the RMSE between the true probabilities and the estimated probabilities for the data in Bickel and Ritov's Figure 2(c), it can be observed that the RMSE is actually highest near the points with a true probability of 1/2.

In conclusion, Bickel and Ritov have presented a couple of low dimensional examples in which overfitting occurs unless early stopping is applied, while our paper presents higher dimensional examples where this is not the case. Since "everybody can produce examples" as Bickel and Ritov state, one may wonder which set of examples is more useful to study for the purpose of understanding boosting. We make the argument for our examples based on the fact that higher dimensional examples are more common of a use case for boosting, and that our examples are more similar to the many examples in which overfitting does not occur that first led practitioners to be attracted to boosting originally. We feel that understanding such examples is most useful for understanding boosting.

7. Rejoinder for Peter Buhlmann and Bin Yu

Of the six discussions, Buhlmann and Yu seem to be the least accepting of the empirical findings in our paper. They state that the "main reason" why we obtain contrary evidence in many cases is the functional form of the simulation model, and they propose considering a model that is additive on the logit scale. The model they propose turns out to be a special case of our model in Section 5 with $k = 8$. As stated in the paper, we had already considered the results for this specific model carefully using various values of k, and those results were consistent qualitatively with the other simulations in the paper. Thus, we were surprised to read that Buhlmann and Yu based any disagreement on results from this model. On closer inspection, we learned that the discrepancy is most likely due to sampling error. Buhlmann and Yu considered only a single repetition. The plot in the first panel of their Figure 3 shows that for that single repetition, the 8-node trees begin to overfit near 400 iterations and the stumps have a lower misclassification error throughout the 1000 iterations. However, if the results are averaged over many repetitions, one can learn that this behavior is not true in aggregate. The first plot in Figure 1 in this rejoinder shows the misclassification error for the Buhlmann and Yu model averaged over 100 repetitions. The stumps overfit while the 8-node trees do not, and the 8-node trees achieve a lower misclassification error than the stumps at 1000

iterations. Thus, the results for the Buhlmann and Yu model are consistent with the results for the other simulations in our paper.

It should be noted that with the small sample size of $n = 100$ chosen by Buhlmann and Yu, there is indeed often a problem with floating point overflow errors in R as they mention. They dealt with this issue by making a modification to our code (which is an alternative explanation for the discrepancies in the results). We avoided this issue when making the first plot in Figure 1 in this rejoinder by considering 100 randomly chosen repetitions for which the floating point overflow error did not occur. For readers who are uncomfortable with this, there are a couple of other ways of handling the problem. First, if a larger sample size is chosen then the problem goes away completely. For example, the second plot in Figure 1 in this rejoinder considers our original sample size of $n = 200$. For this sample size there were no problems with overflow errors in 100 repetitions. Again, one can see that the 8-node trees lead to a lower misclassification error at 1000 iterations and do not overfit, while the stumps do overfit. A second way of dealing with the overflow errors for the sample size of $n = 100$ is to consider only the first 500 iterations. When we ran only 500 iterations we did not observe any overflow errors in 100 repetitions, and the results were consistent with those in the first plot in Figure 1 in this rejoinder.

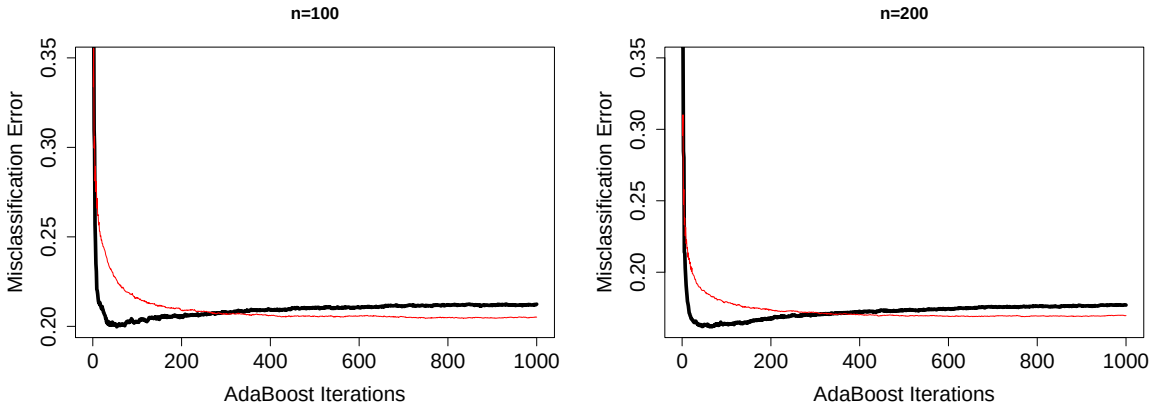


Figure 1: Comparison of AdaBoost with Stumps (Black, Thick) and 8-Node Trees (Red, Thin) for the Buhlmann and Yu Model Using 100 Repetitions and Sample Sizes of $n = 100$ and $n = 200$

Thus, concerning this simulation model proposed by Buhlmann and Yu, we conclude that the results are largely consistent with the other results in our paper. However, Buhlmann and Yu also discuss our Figures 3 and 4 independently of their simulation model. We address this below.

With regard to LogitBoost in our Figure 3, Buhlmann and Yu mention that with early stopping LogitBoost does almost as well as AdaBoost. This assumes one can estimate the optimal stopping time well, when in practice the stopping time estimation using LogitBoost can be difficult. In fact, we mentioned in our paper that the authors of the particular LogitBoost code we used reported that the stopping estimation was not effective for their purposes. Conversely AdaBoost performs fine in our Figure 3 without early stopping. Also, considering that the creators of LogitBoost (Friedman, Hastie and Tibshirani) mentioned in their discussion that they "have moved on" from LogitBoost, there seems to be little left to be said in its defense.

With regard to our Figure 4, Buhlmann and Yu seem unimpressed by this example which shows how AdaBoost can reduce variance. In fact, despite its excellent performance they lament that the classifier "gives the wrong impression." This is an interesting statement and illustrates the desire on the part of the statistical community to view AdaBoost as an interpretable model rather than a black box classifier. Certainly a model can give one the wrong impression but a classifier arguably only classifies well or does not classify well.

In conclusion, Buhlmann and Yu are considerably opposed to our most important claims: 1) that large trees generally work better than stumps and 2) that overfitting is usually not a problem and 3) that early stopping initiatives are often not only unnecessary but also counterproductive. With regard to classification we have shown that their simulation does not provide strong evidence to support their case. However, when the goal is the more difficult problem of conditional probability estimation, we will not offer any disagreement. We have not focused our analysis on this problem, and we do not feel that boosting algorithms should be regarded as "state-of-the-art" probability estimators, since they are usually outperformed by competitors like Random Forests or even logistic regression. Rather, it is the statistical view that asserts that AdaBoost works, erroneously in our opinion, because it estimates probabilities. Indeed, the efforts of Buhlmann and Yu to understand and improve the performance of boosting for probability estimation is productive and worthwhile. Where we part company is in name only. We question whether it is logical to continue to call the algorithms boosting algorithms since there is a considerable disconnect from the original AdaBoost algorithm for classification.

8. Conclusion

We believe the discussions provided by the six sets of authors have been extremely valuable. We are encouraged by the amount of discussion of two main ideas which we feel will lead to a better understanding of boosting.

First, the discussions have helped to clarify that (out of sample) minimization of the surrogate loss function (and equivalently probability estimation) is often a very different problem from (out of sample) minimization of misclassification error. The original AdaBoost algorithm was intended for the latter but much research in the statistical community has focused on the former.

Secondly, with regard to minimization of misclassification error, the idea that boosting is reducing variance has been acknowledged in the discussions. The extent to which this phenomenon is essential for boosting's success and of interest as a research topic can be debated, but we are encouraged to see it acknowledged by such a prominent group of researchers.

References

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28:337–374, 2000.

D. Mease, A. Wyner, and A. Buja. Boosted classification trees and class probability/quantile estimation. *Journal of Machine Learning Research*, 8:409–439, 2007.