The Annals of Statistics, 26(3):824-832, 1998.

Discussion of the paper "Arcing Classifiers" by Leo Breiman

Yoav Freund **Robert E. Schapire**
AT&T Labs
180 Park Avenue
Florham Park, NJ 07932-0971 USA
{yoav, schapire}@research.att.com

September 19, 1997

We would like to thank Leo Breiman for his interest in our work on boosting, for his extensive experiments with the AdaBoost algorithm (which he calls arc-fs) and for his very generous exposition of our work to the statistics community. Breiman's experiments and our intensive email communication over the last two years have inspired us to think about boosting in new ways. These new ways of thinking, in turn, led us to consider new ways for measuring the performance of the boosting algorithm and for predicting its performance on out-of-sample instances.

It is exciting for us to have this communication channel with such a prominent practical statistician. As computer scientists we try to derive our algorithms from theoretical frameworks. While these frameworks cannot capture all of our prior beliefs about the nature of the real-world problems, they can sometimes capture important aspects of the problem in new and useful ways. In our case, boosting was originally derived as an answer to a theoretical question posed by Kearns and Valiant [7] within the PAC framework, a model for the study of theoretical machine learning first proposed by Valiant [15]. We probably would have never thought about these algorithms had the theoretical question not been posed. On the other hand, an experimental statistician such as Leo is usually more interested in the actual behavior of algorithms on existing data-sets and pays a lot of attention to the actual values of various variables during the run of the algorithm. Running AdaBoost on several synthetic and real-world datasets, Breiman observed that the algorithm has surprisingly low generalization error, which, while consistent with our theory at the time, was not predicted by it.[1]

It is this challenge from the experiments of Breiman reported here, as well as those of Drucker and Cortes [3] and Quinlan [10], that prompted us to think harder about the problem and come up with a new theoretical explanation of the surprising behavior, which we describe in our paper with Bartlett and Lee [13]. This explanation suggests new measurable parameters, which can be tested in experiments, and the adventure continues! Theory suggests new algorithms and experiments, while experiments give rise to new observations which challenge the theory to come up with tighter bounds.[2]

Our communication with Leo has been challenging and exciting. We hope to see further communication developing between researchers in computational learning theory and statistics.

1 Boosting and bagging

Breiman's paper is about improving the performance of a *learning algorithm*, sometimes also called a prediction algorithm or classification method. Such an algorithm operates on a given set of *instances*

[1] Theories usually give only upper and lower bounds on the actual performance of algorithms. The gap between these bounds is a reflection of the degree to which our theories fail to reflect the world and of our shortcomings in the mathematical analysis.

[2] All the theoretical bounds to which we refer in this discussion are *non-asymptotic* and can be used to generate specific numerical bounds for finite sample sizes. However, these bounds are still numerically pretty loose.

(or cases) to produce a classification rule which we refer to as a *hypothesis*. The goal of a learning algorithm is to find a hypothesis with low *generalization* or *prediction error*, i.e., a low misclassification rate on a separate test set.

Bagging and boosting are two general methods for improving the performance of a given learning algorithm, which we call the *base* learning algorithm. On a certain level, the algorithms are very similar; they both work by feeding perturbed versions of the training set to the base learning algorithm and combining the resulting rules by a majority vote.

While these similarities are apparent, there are some important differences between the two algorithms. Probably the most important difference is that the perturbations introduced by bagging are *random* and *independent* while the perturbations introduced by boosting (on a given training set) are chosen *deterministically* and *serially*, with the nth perturbation *depending strongly* on all of the previously generated rules.

In this paper, Breiman uses boosting-by-resampling, instead of boosting-by-reweighting and in this way combines the two methods.[3] However, in Section 8.2, Breiman reports results from using the deterministic version of boosting and his results indicate that in the experiments reported here randomization is not an important element. On the other hand, Breiman has informed us that there are other, yet unpublished, experimental results regarding boosting of unpruned decision trees, in which boosting-by-resampling seems to have an advantage over boosting-by-reweighting.

It seems to us that the difference between boosting-by-reweighting and bagging should not be overlooked. Breiman analyzes both bagging and boosting in terms of the bias and variance decomposition of the error. We argue that this analysis is not appropriate for boosting. We have proposed a different analysis that seems to be appropriate for boosting but leaves out the effects of randomization. Giving a good theory of these randomization effects and a characterization of the cases in which they are advantageous is an interesting open problem.

The rest of this discussion is organized as follows. In Section 2, we give a historical perspective of the development of boosting algorithms. In Section 3, we summarize our arguments against explaining boosting using the bias-variance decomposition. In Section 4, we sketch our explanation for the small generalization error of boosting (a full description of this analysis appears in our paper with Bartlett and Lee [13]). We conclude by describing some practical and theoretical open questions.

2 Boosting in and out of the PAC framework

The differences between boosting and bagging reflect the very different frameworks in which the two algorithms have been developed. Breiman [1, 2] developed bagging in the context of reducing the variance of learning algorithms, while boosting was developed as an answer to a theoretical question posed by Kearns and Valiant [7] within the PAC learning literature. Stated somewhat informally, the question is: suppose we have a computationally efficient learning algorithm that can generate a hypothesis which is slightly better than random guessing for *any* distribution over the inputs. Does the existence of such a "weak" learning algorithm imply the existence of an efficient "strong" learning algorithm that can generate arbitrarily accurate hypotheses?

The answer to this question, given first by Schapire [12], is "yes." The proof is constructive, i.e., it describes an efficient algorithm which transforms any efficient weak learning algorithm into an efficient strong one. Later, Freund [5] described a simpler and considerably more efficient boosting algorithm called *boost-by-majority*. Our AdaBoost algorithm is the most recent of the proposed boosting

[3] In boosting-by-reweighting, we assume that the learning algorithm can work directly with a weighted training sample, while in boosting-by-resampling, training samples are generated by picking examples at random according to the distribution over the training set.

algorithms [6]. While nearly as efficient as boost-by-majority, AdaBoost has certain practical advantages over the preceding boosting algorithms which we discuss below.

The goal of all boosting algorithms, starting with Schapire's, has always been to generate a combined hypothesis whose *generalization error* is small. One of the *means* for achieving this reduction has been to reduce the error of the combined hypothesis on the the training set. The expectation that reducing the training error of the boosted hypothesis will also reduce the test error was justified by appealing to uniform convergence theory [16] (VC theory) or to arguments that rely on sample compression [4]. As a result of this analysis, the expectation was that there would be no point in running boosting beyond the point at which the training error of the combined hypothesis is zero.

It is only with the experimental work of Drucker and Cortes [3], Quinlan [10] and the work of Breiman reported here, that it was realized that it is sometimes worthwhile to run the boosting algorithm beyond the point at which the error of the combined hypothesis is zero. The fact that doing so can cause the test error to decrease even further was very surprising to us, and, indeed, completely contradicted our intuitions about the relations between the training error and test error of learning algorithms.

The fact that these discoveries were made on the latest boosting algorithm "AdaBoost" rather than previous boosting algorithms has possibly more to do with the fact that this algorithm is very efficient in practice and less to do with its generalization properties. Indeed, both previous boosting algorithms can be classified as "arcing" algorithms according to Breiman's terminology. It would be interesting to see whether these previous boosting algorithms also decrease the test error after a training error of zero has been reached. In addition, boost-by-majority is quite different than AdaBoost in the way it treats outliers, which suggests that it might be worth exploring experimentally.

The reason that AdaBoost is especially efficient in practice is that it is *adaptive*. Previous boosting algorithms had to receive as input, in advance of observing the data, a parameter $\gamma > 0$. This parameter is the amount by which we believe that our classification method is better than random guessing. More formally, in order for the proof on the prediction error of the combined rule to work, the hypotheses generated by the weak learning algorithm have to all have error smaller than $1/2 - \gamma$. In practice, it is hard to know how to set γ in advance. There are two aspects to this problem which we discuss in turn:

1. As Breiman remarks in the appendix, it might be the case that for some distributions over the inputs there is *no* hypothesis whose error is smaller than $1/2$. This reflects a limitation of the original PAC framework, in which boosting was originally analyzed, in which the assumption of weak learning is made uniformly with respect to *all* distributions. However, boosting algorithms can be analyzed outside this framework, as was done for AdaBoost where the theoretical framework of the analysis was expanded to better reflect the real-world problems. The cost of this expansion is that in the more general framework we lose the clear and simple definition of a "weak learner." In other words, the degree to which a learning algorithm can be boosted is characterized only as "an algorithm that gives small errors when run under AdaBoost." This is a very unsatisfying characterization because it involves both the base classification method and the boosting algorithm, while we would like to have a characterization that involves only the classification method.[4]

2. It may be possible to use boost-by-majority in cases where the weak learning algorithm depends on the input distribution. The fact that we need to know γ in advance may not really be such a big problem because we might be able to run the algorithm several times and perform a binary search for the largest value of γ that works. However, there is an important *computational efficiency* problem. The problem is that the number of iterations required by boost-by-majority grows like

[4] A somewhat different framework was used to give an extended analysis of boost-by-majority which, to some degree, overcomes this difficulty [5, Section 4.1].

		Kong & Dietterich [9] definitions						Breiman definitions					
		stumps			C4.5			stumps			C4.5		
name		–	boost	bag	–	boost	bag	–	boost	bag	–	boost	bag
waveform	bias	26.0	3.8	22.8	1.5	0.5	1.4	19.2	2.6	15.7	0.9	0.3	1.4
	var	5.6	2.8	4.1	14.9	3.7	5.2	12.5	4.0	11.2	15.5	3.9	5.2
	error	44.7	19.6	39.9	29.4	17.2	19.7	44.7	19.6	39.9	29.4	17.2	19.7
twonorm	bias	2.5	0.6	2.0	0.5	0.2	0.5	1.3	0.3	1.1	0.3	0.1	0.3
	var	28.5	2.3	17.3	18.7	1.8	5.4	29.6	2.6	18.2	19.0	1.9	5.6
	error	33.3	5.3	21.7	21.6	4.4	8.3	33.3	5.3	21.7	21.6	4.4	8.3
threenorm	bias	24.5	6.3	21.6	4.7	2.9	5.0	14.2	4.1	13.8	2.6	1.9	3.1
	var	6.9	5.1	4.8	16.7	5.2	6.8	17.2	7.3	12.6	18.8	6.3	8.6
	error	41.9	22.0	36.9	31.9	18.6	22.3	41.9	22.0	36.9	31.9	18.6	22.3
ringnorm	bias	46.9	4.1	46.9	2.0	0.7	1.7	32.3	2.7	37.6	1.1	0.4	1.1
	var	–7.9	6.6	–7.1	15.5	2.3	6.3	6.7	8.0	2.2	16.4	2.6	6.9
	error	40.6	12.2	41.4	19.0	4.5	9.5	40.6	12.2	41.4	19.0	4.5	9.5

Table 1: Bias-variance experiments using boosting and bagging on synthetic data. Columns labeled with a dash indicate that the base learning algorithm was run just once.

$1/\gamma^2$ so, for example, if $\gamma = 0.01$ we need several tens of thousands of boosting iterations. This problem was fixed by AdaBoost which can take advantage of the iterations in which the error of the weak hypotheses is smaller than $1/2 - \gamma$ and gain a large computational efficiency from that. While this last point might seem subtle, it is the main reason that AdaBoost is very efficient on real-world problems and this is ultimately the reason that it has so far played the dominant role in the application of boosting to real-world problems.

The development of boosting algorithms is a result of continuous interaction of practical and theoretical considerations, which demonstrates the importance of the interaction between these two modes of research. Breiman's work and our response to it represent the most recent chapter of this interaction.

3 The bias/variance explanation

Breiman's analysis of bagging and boosting is based on a decomposition of the expected error of the combined classifier into a bias term and a variance term. There are several difficulties with the use of this analysis for boosting (more details are given in our paper with Bartlett and Lee [13]):

1. The bias-variance decomposition originates in the analysis of quadratic regression. Its application to classification problems is problematic, as reflected in the large number of suggested decompositions [8, 9, 14], in addition to the one given by Breiman in this paper. One unavoidable problem is that voting over several independently generated rules can sometimes increase, rather than decrease, the expected error.

2. Even in those cases where voting several independent classifiers is guaranteed to decrease the expected error, it is not always the case that voting over several bagged classifiers will do the same. The analysis of bootstrap estimation, which underlies bagging, has only asymptotic guarantees that might not hold for real-world sample sizes.

3. We have performed an analysis of the behavior of bagging and boosting on top of both Quinlan's C4.5 decision-tree algorithm [11] and an algorithm which we call "stumps" which generates the best rule which is a test on a single feature (i.e., a one-level decision tree or decision "stump.") We computed the bias and variance of these methods on some of the synthetic problems used by Breiman. We used the definitions of bias and variance of both Kong and Dietterich [9] and of Breiman. The results are summarized in Table 1. It seems clear that while bagging is mostly a

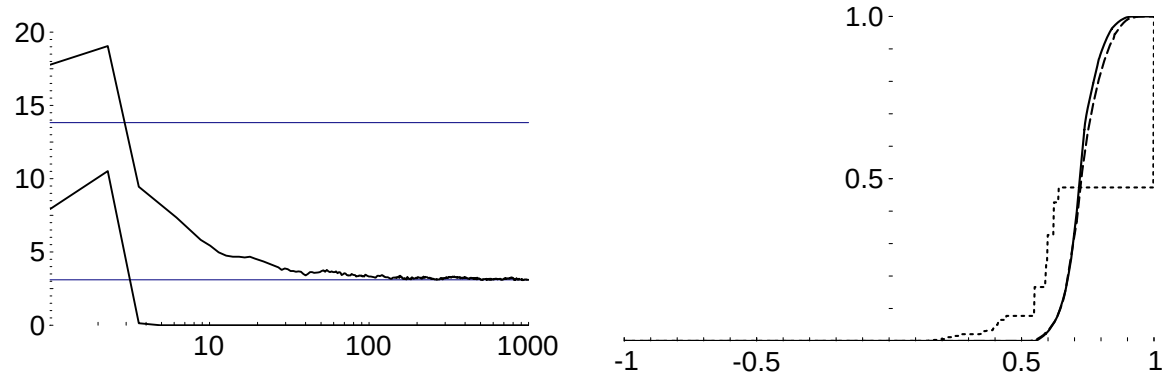

Figure 1: Error curves and the cumulative margin distribution graph for boosting C4.5 on the "letters" dataset.

variance reducing procedure, boosting can reduce both variance and bias. This is evident mostly in the experiments using stumps, which is a learning algorithm with very high bias. Moreover, in the experiment on the "ringnorm" data using boosting on stumps actually *increases* the variance, while at the same time it decreases the bias sufficiently to reduce the final error. These experiments demonstrate that variance-reduction cannot completely explain the performance of boosting.

4 The margins explanation

In our paper with Bartlett and Lee [13], we describe an alternative explanation for the fact that boosting can often decrease the test error even after the training error is zero. Here is a sketch of the explanation. Assume, for the sake of simplicity, that the problem is a *binary* classification problem and that the two possible labels are -1 and $+1$. Denote the input by x, the prediction of the i'th rule by $h_i(x)$ and the correct label by $y \in \{-1, +1\}$. Then the weighted vote rule generated by AdaBoost can be written as $\text{sign}(\sum_{i=1}^n \alpha_i h_i(x))$, where $\sum_{i=1}^n |\alpha_i| = 1$. The vote is correct on the example (x, y) if $y \sum_{i=1}^n \alpha_i h_i(x) > 0$. It is natural to think of $|\sum_{i=1}^n \alpha_i h_i(x)|$ as a measure of the *confidence* of the prediction. With this intuition in mind we define $m(x, y) \doteq y \sum_{i=1}^n \alpha_i h_i(x)$ to be the *margin* of (x, y). A large positive margin indicates a confident correct prediction, a large negative margin indicates a confident but incorrect prediction, and a small margin indicates unconfident predictions.

The claim of our explanation is that after boosting achieves zero training error, it goes on to generate a combined hypothesis whose margin is large on all of the examples in the training set and that it is this large margin that causes a decrease in the generalization error.

For example, in one experiment, we ran AdaBoost on top of C4.5 on the "letters" dataset, used also by Breiman in his paper. On the left of Figure 1, we have shown the training and test error curves (lower and upper curves, respectively) of the combined hypothesis as a function of the number of trees combined. The test error of C4.5 on this dataset (run just once) is 13.8%. The test error of boosting 1000 trees is 3.1%. (Both of these error rates are indicated in the figure as horizontal grid lines.) After just five trees have been combined, the training error of the combined hypothesis has already dropped to zero, but the test error continues to drop from 8.4% on round 5 down to 3.1% on round 1000.

As indicated above, our explanation for this phenomenon is based on the distribution of the margins of the training examples. We can visualize these margins by plotting their cumulative distribution (i.e., we can plot the fraction of examples whose margin is at most x as a function of $x \in [-1, 1]$). On the right side of Figure 1, we show the cumulative margin distributions that correspond to the experiment described above. The graphs show the margin distributions after 5, 100 and 1000 iterations, indicated

by short-dashed, long-dashed (mostly hidden) and solid curves, respectively.

Our main observation is that boosting tends to significantly increase the margins of the training examples, even after the training error reaches zero. In this case, although the training error remains unchanged (at zero) after round 5, the margin distribution changes quite significantly so that after 100 iterations all examples have a margin larger than 0.5. In comparison, on round 5, about 7.7% of the examples have margin below 0.5.

We present both experimental and theoretical evidence for the margin-based explanation for the effectiveness of boosting. Examining the margin distributions for a variety of problems and algorithms, we demonstrate a connection between the generalization error and the distribution of margins on the training set. We then back up these empirical observations with a theoretical explanation in two parts: First, we prove that, for sufficiently large training sets, there is a bound on the generalization error which is a function of the margin and which does *not* depend on the number of base hypotheses combined into the boosted hypothesis. Second, we prove that if the training errors of the base hypotheses are sufficiently small, then boosting is guaranteed to generate a combined hypothesis with large positive margins on all of the examples.

5 Some open problems

Breiman's work demonstrates the effectiveness of "perturb and combine" methods for reducing classification error. While a lot of understanding has been gained, many questions remain. Here are a few questions that seem particularly interesting to us:

- What is the relation between the randomized effect of bagging and the deterministic effect of boosting? Can the two effects be separated in experiments?
- Alternatively, is there a unified theory, based on provable theorems, which explains both boosting and bagging in a single framework?
- Can we characterize the learning algorithms and/or the data generation processes which are most likely to benefit from boosting or from bagging or from their combination.
- Is resampling the best way for randomly perturbing the training data? What about adding random noise to the label or to the features? How can we analyze these effects?
- Quinlan [10] has reported that, in unusual cases, boosting can actually *increase* the generalization error of the base learning algorithm by a small amount. Can we characterize or predict when boosting will fail in this manner?
- In Section 4, we discussed one explanation [13] for theoretically bounding the generalization error of voting methods like bagging and boosting. Are there other more practical and accurate methods for estimating the generalization error?

References

[1] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[2] Leo Breiman. The heuristics of instability in model selection. *Annals of Statistics*, 24:2350–2383, 1996.

[3] Harris Drucker and Corinna Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pages 479–485, 1996.

[4] Sally Floyd and Manfred Warmuth. Sample compression, learnability, and the Vapnik-Chervonenkis dimension. *Machine Learning*, 21(3):269–304, 1995.

[5] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[6] 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.

[7] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the Association for Computing Machinery*, 41(1):67–95, January 1994.

[8] Ron Kohavi and David H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 275–283, 1996.

[9] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 313–321, 1995.

[10] J. R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730, 1996.

[11] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[12] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[13] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machine Learning: Proceedings of the Fourteenth International Conference*, 1997.

[14] Robert Tibshirani. Bias, variance and prediction error for classification rules. Technical report, University of Toronto, November 1996.

[15] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[16] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.