LETTER **Communicated by Robert Jacobs**

Boosting Regression Estimators

Ran Avnimelech
Nathan Intrator
Department of Computer Science, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv, Israel

There is interest in extending the boosting algorithm (Schapire, 1990) to fit a wide range of regression problems. The threshold-based boosting algorithm for regression used an analogy between classification errors and big errors in regression. We focus on the practical aspects of this algorithm and compare it to other attempts to extend boosting to regression. The practical capabilities of this model are demonstrated on the laser data from the Santa Fe times-series competition and the Mackey-Glass time series, where the results surpass those of standard ensemble average.

1 Introduction

Boosting algorithms are ensemble learning algorithms that achieve improved performance by training different learners on different distributions of the data and combining their output (Schapire, 1990; Freund & Schapire, 1995). Boosting was found to be an effective method to achieve improved performance in many classification tasks. The success and increasing interest in ensemble methods for regression tasks encourage the application of boosting to regression tasks. There have been several different suggestions regarding the way in which this extension should be performed, each one considering a different analogy between classification errors and regression errors. We follow the version suggested by Freund (1995) and study the practical effects of the difference between classification and regression errors and the modifications that may lead to better performance in practice. We find that this version of boosting for regression can reduce the error rate when a small number of big errors contribute a significant part of the mean squared error (MSE). In addition to the theoretical analysis of the algorithm, we present empirical tests, including a case study of the behavior of the different predictors in one of the tests and how it contributes to error reduction.

Section 2 reviews the basic boosting algorithm and the AdaBoost algorithm and their applications. Section 3 reviews the usage of ensemble algorithms for regression tasks. Section 4 presents the algorithm and a theoretical analysis of it within an appropriate model. Section 5 reviews other attempts to extend boosting to regression and highlights the advantages and

disadvantages of each suggestion. We also claim that the algorithm may be effective in many tasks that are not fully compliant with the model, although it is not analytically shown. Empirical results are shown in section 6.

2 The Boosting Algorithm

2.1 The Original Boosting Algorithm. The original boosting algorithm (Schapire, 1990) was suggested in the context of the PAC learning model (Valiant, 1984). Theoretically, it enables achieving an arbitrarily low error rate, requiring the basic learners only to be able to achieve performance that is slightly better than random guessing on any input distribution. The algorithm trains the first learner on the original data set, and new learners are trained on data sets enriched with difficult patterns—patterns misclassified by some of the previous classifiers. There have been various improvements to the original boosting algorithm. Original boosting uses hierarchies of three-classifier ensembles; boosting by majority uses a simple ensemble that may consist of more classifiers, thus achieving the same improvement with less classifiers (Freund, 1995).

The boosting algorithm was successfully used in various real-world classification tasks, despite the fact that the assumptions of the PAC model do not hold in this more general context. Ensembles of neural networks, constructed by the boosting algorithm, significantly outperformed a single network or a simple ensemble on a digit recognition task (Drucker, Schapire, & Simard, 1993) and on a phoneme recognition task (Waterhouse & Cook, 1997).

In this article, we extend the boosting algorithm to regression problems by introducing the notion of weak and strong learning and an appropriate equivalence theorem between them. Practical implications are demonstrated on the laser data set.

2.2 AdaBoost. AdaBoost is an extension of boosting that applies to a more general context and takes into consideration the different error levels of the various classifiers (Freund & Schapire, 1995). The error rate determines the reweighting applied to the data when they are adaptively resampled for providing a training set for the next classifier. The error rate also determines the coefficient of each classifier when computing the ensemble output. This modification makes boosting effective when performance on the difficult sets is much worse than on the original set.

The reweighting procedure tries to construct decelerated classifiers by assigning weights to the new training set s.t. the (weighted) error rate of the previous learner on it would be 0.5. Initially the weights are uniform ($w_i^\mu = 1/N$) Let $c(\mu)$ be the true class of pattern μ (for a two-class task), h_t the hypothesis generated at step t, $d_t(\mu)$ the boolean that is equal to 1 when $h_t(\mu) \neq c(\mu)$, ϵ_t the (weighted) error rate of h_t, and $\beta_t = \epsilon_t/(1-\epsilon_t)$.

The weight updates used by AdaBoost are $w_{t+1}^{\mu} = w_t^{\mu} \cdot \beta_t^{d_t(\mu)} \cdot C_t$, where C_t is a normalization factor s.t. $\sum_{mu} w_{t+1}^{\mu} = 1$. The combination model is a weighted voting of the classifiers, with weights

$$\alpha_t = \ln\left(\frac{1}{\beta_t}\right) = \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right).$$

If the errors of different classifiers were independent (because of the stepwise deceleration), this would be the optimal Bayes decision.

Recently, several successful applications of AdaBoost have been reported (Breiman, 1996b; Schwenk & Bengio, 1997). Breiman applied AdaBoost to decision trees on various data sets and achieved improved performance (compared to bagging). Schwenk and Bengio applied AdaBoost to multilayer perceptrons (MLPs) and autoencoder-based classifiers ("diabolo networks") on character recognition tasks.

There have been several explanations as to why AdaBoost works. Breiman (1996b) suggested that AdaBoost and other algorithms that adaptively resample the training data and combine the classifiers (*arcing*) gain by increasing the variance component of the error and reducing the variance, which enables gaining more from the combination.

Schapire, Freund, Bartlett, and Lee (1997) studied the success of boosting low-variance algorithms and suggested an alternative explanation. They present examples in which the error rate on the test set keeps decreasing when new hypotheses are added to the ensemble, even after the training error reaches zero. They explain this apparent contradiction with standard learning curves and the Occam's razor principle by studying the behavior of a quantity they term *margin*. For a convex function (i.e., linear combination with coefficients summing to 1) of a set of classifiers, the margin is defined (for any pattern) as a weighted sum of +1 for correct classifications and -1 for errors. Ensembles gain by increasing the minimal value of the margin. AdaBoost is specifically designed to concentrate on the patterns with low margin, and this leads to its improved performance. This article also includes a statistical bound to the error rate on unseen data based on the minimal margin and the VC-dimension of the individual learners, which is lower than the bound when the margin is not considered. This explains why in some cases the error on test data kept decreasing even after training error reached 0. A quantity edge, which is an extension of margin, was suggested in Breiman (1997). AdaBoost and other arcing algorithms are optimization algorithms for minimizing some function of the edge.

3 Ensemble of Predictors in Regression Setup

Regression learning may exhibit complex behavior such as nonlinearities, chaotic behavior (especially in time-series prediction), local effects or nonstationarity, and high levels of noise. Ensemble averaging of predictors can improve the performance of single predictors and add to its robustness

under these circumstances. This occurs when the errors made by different predictors are independent, and thus the ensemble average reduces the variance portion of the error (Geman, Bienenstock, & Doursat, 1992). There are various ways to increase the independence of the errors. The simplest is to split the data into independent sets (Meir, 1995); however, such a reduction in the number of training patterns may degrade the results of each predictor too much. Another approach is to bootstrap several training sets with a small percentage of nonoverlapping patterns (Efron & Tibshirani, 1993) or training sets constructed by sampling with repetition (Breiman, 1996a). A recently proposed method increases independence between the predictors by adding large amounts of noise to the training patterns (Raviv & Intrator, 1996).

A different approach to ensemble averaging is the adaptive mixture of experts (Jacobs, Jordan, Nowlan, & Hinton, 1991). This method is a divide-and-conquer algorithm that cotrains a gating network for (soft) partitioning the input space and expert networks modeling the underlying function in each of these partitions.

In this article, we introduce a boosting algorithm for regression as a method for training an ensemble of predictors so as to optimize their collective performance. The algorithm is based on a fundamental observation that often the MSE of a predictor is significantly greater than the squared median of the error due to a small number of large errors. By reducing the number of large errors, we are able to reduce the MSE.

4 The Regressor-Boosting Algorithm

4.1 Model Definitions. We introduce a regression notion of weak learning taken from the PAC framework (Schapire, 1990). The essence of the regression problem is constructing a function $f(\mathbf{x})$ based on a training set $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_N}, y_N)$, for the purpose of approximating y at future observations of $\mathbf{x}$. Usually the goal is to approximate just $E[y|\mathbf{x}]$ and not the conditional distribution $P[y|\mathbf{x}]$, that is, to approximate the underlying function G: $y = G(x) + \varepsilon(x)$ with $\varepsilon(x)$ a zero-mean (for any x) noise. It is assumed that the samples in the training set were drawn from the unknown distribution $P(\mathbf{x}, y) = P(\mathbf{x}) \cdot P(y|\mathbf{x})$ and the goal is to approximate $E[y|\mathbf{x}]$.

The model we describe includes some simplifying assumptions, and the more realistic case is discussed at the end of this section. We assume no noise is present, $y = G(x)$, and this unknown function should be approximated . Another assumption is the existence of unlimited number of pairs $\langle x, G(x)\rangle$ drawn from the joint distribution $P(\mathbf{x}, y)$. Therefore, we ignore within this analysis the different performance on training data and unseen data. In this context, we regard various input distributions by applying selective filters to the input: $P'(\mathbf{x}) \equiv P(\text{`filter'}, \mathbf{x}) = P(\mathbf{x}) \cdot P(\text{`filter'}|\mathbf{x})$.

Given an unknown function G and a source of pairs $\langle x, G(x)\rangle$, the following definitions refer to learners—algorithms that approximate a function G

by a function f_D that depends on the distribution $D \equiv P_D(x)$:

> *γ-Weak learner.* A learner for which exists some $\alpha < 1/2$, such that it is capable, for any given distribution D and $\delta > 0$, of finding, with probability $(1-\delta)$,[1] a function f_D such that $\Pr_D[|f_D(x)-G(x)| > \gamma] < \alpha$.
>
> *γ-Strong learner.* A learner capable (for any given D, δ) of finding, with probability $(1-\delta)$, a function F_D such that $\Pr_D[|F_D(x)-G(x)| > \gamma] < \epsilon$ for any $\epsilon > 0$.
>
> *Big error (with reference to γ).* An error g.t. γ. The big error rate (BER) for h on D_0 is $\Pr_{D_0}[|h(x) - G(x)| > \gamma]$. When the term *big error* is mentioned not in the context of a specific γ, it applies to any γ for which the learning algorithm estimating the underlying function is a γ-weak learner.
>
> *γ-Weak (γ-Strong) learnability.* Whether any γ-weak (γ-strong) learner for G exists.

Some immediate results are obvious from these definitions:

- Any γ-strong learner is a γ-weak learner.
- For $\gamma_1 < \gamma_2$ any γ_1-weak (strong) learner is a γ_2-weak (strong) learner.
- A γ-weak learner can find, for any distribution D, a function f_D whose median error is smaller than γ (i.e., BER $< \frac{1}{2}$).

It is not clear from the definitions whether γ-weak learnability is equivalent to γ-strong learnability. The rest of this section will prove the equivalence of the two terms and emphasize the significance of this equivalence.

4.2 The Regressor-Boosting Theorem. The following theorem suggests that in problems for which a γ-weak learner exists, an arbitrarily low rate of big errors may be achieved. This can reduce the MSE.

Theorem. *Given an unknown function G, a source of pairs $\langle x, G(x)\rangle$, and γ: If a γ-weak learner is available, then a γ-strong learner may be constructed too. In other words, γ-weak learnability is equivalent to γ-strong learnability.*

The contribution of the big errors to the general MSE is a function not only of their percentage but also of their MSE. However, given ϵ—the BER of the ensemble—and the error distribution of the individual predictors, this contribution is bounded. *At most* they contribute as much as the ϵ highest errors of a simple predictor. Each such error is at most the median error of several functions generated by the weak learning algorithm at this input point. Similarly, the error distribution on the inputs, which originally were

[1] The probability limit is for a misrepresentative data set.

not big errors, is not only bounded by γ, but it also is not likely to get worse.

This theorem implies that if the MSE is dominated by a small number of relatively large errors (i.e., the average (RMS) error is much greater than median error), the RMS error can be reduced to values close to the median error. This occurs in many cases. (Figure 3 shows such an example of the error distribution taken from the empirical tests we performed.) Another example is gaussian error distribution, where 32% errors that are higher than the RMS error contribute about 80% of the MSE. The choice of γ is arbitrary (as long as there exists a γ-weak learner), and there is a trade-off between lower γ and lower ϵ. Therefore, a γ could be chosen that would achieve a greater error reduction.

4.3 Proof. The following constructive proof follows the proof for boosting in classification (Schapire, 1990). Instead of the majority vote of an ensemble used for classification tasks, the median of an ensemble is used in regression problems. We discuss several variants: `Boost1` is the variant suggested in Freund (1995). `Boost2`, which is also appropriate for the proof, is a variant more focused in reducing the MSE by considering the different kinds of errors. Another variant, BOOST3, which does not fit this proof, is focused on the MSE.

The essence of the algorithm is the construction of ensembles of three estimators and combining them to reduce the BER from α to $(3\alpha^2 - 2\alpha^3)$ or less. We present two versions of this step (BOOST1, in Figure 1, and BOOST2). Using hierarchies of such ensembles leads to an arbitrarily low BER.

> BOOST1: The first estimator in such an ensemble is trained on the original input distribution. Fifty percent of the data set used for training the second estimator are patterns on which the first estimator has a big error and 50% are those on which it has not (no change in the internal distribution of each of the two groups). The training set for the third estimator consists only of patterns on which exactly one of the previous estimators had a big error. The ensemble output is the median of the outputs of the different estimators. Figure 1 shows the different distributions of the training sets of the three predictors and how they are combined to achieve a low BER.
>
> BOOST2: Similar to BOOST1, but the training set of the third estimator contains patterns on which the previous estimators had big errors of different signs, in addition to those on which exactly one of them had a big error. This algorithm ensures an error rate even lower than $(3\alpha^2 - 2\alpha^3)$.

A summary description of the algorithm is presented below. It includes

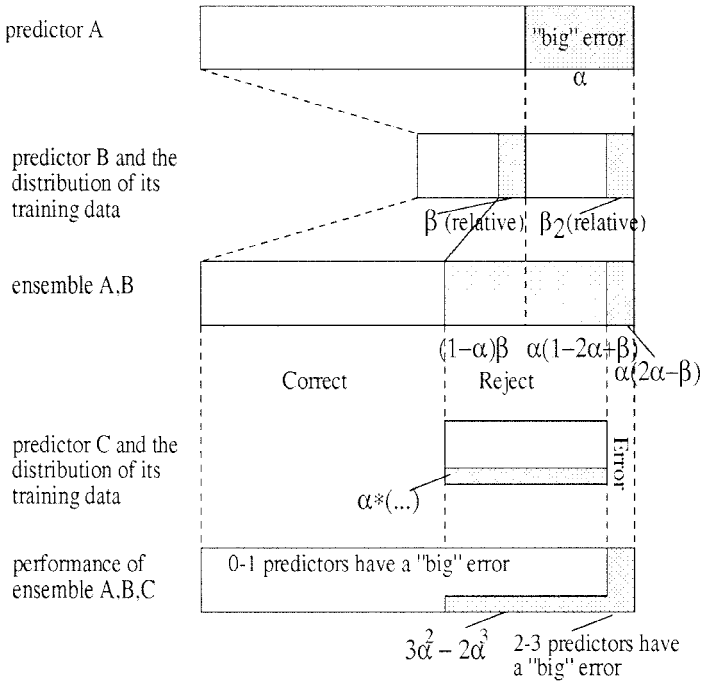

Figure 1: Main step of the boosting algorithm (BOOST1). Three predictors trained on different input distributions are combined. Training set of $\mathcal{B}$ contains the patterns on which $\mathcal{A}$ has a big error and a similar amount of patterns on which it does not have a big error. A and B may be considered an ensemble whose output pairs are evaluated according to what they imply on the median of them and another output: whether it would necessarily have a big error (*Error*) or it could not have a big error (*Correct*), or it depends on the value of the third output (*Reject*). The training set of $\mathcal{C}$ contains only *Rejects*—patterns on which $\mathcal{A}$ XOR $\mathcal{B}$ had a big error. This scheme shows that only on $(3\alpha^2 - 2\alpha^3)$ of the patterns 2-3 predictors have a big error. The median may have a big error only on these patterns. See proof for the mathematical details

BOOST3, a modification of BOOST2 described and justified later:

1. *Split the training set to three Sets. The first set should be smaller than the other two sets, because it is used as a whole for training.*
2. *Train the first expert on Training Set 1 = Set 1.*
3. *Assign to Training Set 2 all the patterns from Set 2 on which expert 1 has a big error and a similar number of patterns from this Set 2 on which it does not have a big error and train the next expert on it.*

4. *Assign all patterns from Set 3 on which the third expert may have a critical influence on the median to Training Set 3 for the third expert and train. The criterion for this set may differ by version:*

   Boost1 *Any pattern on which exactly one of the first experts has a big error.*

   Boost2 *Set (*BOOST1*) + Any pattern on which both experts have a big error, but these errors have different signs.*

   Boost3 *Set (*BOOST2*) + Any pattern on which both experts have a big error, but there is a "big" difference (see details in section 4.4) between the magnitude of the errors.*

5. *The ensemble output for any test pattern is the median of the outputs of the three experts.*

Proof. Given a test set and three weak learners trained as above, each with big error rate α on appropriate distribution, we show that in an ensemble of three such estimators, $(3\alpha^2 - 2\alpha^3)$ of the patterns are those on which two to three estimators have big errors. Therefore, the median has a big error rate of $(3\alpha^2 - 2\alpha^3)$ at most (the median may have a big error only if two to three estimators have such error, but if two estimators have big errors with different signs, the median would be the estimator with a small error).

BOOST1: We term the ratio of patterns on which the first two estimators have big errors as $Error_{\mathcal{AB}}$ and that of patterns on which one of them has a big error as $Reject_{\mathcal{AB}}$ (see Figure 1). $Error_{\mathcal{ABC}}$ is the number of patterns on which most of the estimators have big errors. The unknown BER of $\mathcal{B}$ on patterns for which $\mathcal{A}$ produces a γ-accurate prediction is marked as β and the BER of $\mathcal{B}$ on patterns on which $\mathcal{A}$ has a big error is marked as β_2. (It should be noted that as we randomly choose patterns from an infinite source, there are no two distinct groups of patterns on which $\mathcal{A}$ produces a γ-accurate prediction—those used and those not used for training of $\mathcal{B}$.)

The weak learning assumption states that $\mathcal{B}$'s BER on its training set distribution is at most α. Half of $\mathcal{B}$'s training set is patterns on which $\mathcal{A}$ had a big error. By applying the weak learning assumption on this distribution, we get: $\frac{1}{2} \cdot \beta + \frac{1}{2} \cdot \beta_2 \leq \alpha$. Assuming worse case (equality), $\mathcal{B}$'s BER on $\mathcal{A}$'s big errors is $\beta_2 = 2\alpha - \beta$, leading to:

$$
\begin{aligned}
Error_{\mathcal{AB}} &= \alpha \cdot (2\alpha - \beta) \\
Reject_{\mathcal{AB}} &= \beta \cdot (1-\alpha) + (1 - (2\alpha - \beta)) \cdot \alpha \\
Error_{\mathcal{ABC}} &= Error_{\mathcal{AB}} + \alpha \cdot Reject_{\mathcal{AB}} \\
&= \alpha \cdot (2\alpha - \beta) + \alpha \cdot [\beta \cdot (1-\alpha) + (1 + \beta - 2\alpha) \cdot \alpha] \\
&= 2\alpha^2 - \alpha\beta + \alpha\beta - \alpha^2\beta + \alpha^2 + \alpha^2\beta - 2\alpha^3 = 3\alpha^2 - 2\alpha^3
\end{aligned}
$$

BOOST2: $Error'_{\mathcal{AB}}$ is the percentage of patterns on which the median would have a big error regardless of the third estimator (i.e., both A and B

have big errors with common sign). $Reject'_{\mathcal{AB}}$ is the percentage of patterns for which the median would have a big error iff the third predictor had a big error. The unknown ratio of patterns on which $\mathcal{A}$ and $\mathcal{B}$ had big errors of different signs (relatively to the total number of common *big errors*) is marked ζ:

$$\begin{aligned} Error'_{\mathcal{AB}} &= \alpha \cdot (2\alpha - \beta) \cdot (1 - \zeta) \\ Reject'_{\mathcal{AB}} &= \beta \cdot (1 - \alpha) + (1 - (2\alpha - \beta)(1 - \zeta)) \cdot \alpha \\ Error'_{\mathcal{ABC}} &= Error'_{\mathcal{AB}} + \alpha \cdot Reject'_{\mathcal{AB}} \\ &= 3\alpha^2 - 2\alpha^3 - \alpha\zeta(2\alpha - \beta) + \alpha^2\zeta(2\alpha - \beta) \\ &= 3\alpha^2 - 2\alpha^3 - \zeta\alpha(1 - \alpha)(2\alpha - \beta) \end{aligned}$$

4.4 Practical Considerations and Limitations. The above model refers to a threshold γ for big errors. However, in regression problems, the goal is usually to reduce the MSE, and this presents a dilemma about the desired value of γ. Theoretically, the choice may be the lowest value for which we have a γ-weak learner. In practice there are several considerations:

- The size of the data set is finite.
- Only a limited number of estimators are combined.
- Boosting may be effective, although our learner is not a γ-weak learner (just like in classification).

The optimal γ is one for which the big errors are responsible for a significant part of the MSE, but the BER is low (usually the sets on which the second and third estimators are trained are more difficult and have a higher BER). In most cases, the choice of a good γ may require tuning.

The use of this boosting algorithm may also change the basic learner appropriate for the problem. It encourages the use of learners that are robust to the presence of outliers (e.g., minimize MSE$' \equiv$ MSE excluding x% greatest errors). In the training stage, the worse points are less crucial, as they will be learned by the other estimators. When the ensemble is used for prediction, the worst of the three estimates (for any sample point) is not relevant, as the median must be one of the other estimates.

In practical cases, noise will also be present. Its various effects on the algorithm are mentioned in Section 4.6, but the bottom line is that boosting may be effective when the errors of its basic estimators are large compared to the noise level.

One of the limits of boosting in regression is that while its focus on the real goal (MSE reduction) is limited, the basic learner already focuses on the harder patterns (e.g., the learning rule in backpropagation practically assigns a higher weight to patterns on which there is a big error). In iterative prediction, however, training a predictor optimized for stepwise prediction and using it iteratively is usually simpler and more effective than designing

a predictor specifically for iterative prediction. Yet it is simple to reweight the training set according to the performance of the iterative prediction, while training the individual predictors (using the weighted training sets) with a single-step prediction goal.

An extension of the principle that guided us to suggest BOOST2 is including some of the patterns on which the first two predictors have big errors (of same sign) in the training set of the third set. This may increase the BER of the median predictor, but if these are patterns, for which there is a big difference between the errors of the first two estimators, it is likely to reduce its MSE. The two extents are using the original BOOST2 or adding all such patterns to the set. A criterion for choosing some of the patterns should consider the effect of choosing between the two predictors (i.e., the difference in their squared errors) and the expected cost to performance on other patterns of including it in the training set (i.e., the pattern's difficulty). A simple criterion we use is including those patterns on which the difference between the predictions is g.t. the threshold γ by which we defined big errors.

4.5 Extensions of the Algorithm.

4.5.1 AdaBoost. Applying AdaBoost to regression tasks is done in a similar way to the way in which boosting was applied to regression. The description of the actual algorithm is as follows:

1. Initialize uniform weights to patterns: $w_1^\mu = 1/N$.
2. Train single predictor according to current weights.
3. Compute error ratio ϵ_t according to pattern errors ϵ_t^μ: $\epsilon_t = \sum_{\mu:\epsilon_t^\mu > \gamma} w_t^\mu$.
4. If $\epsilon_t > 0.5$ update weights: If pattern had a big error: $w_{t+1}^\mu = w_t^\mu \cdot \frac{0.5}{\epsilon_t}$, otherwise: $w_{t+1}^\mu = w_t^\mu \cdot \frac{0.5}{1-\epsilon_t}$.
5. If halting condition has not been reached, return to step 2; halting condition is either maximal number of hypotheses (predictors) or maximal number of consecutive failures (BER > 0.5).

The final prediction is a weighted median of the predictors with coefficients $\alpha_t = \ln \frac{1-\epsilon_t}{\epsilon_t}$.

In practice, AdaBoost may be less effective due to several reasons. In many cases, the first regressor is significantly better than other regressors (because it is evaluated on the original distribution) and $\alpha_1 > \sum_{t=2}^{N} \alpha_t$. Therefore, the median is the first estimator. Another practical limit is the fact that the first estimator is already a compromise between better local estimations and the fact that in regression, the bigger errors already have a big effect due to minimizing the squared error (unlike classification).

Because we are interested in reducing the MSE rather than the BER, a nonweighted combination may result in a performance similar to or better than that achieved by a weighted combination. The coefficients in the weighted combination model are determined according to the BER of each predictor, which is just a partial indicator of the quality of the predictor. Similarly, combining the predictors through the mean is likely to be as good as using the median.

4.5.2 Iterative Approach: Several Error Thresholds. A disadvantage of the algorithm presented here is that it is effective only in reducing the errors to be below some threshold (which cannot be lower than the median error). A solution is using an iterative approach. Once the errors of the vast majority of the patterns are lower than the threshold, a lower threshold can be chosen, and small ensembles may be trained on the training set reweighted according to this threshold.

Assuming the error sizes are uniformly distributed beneath the old threshold (a pessimistic assumption), a new threshold that is 0.7 the old one (half the squared error) will still have a sufficiently small BER.

4.6 Boosting and Noisy Estimation. The description of the model ignored noise. The presence of noise poses the problem of overfitting the data—learning the noise rather than just the underlying function. There are cases in which the use of boosting may increase the sensitivity to noise. Therefore, boosting will be effective when the error level of the basic estimators is higher than the noise level. Noise will not only interrupt the learning process of each predictor, an effect emphasized by the split and "waste" of the data, but also mark good estimates as big errors, thus overemphasizing them.

A complete analysis of the behavior of boosting in the presence of noise depends on the error distribution of the basic learner, the noise distribution, the exact effect the reweighting has on the basic estimators, and other factors. However, if we have some estimate of the noise level, there may be some rules of thumb that may indicate when noise may disturb boosting. The reweighting should not emphasize the noise; the noise should be low compared with the error level of the single estimator. It should also be low relative to the threshold γ; noise higher than γ or close to it (which may mark small errors as big) should be rare. When a series of estimators is used, the weights of patterns that had big errors by most or all the estimators should be bounded according to the probability of noise $> \gamma$. This limits the number of estimators constructed and combined by the algorithm. This limit may also be encountered if the threshold is too high compared to the error level of a single estimator (because that will cause massive weight updates). These limits may become relevant not only as a result of noise, but also due to the capacity of the estimators. Another effect that should be noticed in the presence of noise is the relation between the BER and the MSE. If γ is much

higher than the average error, estimators trained on the reweighted set may have an MSE that is higher than or similar to that of previous estimators but a BER that is much lower. This may lead to giving them unjustified high coefficients in the weighted median used by AdaBoost. Such estimators may overfit a few points with high noise.

The combination model of boosting has better immunity to noise than the generation of new estimators. One advantage is that the (weighted) median is at least as smooth as the single estimators. If there exist some metric D and some function G s.t. for every estimator $|f_i(x) - f_i(y)| < G(D_{xy})$ then the same smoothness condition is also true for the median. Furthermore, the estimators that suffer from overfitting usually have a BER closer to 0.5 and will have lower coefficients.

When the noise level varies (or the confidence level of the estimator), it may be useful to estimate it locally, and let that influence the flexible threshold $\gamma(x)$. Using such a threshold may focus more on errors of the estimator rather than noise.

5 Other Attempts to Extend Boosting to Regression

There have been several other attempts to extend boosting to regression tasks, one of which was also empirically tested (Freund & Schapire, 1995; Drucker, 1997).

5.1 Function Estimation as a Set of Boolean Queries. Freund and Schapire (1995) suggest an approach that considers a function estimation task as an infinite series of boolean queries: $c(x_i, y) = (y_i > y)$. A squared error cost is achieved by the distribution of queries: $P(y|x_i) \propto |y - y_i|$ (the possible values of y are bounded). Applying AdaBoost directly changes the distribution of pairs; the conditional distribution $P(y|x)$ changes as well as $P(x)$.

This method has several advantages. It attempts to reduce the MSE directly, rather than the number of big errors. It attempts to reduce the error to zero (but stops when the hypothesis error exceeds 0.5). `AdaBoost.R` also favors errors whose sign is opposite to the sign of previous errors. The main disadvantage of this method is related to its implementation: The weak learner has to perform a task that is more complex than minimizing the weighted MSE. It may also be inappropriate for learning algorithms that rely on the gradient of the error function (e.g., backpropagation). The gradient increases only in the range between the target value and the current prediction. Thus, the patterns that will have a small error at an early stage of the training of an additional predictor will be emphasized. Another disadvantage is the massive weight changes when the errors are small compared with the y bounds. The initial massive weight changes also cause the coefficient of the first regressor in the combination model to be much greater than any other coefficient (even if error distribution is similar). Thus, in small en-

sembles, the median is actually the first predictor. This method also contains a hidden hyperparameter: extending the y bounds would change the percentage of errors (by adding many easy queries), thus severely changing the reweighting factors.

5.2 Boosting for Regression Using a Continuous Loss Function. Drucker (1997) suggests a different approach. A loss function L_μ is assigned to each pattern (for each estimator) that is a function (e.g., identity, square, exponent) of the ratio between its error and the maximal error (in the range $[0, 1]$). The error rate is the (weighted) average loss, $\beta_t = (1 - \bar{L})/\bar{L}$. The weight updates are $w^\mu_{t+1} = w^\mu_t \cdot \beta^{L_\mu} \cdot C_t$. The algorithm terminates when $\bar{L}$ exceeds 0.5. Drucker reports that the algorithm outperforms bagging on a set of benchmark synthetic tasks and on the Boston housing data set. Drucker's method is characterized by the use of a continuous loss function for the reweighting of the patterns. The main advantages of this method are that it concentrates on the big errors with no need to adjust a parameter to the typical error, and it does not increase the complexity of the task presented to the basic learner. Its main disadvantage is the dependence of the loss function on the maximal error. This means that two estimators with the same error distribution relative to the maximal error—but with the maximal error of one double that of the other (i.e., $\forall x : P(\epsilon_2 = x) = P(\epsilon_1 = 2x)$)—are considered to have similar performance. This also leads to big changes in the weighting as a single extreme value varies. If the algorithm is used just for reweighting the patterns, while the combination is through a nonweighted median / mean (which may be appropriate, as previously suggested), this disadvantage becomes less significant.

5.3 Strengths and Drawbacks of Threshold-Based Boosting. The main disadvantages of the method used in this work and in Freund (1995) are the fact that it is limited to reducing the errors to the chosen threshold and not down to zero and the need to choose this threshold. The significance of the first limit varies with the error distribution. If further error reduction is required, the method may be applied recursively, using a different threshold at each level. The need to choose the threshold for big errors may actually be an advantage at nontrivial tasks by providing flexibility and suggesting a choice of threshold based on the error distribution. (For simple tasks, a threshold slightly higher than the RMS error should be fine.) Another advantage of this algorithm is the simplicity of its implementation.

A summary of such a comparison cannot state that one of the variants is always superior to another. The performance of each method relies on the specifics of the problem and the basic learner used. The mere fact that a method can theoretically reduce the error to zero does not imply such results in practice. `AdaBoost.R` is most appealing theoretically, but may

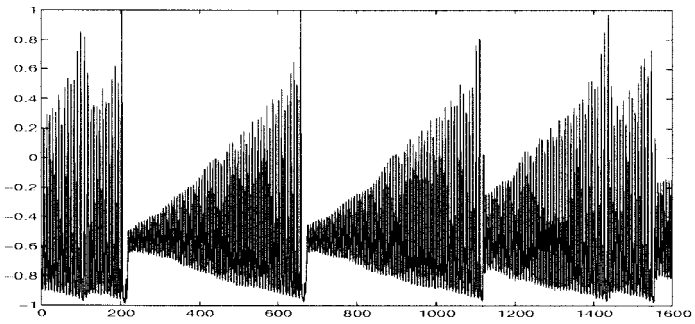

Figure 2: Typical segment of laser-intensity time series.

suffer severely from practical drawbacks, such as its massive weight updates and its error gradient. Drucker's variant may suffer from the fact that its coefficients for combining the predictors are not directly related to the performance of each predictor (if a weighted combination is used). The approach we followed may be limited theoretically and in its maximal contribution but has two advantages for the layman. One advantage is that it is simple to implement and does not affect the basic learner. The other advantage is that by observing the error distribution and choosing the threshold, one may be aware in advance of the effect this version of boosting may have on the overall performance.

6 Results

6.1 Boosting on Laser Data. We demonstrate the capabilities of the boosting algorithm on laser data from the Santa Fe times-series competition[2] (data set A) (Weigend & Gershenfeld, 1993). This time series is the intensity of a NH_3-FIR laser, which exhibits Lorenz-like chaos (see Figure 2). It has a sampling noise due to the A/D conversion to 256 discrete values. The behavior of the time series may be described as having "normal" behavior and several types of "collapses." Many models may adequately fit the "normal" behavior while failing to learn the "catastrophic" behavior (see Figure 3). The comparison of the performance is followed by a detailed analysis of the behavior of each of the estimators and the resulting median estimator, which may provide a greater sense of how this algorithm actually works.

We compared the performance of standard "bagging" ensembles and boosted ensembles, all consisting of three networks. The basic learners were

[2] http://www.cs.colorado.edu/ andreas/Time-Series/SantaFe.html.

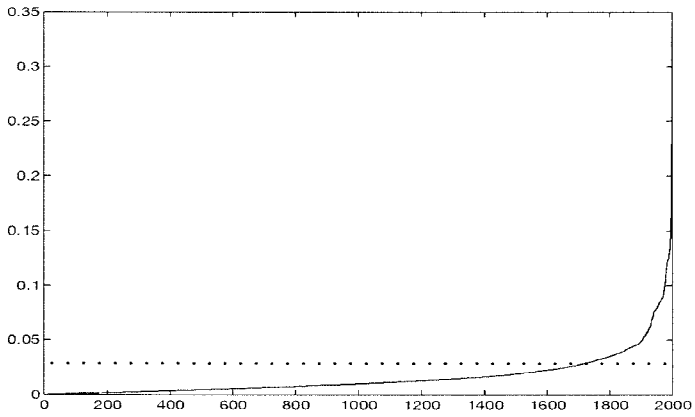

Figure 3: Error distribution of neural net predictor on test set. Dotted lines mark the RMS error.

Table 1: Normalized MSE X10^{-3} of Different Types of Three-Predictor Ensembles on the Laser Data.

Ensemble Type	Bagging (3)	Simple Boosting			AdaBoost-3	
		BOOST1	BOOST2	BOOST3	Nonweighted	Weighted
Average Predictor	2.8 (0.3)	2.6 (0.2)	2.5 (0.2)	2.6 (0.2)	2.7 (0.3)	2.7 (0.4)
Median Predictor	3.1 (0.3)	2.7 (0.1)	2.6 (0.2)	2.7 (0.2)	3.0 (0.3)	3.1 (0.3)

two-layer feedforward neural networks predicting the next value according to the 16 previous values that were used as the net input. The hidden layer consisted of six units. The results presented were collected using the first 8000 points (of the combined set of original training and continuation data) as the training set and the following 2000 points as test data.

Table 1 compares the performance of ensembles implementing bagging, the different variants of simple boosting, and AdaBoost. For each method we presented two results: the performance of the ensemble with an average-based combination model and its performance using a median-based combination model.

The results are presented in normalized mean squared errors (NMSE). NMSE is the MSE divided by the variance across the data. (When scaled to the range $[-1, 1]$ the laser data had mean -0.5326 and S.D. 0.3676.) The performance achieved by our ensembles is significantly better than that re-

ported in Nix and Weigend (1995) and the better-performing participants in the Santa Fe time-series competition (Weigend & Gershenfeld, 1993): Nix and Weigend report NMSE $= 0.0139$ for a single predictor (they used just 1000 points—unsampled them with an FFT method with factor 32—for training.) The results presented in the competition are mostly of iterated prediction. According to the partial results presented for single-step prediction, the overall NMSE of the best method is about 0.01. The NMSE of the other competitors is significantly higher.

6.1.1 Analysis of the Behavior of the Three Estimators. Figure 4 shows how the different estimators behave on different patterns in one of the tests we performed. The estimators colored red, green, and blue, respectively, were analyzed on two groups of patterns: The easy patterns are the 50% of the data on which both the first and second estimator had an error smaller than 0.02 (this is lower than the threshold γ used in this test). The difficult patterns are the 3% patterns on which at least one of the first two patterns had an error g.t. 0.1 (this is higher than the γ used).[3]

6.1.2 Errors on Easy Patterns. The first estimator was trained on a data set that represented the original distribution and has very accurate predictions on these patterns (MSE $\simeq 0.6 \cdot 10^{-4}$, NMSE $=$ MSE $\cdot$ 7.4). The second estimator was trained on a set in which these patterns were underrepresented and had accurate predictions (MSE $\simeq 1.2 \cdot 10^{-4}$). The third estimator was trained only on more difficult patterns, so its error level on these patterns (MSE $\simeq 9 \cdot 10^{-4}$) is higher.

The ensemble output is the median, so it is hardly influenced by the worst result and has an MSE slightly lower than that of the first estimator. Another effect, which is evident in these graphs, is the lack of correlation in size and sign between errors of the different estimators on the patterns in this group.

6.1.3 Errors on Difficult Patterns. The first estimator (MSE $\simeq 0.025$ on these patterns) and the second estimator (MSE $\simeq 0.03$) were influenced mainly from the "normal" patterns and err on these patterns. Because the errors of both estimators on these patterns are quite decorrelated (compared to independently trained estimators), a better estimator can influence the median. The figure also shows that some of the patterns that were estimated accurately by the first estimator were underrepresented in the train set of the second estimator, and it had big errors on them.

[3] The criteria for the two groups create some of the asymmetry between the estimators because they filter according to a condition on the error of the first two estimators and not the third one, but this effect is significantly smaller than the actual asymmetry revealed in the graphs.

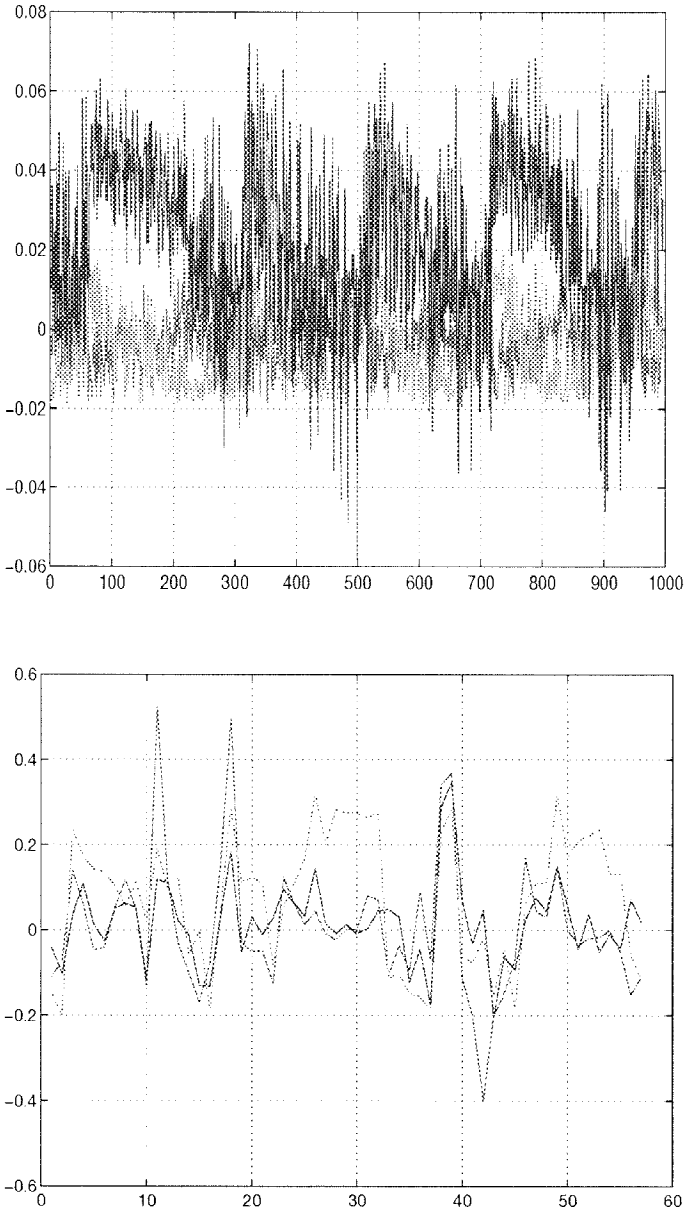

Figure 4: Errors of the three estimators on the easy patterns (top) and the difficult patterns.

The third estimator was trained only on "difficult" patterns where its output may have a great impact on the ensemble output. Therefore, it has a better performance on such patterns (MSE $\simeq$ 0.01). Its error on these patterns is almost always similar to the better estimator of the previous two, or better.

The ensemble output—the median—has MSE $\simeq$ 0.012 on these patterns, due to the relatively good performance of the third estimator and the relatively weak correlation between errors of the first two estimators.

6.2 Iterative Time-Series Prediction. We performed a further set of tests on the laser data. In these tests, the goal was to predict the next 16 values. The error measure was the sum of the squared errors on all 16 predicted values. The predictors used were the same 16-input, 6-hidden neural networks used previously. These networks were trained according to the single-step prediction goal. However, the reweighting used by the boosting algorithm was based on the performance of the iterative prediction.

We compare the performance of ensembles constructed by the several variants of the boosting algorithm to those based on bagging and to single networks. For each ensemble, there are four possible outputs: The combination may be performed either at each step or on the final predictions, and it may use mean or median. The NMSE of the individual networks was 0.0112 ± 0.022. (The first networks in the boosting esembles were slightly better: $.0095 \pm 0.0037$.)

Table 2 compares bagging ensembles, AdaBoost ensembles, and two variants of simple Boosting (BOOST2 is meaningless in this context). The threshold used for boosting was NMSE of 0.22; typically 5% to 10% of the patterns have an error beyond this threshold, and their contribution to the total MSE is 85% to 90%. There was no significant difference between the performance of bagging as the size of the ensemble changed from two to six. We present just these two ensembles. The ensemble mean in both variants of simple boosting is typically with NMSE of order of 1 and is not presented in Table 2. The combination model uses uniform weights unless stated otherwise. For the five-predictor AdaBoost ensembles, we present two weighted combinations: using the weights specified in AdaBoost, or these weights if they are positive and 0 otherwise.

These results demonstrate the advantage of boosting over standard ensembles in scenarios in which its explicit reweighting of the patterns differs from the implicit one of the individual predictors. Boosting becomes more effective than bagging, even when only two predictors are combined. Using five predictors within an ensemble, a 50% error reduction is achieved. These results also support the MSE-oriented modification we introduced to simple boosting. (The advantage of BOOST3 is greater than the standard deviation in the table implies, as it was rarely inferior to BOOST1 using the same first two predictors.) The advantage of the overall combination model over the stepwise combination may be attributed to the reweighting, although

Table 2: Normalized MSE X10^{-2} of Different Types of Ensembles on the Iterated Prediction Task.

Combination Model Combination Model	Step-wise Mean	Step-wise Median	Overall Mean	Overall Median
Bagging—2 nets	8.2 (1.0)		7.8 (1.2)	
Boosting—2 nets	6.9 (1.1)		6.3 (1.3)	
Bagging—6 nets	8.5 (0.8)	9.0 (0.5)	**7.4** (0.3)	7.9 (0.7)
AdaBoost—3 nets	5.5 (0.8)	5.6 (1.0)	4.8 (0.6)	5.0 (1.2)
BOOST1	—	6.7 (1.4)	—	6.4 (0.8)
BOOST3	—	5.8 (1.4)	—	5.4 (1.0)
AdaBoost—4 nets	5.3 (1.4)	5.0 (1.4)	4.5 (1.0)	4.3 (1.1)
AdaBoost—5 nets	5.0 (1.1)	4.8 (1.3)	**4.2** (0.8)	4.3 (1.0)
AdaBoost—5 nets weighted averages	6.6 (1.6)	9.3 (3.9)	5.5 (1.6)	9.4 (3.7)
AdaBoost—5 nets positive weights	6.5 (1.6)	9.5 (3.7)	5.4 (1.6)	9.5 (3.7)

a smaller but similar effect exists in bagging too. Median is the combination model for simple boosting, but in AdaBoost it has no advantage over mean (in bagging, mean is better). Our tests also found simple averaging (mean or median) to outperform the weighted versions. This is due to the large coefficients of the earlier predictors (especially the first one), while the performance of the different predictors is similar (actually, for the first few predictors, there is a gradual improvement). The median was usually the first predictor. The mean was less affected. It clearly outperforms bagging but is inferior to the nonweighted mean.

6.3 Mackey-Glass Time Series. The Mackey-Glass differential-delay equation (Mackey & Glass, 1977),

$$\frac{dx(t)}{d(t)} = -bx(t) + a\frac{x(t-\tau}{1+x(t-\tau)^10} \tag{6.1}$$

and the time series resulting from its integration have attracted much focus as a statistical learning benchmark (Moody, 1989; Crowder, 1990). We performed tests on a data set of this time series in the CMU repository.[4] This specific time series was generated with $\tau = 17$, $a = 0.2$, and $b = 0.1$. The input data used are $x(t-18)$, $x(t-12)$, $x(t-6)$, $x(t)$, and the task is to predict $x(t+6)$. Training data consist of 3000 data points, and the test set consists of 500 data points, starting 1800 time steps after the end of the training data.

We compared the performance of ensembles constructed by the AdaBoost algorithm to those constructed by bagging. The basic learner we

[4] http://www.boltz.cs.cmu.edu/benchmarks/mackey-glass.html

Table 3: Normalized RMS $X10^{-2}$ of different types of ensembles on Mackey-Glass data

Ensemble-Type	Bagging		AdaBoost			
Combination Method	Median	Mean	Weighted Median	Weighted Mean	Nonweighted Median	Nonweighted Mean
3 predictors	1.30 (0.07)	1.25 (0.04)	1.28 (0.04)	1.13 (0.03)	1.20 (0.04)	1.13 (0.02)
4 predictors	1.24 (0.03)	1.23 (0.03)	1.22 (0.05)	1.10 (0.01)	1.12 (0.02)	1.10 (0.01)
5 predictors	1.24 (0.04)	1.22 (0.04)	1.21 (0.05)	1.08 (0.02)	1.09 (0.02)	1.07 (0.01)

used was a two-layer neural network with 20 hidden units. Such a learner achieved normalized RMS of 0.014, which compares favorably with other results achieved on these data. Table 3 shows the normalized RMS of ensembles constructed by bagging and by AdaBoost, using either the average or the median as the ensemble output. For AdaBoost we present the performance of both the weighted and nonweighted median / average. The normalized RMS of simple boosting (three predictors) was 0.0124 ± 0.0004 using the median as the output and 0.0148 ± 0.0012 using the mean. (This result is inferior to AdaBoost and similar to Bagging.)

These results show the advantage of AdaBoost over bagging. They also demonstrate that the nonweighted averages may be at least as good as the weighted averages and that the mean may be as least as good as the median. The relatively good performance of the nonweighted versions is due to the limit of the BER in indicating the quality. The BER provides some measure of the relative performance of each estimator, but because these coefficients ignore the finer details of the error distribution, they lack an advantage over a simple average. The relatively poor performance of the weighted mean results from the fact that in some cases, it is exactly the first predictor, due to its higher coefficient. The dichotomy between these cases and the cases in which all the predictors are used also leads to the higher variance in its performance.

7 Discussion

This work reviews the extension of the boosting algorithm to fit regression problems and focuses on a threshold-based application of boosting for regression. This method is designed to fit tasks in which poor performance is due to the effect of difficult patterns, unfitted by the model. Various tasks, for which large data sets are available, exhibit this behavior and may gain from this new procedure. The basic principle of this method is regarding "big" estimation errors as "classification-like" errors and implementation of a mechanism to reduce their amount.

We focus on the practical aspects of boosting in regression. The model for extending the algorithm is based on an analogy of regression and classification errors. While this leads to a possible algorithm that reduces the number of errors beyond a given threshold and consequently reduces the MSE, certain minor modifications may be more appropriate for reducing the MSE. We also present a comparison between this method and other versions of boosting in regression (Freund & Schapire, 1995; Drucker, 1997). Although this comparison does not lead to a conclusive choice of one of these versions as superior to others, it emphasizes the advantages and drawbacks of each method. Other methods may be more appealing theoretically, but they seem to have practical drawbacks.

The results achieved on the laser data and the Mackey-Glass time series demonstrate the potential of decorrelating the errors of different predictors, using threshold-based boosting, and combining them in a robust manner. Nonweighted averages and the mean of the predictors, rather than the median (in AdaBoost), may in many cases perform as least as good as the weighted median specified in the theoretical model. This is due to the fact that the weights are based on the BER rather than the MSE.

The tests performed on an iterative prediction task emphasize the advantage of using boosting when the goal presented to the individual predictors does not fully represent the real goal. In such cases, the boosting mechanism may contribute to the learning, something that is not handled by the individual predictor, thus being more effective.

Our analysis of the behavior of different estimators on different kinds of patterns (for simple boosting) provides insight into the way error reduction is achieved. The performance of the third estimator on easy patterns is not as good as of the first two, but it has almost no influence on the median. On the difficult patterns, however, it has a lower error rate; thus, the median will usually have either the smaller of the two errors or some intermediate value when these errors have different signs.

References

Breiman, L. (1996a). Bagging predictors. *Machine Learning, 24* (TR-421), 123–140.

Breiman, L. (1996b). *Bias, variance and arcing classifiers* (Tech. Rep. TR-460). Berkeley: Department of Statistics, University of California, Berkeley.

Breiman, L. (1997). *Arcing the edge* (Tech. Rep. TR-486). Berkeley: Department of Statistics, University of California, Berkeley.

Crowder, S. (1990). Predicting the Mackey-Glass timeseries with cascade correlation learning. In *Connectionist Models: Proceedings of the 1990 Summer School*.

Drucker, H. (1997). Improving regressors using boosting techniques. In *14th International Conference on Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Drucker, H., Schapire, R., & Simard, P. (1993). Improving performance in neural networks using a boosting algorithm. In S. J. Hanson, J. D. Cowan, &

C. L. Giles (Eds.), *Advances in neural information processing systems, 5* (pp. 42–49). San Mateo, CA: Morgan Kaufmann.

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap*. New York: Chapman & Hall.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation, 121*(2), 256–285.

Freund, Y., & Schapire, R. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *2nd European Conference on Computational Learning Theory*.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias-variance dilemma. *Neural Computation, 4*, 1–58.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*(1), 79–87.

Mackey, M., & Glass, L. (1977). Oscilations and chaos in physiological control systems. *Science*, (197).

Meir, R. (1995). Bias, variance and the combination of least square estimators. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 295–302). Cambridge, MA: MIT Press.

Moody, J. (1989). Fast learning in multi-resolution hierarchies. In *Advances in neural information processing systems, 1* (pp. 29–39). San Mateo, CA: Morgan Kaufmann.

Nix, D. A., & Wiegend, A. S. (1995). Learning local error bars for nonlinear regression. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 489–496). Cambridge, MA: MIT Press.

Raviv, Y., & Intrator, N. (1996). Bootstrapping with noise: An effective regularization technique. *Connection Science, 8*(3/4), 355–372.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning, 5*(2), 197–227.

Schapire, R., Freund, Y., Bartlett, P., & Lee, W. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. In *Machines That Learn—Snowbird*.

Schwenk, H., & Bengio, Y. (1997). *Adaptive boosting of neural networks for character recognition* (Tech. Rep. TR-1072). Montreal: Department d'Informatique et Recerche Operationnelle, Université d'Montreal.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134–1142.

Waterhouse, S. R., & Cook, G. (1997). Ensemble methods for phoneme classification. In M. Mozer, M. Jordan, & T. Petsche (Eds.), *Advances in neural information processing systems, 9*. Cambridge, MA: MIT Press.

Weigend, A. S., & Gershenfeld, N. A. (Eds.). (1993). *Time series prediction: Forecasting the future and understanding the past*. Reading, MA: Addison-Wesley.

Received June 5, 1997; accepted April 20, 1998.