Machine Learning, 35, 41–55 (1999)
© 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

An Efficient Method To Estimate Bagging's Generalization Error

DAVID H. WOLPERT dhw@ptolemy.arc.nasa.gov
NASA Ames Research Center, Caelum Research, MS 269-1, Moffett Field, CA 94035

WILLIAM G. MACREADY wgm@biosgroup.com
Bios Group, LP, 317 Paseo de Peralta, Santa Fe, NM 87501

Editor: Pazzani

Abstract. Bagging (Breiman, 1994a) is a technique that tries to improve a learning algorithm's performance by using bootstrap replicates of the training set (Efron & Tibshirani, 1993 , Efron, 1979). The computational requirements for estimating the resultant generalization error on a test set by means of cross-validation are often prohibitive, for leave-one-out cross-validation one needs to train the underlying algorithm on the order of $m\nu$ times, where m is the size of the training set and ν is the number of replicates. This paper presents several techniques for estimating the generalization error of a bagged learning algorithm without invoking yet more training of the underlying learning algorithm (beyond that of the bagging itself), as is required by cross-validation-based estimation. These techniques all exploit the bias-variance decomposition (Geman, Bienenstock & Doursat, 1992, Wolpert, 1996). The best of our estimators also exploits stacking (Wolpert, 1992). In a set of experiments reported here, it was found to be more accurate than both the alternative cross-validation-based estimator of the bagged algorithm's error and the cross-validation-based estimator of the underlying algorithm's error. This improvement was particularly pronounced for small test sets. This suggests a novel justification for using bagging— more accurate estimation of the generalization error than is possible without bagging.

Keywords: Bagging, cross-validation, stacking, generalization error, bootstrap

1. Introduction

This paper is concerned with the problem of supervised learning, in which one is provided a training set of input-output pairs sampled from a "target" input-output distribution, and uses a "learning algorithm" to convert that training set into an estimate of the target. In the technique of "bagging" (Bootstrap Aggregating (Breiman, 1994a)) one tries to improve this inference of the target, by running one's learning algorithm many times and averaging the results. Bagging often works quite well. Unfortunately though, the computational requirements for estimating its generalization error by means of cross-validation are often prohibitive, due to the many rerunnings of the underlying learning algorithm.

In this paper we investigate fast alternatives to traditional cross-validation for estimating bagging's generalization error. The best-performing of those alternatives, which exploits the idea of "stacking" (Wolpert, 1992), is not only far faster than cross-validation; it also results in more accurate estimation of the bagged algorithm's generalization error than does cross-validation. The estimate is even more accurate than is cross-validation run on the original, unbagged algorithm. This suggests a novel justification for using bagging— more accurate estimation of the generalization error than is possible without bagging.

Our notation is as follows: Let X be an input space and Y an output space. Let d be a training set of m input-output pairs, $\{x^i, y^i\}$, formed by sampling a "target" input-output distribution $P(y \in Y|x \in X)$, indicated by P_f, or just f for short. Unless explicitly noted otherwise, "$y \in Y$" will always refer to an output value formed by sampling the target.

Let h indicate a hypothesis input-output distribution $P(y \in Y|x \in X)$, generated by a learning algorithm G in response to a training set. All hypotheses considered in this paper (e.g., in the experiments) will be single-valued functions from X to Y (*i.e.*, the hypothesis probability distributions will be x-parameterized delta functions over Y), though the ideas naturally extend to non-single-valued hypotheses. Similarly, we will only explicitly discuss deterministic learning algorithms that always guess the same h in response to the same d, although the ideas extend to algorithms with a stochastic component. When the training set must be indicated but G can be assumed, we will write h_d.

In bagging, rather than use the hypothesis G creates after training on d, one uses the average of the hypotheses G creates by training on the ν sets d'_i. The d'_i are created in some manner from d. In Breiman's work the d'_i are "bootstrap replicates" of d, *i.e.*, each d'_i is created by sampling the pairs in d uniformly with replacement m times (Efron & Tibshirani, 1993 , Efron, 1979). In general, one would expect that bagging improves generalization for "unstable" learning algorithms, that is for algorithms for which h_d is a sensitive function of d. Bagging does not usually help "stable" learning algorithms (Breiman, 1994b).

When used directly to generalize from d, we will refer to G as the "underlying algorithm". When instead one averages the hypotheses G creates from the d'_i, we will refer to the resultant mapping from d to h_d as the "bagged algorithm".

Clearly the use of leave-one-out cross-validation to estimate the generalization error of the bagged algorithm can be prohibitive, since one needs to train the underlying algorithm $m\nu$ times. Even for J-fold cross-validation, you have to train the underlying algorithm on the order of $J\nu$ times. Such onerous computational requirements for estimating the associated generalization error are a major impediment to the real-world use of bagging.

This paper presents several techniques for circumventing this problem, and thereby facilitating the real-world use of bagging. In particular, it is shown how the bias-variance formula (Geman, Bienenstock & Doursat, 1992, Wolpert, 1996) can be exploited to estimate the generalization error of a bagged learning algorithm without additional training of the underlying learning algorithm; only the original ν training runs used to create the bagged algorithm's hypothesis are needed.

In a set of experiments, the accuracy of several such estimators based on the bias-variance formula are compared to both the accuracy of using cross-validation to estimate the generalization error of the bagged algorithm, and to the accuracy of using cross-validation to estimate the generalization error of the underlying learning algorithm. The best of our estimators exploits the technique of stacking (Wolpert, 1992). It actually estimates the bagged algorithm's generalization error more accurately than does the alternative cross-validation-based estimator of either the underlying or the bagged algorithm's error. This increased accuracy is a novel reason for using bagging. Importantly, this gain in accuracy becomes more pronounced as one moves to smaller test sets. (Note that in the real world, one is often only presented with a single test set element at a time.)

In Section 2 we review bagging and the bias-variance formula, and Section 3 presents several ways to exploit that formula to estimate bagging's error, the last of which turns out to be the best-performing. In Section 4 we present the results of our experiments comparing the accuracy of these estimators to each other and to the accuracy of cross-validation-based estimators. In Section 5 we discuss our results, and in Section 6 we present future work.

2. Bagging

We are interested in the "generalization error" C measuring the disparity between f and h_d on a test set X value q. For this paper, for a provided Y-sample of f at q (*i.e.*, a random sample of $P_f(\cdot|q)$), let the generalization error be the square of the difference between $h(q)$ and that sample. Next define the expectation value $h^*(q) \equiv E(h(q)|f,m) = \sum_d P(d|f,m)h_d(q)$; this is the guess at q made by the average h_d, formed by G in response to training sets d of size m sampled from f according to the likelihood given by $P(d|f,m)$. In this paper, we will implicitly assume the IID likelihood: $P(x^i, y^i|f,m) = \prod_{i=1}^m P_f(y^i|x^i)\pi(x^i)$, where $\pi(\cdot)$, the unconditioned distribution over X, is called the "sampling distribution".

Next define the average Y value of the target at q as $f^*(q) \equiv \sum_y yP_f(y|q)$. Then up to an overall additive constant that is independent of the learning algorithm, the expected value of C conditioned on f, the training set size m, and q, is

$$\begin{aligned} E(C|f,m,q) &= \textstyle\sum_d P(d|f,m)\big(h_d(q) - f^*(q)\big)^2 \\ &= \big(h^*(q) - f^*(q)\big)^2 + \textstyle\sum_d P(d|f,m)\big(h_d(q) - h^*(q)\big)^2. \end{aligned} \tag{1}$$

This is the famous bias-variance formula (Geman, Bienenstock & Doursat, 1992, Wolpert, 1996). The first term on the right-hand side is the (square of the) bias. It measures how well the average h matches f^*. The second term is the variance. It measures how much h_d "bounces around" as d's sampled from f change.

A learning algorithm is defined to be unstable if h_d is a sensitive function of d (Breiman, 1994b). For such a learning algorithm, the variance contributes substantially to the expected error. As an example, Breiman argues that due to the existence of many local minima in the energy surface, the learning algorithm of backprop-trained neural nets is unstable.

Now fix G and therefore $h^*(q)$. Consider a separate d-independent learning algorithm that responds with that same hypothesis $h^*(q)$ for any d it is "trained" on. That algorithm has the same bias as G. However since that algorithm's hypothesis does not vary with d, its variance is zero. Accordingly, if G is unstable, the expected error for that algorithm that always guesses $h^*(q)$ regardless of d is significantly less than that of G.

The difficulty with exploiting this effect to aid real-world generalization, of course, is the fact that we cannot evaluate $h^*(q)$ in practice, since we only have access to a single training set d, whereas $h^*(q)$ is defined in terms of an infinite number of training sets. Breiman's insight was that one can produce a "mimic" for $h^*(q)$. Intuitively, one starts with d, and then uses density estimation techniques to estimate f from d. One then samples that estimate repeatedly and runs the learning algorithm on those samples, and in this way produces a mimic of $h^*(q)$.

To the degree that one's mimic of $h^*(q)$ approximates the true $h^*(q)$ (regardless of the d used to create that mimic), it follows that guessing with the mimic $h^*(q)$ rather than h_d will result in lower expected generalization error. Breiman argues that one should similarly expect that bagging improves average misclassification generalization error (as opposed to the quadratic error discussed here).

The density estimation technique Breiman used also directly provided his samples of that density estimate: it was the bootstrap procedure (Efron & Tibshirani, 1993 , Efron, 1979). This is both the maximum-likelihood density estimator and an unbiased density estimator. The samples of the bootstrap-estimated f are called “replicates”. See (Wolpert & Macready, 1996) for a discussion of some theoretical caveats behind the utility of bagging, and (Wolpert, 1996) for theoretical caveats about the bootstrap procedure in general.

3. Estimating Bagging’s Error

3.1. Overview

Under the hypothesis of bagging, the mimic of $h^*(q)$ produced by bagging is a close approximation to the actual $h^*(q)$, *i.e.*, $h^*(q)$ closely approximates the bagged algorithm’s guess. Therefore, up to a learning-algorithm-independent constant κ, the bagged algorithm’s generalization error is closely approximated by $\big(h^*(q) - f^*(q)\big)^2$, *i.e.*, it is given by the square of the underlying algorithm’s bias. Accordingly, by Equation (1), under these assumptions one could find the error of the bagged algorithm at q by subtracting the underlying algorithm’s variance at q from the underlying algorithm’s expected error there. (κ cancels the similar learning-algorithm-independent constant in the bias-plus-variance formula.) We can write this as

$$E_{Bagged\ G}\ (q) \approx E_G(q) - V_G(q).$$

So to find (an approximation to) the average of the bagged algorithm’s error over all q in a test set, one simply averages the expected error of the underlying algorithm over all such q and subtracts from it the q-average of the underlying algorithm’s variance.

Both that underlying algorithm’s variance and expected error can be estimated in many different ways (*e.g.*, via cross-validation). For our purpose though, since we want to avoid additional retraining of our underlying algorithm, the natural thing to do is estimate that variance and expected error by using some variant of the bootstrap procedure — in forming the bagged algorithm we have already formed the $h_{d'_i}(q)$ needed by that procedure, and therefore with such an approach no retraining is needed to perform our estimations.

3.2. Estimation Schemes Based on the Bias-Variance Decomposition

Our task is to use the bootstrap replicates at hand to estimate the difference between the expected error and variance of the underlying algorithm. In this subsection, we first introduce several estimators all of which perform this task satisfactorily. These estimators all have some (subtle) shortcomings though. These shortcoming motivate our stacking-based estimator, which is introduced below. Although only a slight modification of the estimators introduced here, that stacking-based estimator significantly outperforms the other estimators.

Consider the following three estimators $V(q)$ for the variance at point q.

V_1: In the first variant, we start by calculating the variance at a training set point x^i as one varies the replicates. With ν the number of replicates,

$$V_1(x^i) \equiv \frac{\sum_j \Big(h_{d'_j}(x^i) - \big[\sum_j h_{d'_j}(x^i)/\nu\big]\Big)^2}{\nu - 1}.$$

We then take the average of the variance over the elements of the test set (which is what we're ultimately interested in) to equal the average of $V_1(x^i)$ over the x^i.

V_2: In the second variant, one is careful to avoid the over-fitting problems that can accompany $V_1(x^i)$. So one uses a modification of $V_1(x^i)$ where the two sums occurring in $V_1(x^i)$ each only run over those replicates that do not contain the point x^i (with "ν" implicitly redefined to be the number of such replicates).

V_3: The final variant is also almost identical to $V_1(q)$. Here the difference is that one does not calculate the $V(x^i)$ for all x^i and then assume that the variance at all test set points q is given by the average (over x^i) of $V(x^i)$. Rather one directly estimates the variance at the points q in the test set that one is interested in:

$$V_3(q) \equiv \frac{\sum_j \Big(h_{d'_j}(q) - \big[\sum_j h'_{d_j}(q)/\nu\big]\Big)^2}{\nu - 1}.$$

The average of the variance over the q in the test set is then simply taken to be the average of $V_3(q)$ over those q.

For the expected error of the underlying algorithm we started with the following two estimators:

E_1: $E_1(q)$ is like $V_1(q)$. For each x^i,

$$E_1(x^i) \equiv \frac{\sum_j \big(h_{d'_j}(x^i) - y^i\big)^2}{\nu}.$$

$E_1(q \notin \{x^i\})$ is undefined, and we take the average of $E_1(q)$ over the elements of the test set (which is what we're interested in), E_1, to equal the average of $E_1(x^i)$ over the x^i.

E_2: $E_2(q)$ is identical to $E_1(q)$, except that to avoid over-fitting problems the sum only goes over those replicates that do not contain x^i, and ν is implicitly redefined to the number of such replicates. Similar modifications of E_1 define E_2.

Since there are no y^i provided to us for the q in the test set, there is no analogue of V_3 for the expected error of the underlying algorithm.

Now for large enough ν, for both $i = 1$ and $i = 2$ the difference $E_i(x) - V_i(x)$ is almost never strongly negative, since it is closely approximated by $\big([\sum_j h_{d_j}(x)/\nu] - y\big)^2$ (where the sum implicitly only runs over the replicates appropriate for the choice of i). This is proper behavior, since by definition the bagged algorithm's error is non-negative.

Unfortunately though, unless one is using V_1 together with E_1, or V_2 together with E_2, it is common for the estimate of the bagged algorithm's error to be substantially negative. (*I.e.*, for $i \neq j$, $E_i - V_j$ is often very negative.) The simplest way to address such problems is to replace all such negative estimates with an estimate of zero (so our estimator becomes $[E_i - V_j]^+$). Alternatively, one could just refuse to use the offending combination of V_j and E_i for the situation at hand.

3.3. A More Nuanced Decomposition

There are several assumptions that lie behind the use of the bias-variance formula to estimate the bagged algorithm's error, not least of which is that the bagged algorithm's hypothesis is a good mimic of $h^*(q)$. Most of those assumptions can be circumvented if one instead uses the decomposition introduced in this section.

In actuality, we are not interested in an average error over many training sets; rather we are directly interested in the expected quadratic loss of the bagged algorithm given the precise replicates and test set input values at hand. *I.e.*, we are interested in the expectation value

$$Err(q) \equiv E\Big([\sum_j \frac{h_{d'_j}(q)}{\nu} - y]^2 \Big| f, \{d'_i\}, q \Big)$$

where q is a test set input value. Note that since q and the $\{d'_i\}$ are fixed in this expectation value (as they are whenever we use bagging in practice), the guesses of both the underlying and bagged algorithms are fixed, and the only varying quantity is the value of y, formed by sampling f at q.

Simple algebra verifies that we can write $Err(q) = E_c(q) - V_c(q)$, where

$$E_c(q) \equiv E\Big(\sum_j \frac{[h_{d'_j}(q) - y]^2}{\nu} \Big| f, \{d'_i\}, q \Big),$$

and

$$V_c(q) \equiv \frac{1}{\nu} \sum_j \Big[h_{d'_j}(q) - \big(\sum_i h'_{d_i}(q)/\nu \big) \Big]^2.$$

Note that $V_c(q)$ is identical to $V_3(q)$, except that we divide by ν rather than $\nu - 1$. In particular, we can measure $V_c(q)$ exactly for all q in the test set, just as we can measure $V_3(q)$ exactly for all such q. However there is one crucial distinction between the two quantities: In $V_c(q)$ we have a direct evaluation of one of quantities occurring in the associated decomposition. In contrast, $V_3(q)$ is an *estimate* of a quantity occurring in its associated decomposition (namely, an estimate of the d-averaged variance).

Because $V_c(q)$ as empirically measured is exactly one of the two quantities occurring in the decomposition of what we really want to know (namely $Err(q)$), the only quantity we must estimate — the only possible source of approximation error in our new decomposition - — is $E_c(q)$. This contrasts with the schemes defined above, for which two quantities must be estimated; with this new decomposition not only are we directly addressing what we really want to know, we also have one fewer quantity to estimate.

An obvious way to try to exploit our knowing V_c exactly is to estimate E_c with one of the E_i (E_1 or E_2) introduced above, and then estimate bagging's error by subtracting V_c from that E_i. However if we simply use $E_i(q) - V_c(q)$ as our estimator with one of the $E_i(q)$ introduced above, we do not get an improvement over using $E_i(q) - V_i(q)$ as our estimator (see experimental results below). This despite the fact that that $V_c(q)$ is an (infinitely) more accurate estimate of the quantity we wish to subtract from E_c than is $V_i(q)$.

Part of the reason for this is that whereas (usually) $[E_i(q) - V_i(q)] \geq 0$ —as is $Err(q)$— the quantity $[E_i(q) - V_c(q)]$ is often substantially negative. If one compensates for this by forcing non-negativity on the $E_i(q) - V_c(q)$ estimator (by replacing it with $[E_i(q) - V_c(q)]^+$ — see above), performance improves. However the performance still is no better than that associated with the best of the $E_i(q) - V_i(q)$ estimators, the $E_2(q) - V_2(q)$ estimator (see experimental results below).

To understand why the $[E_2(q) - V_c(q)]^+$ estimator does not outperform the $E_2(q) - V_2(q)$ estimator at approximating $E_c(q) - V_c(q)$, note that the inaccuracy accompanying the replacement of $E_c(q)$ with $E_i(q)$ — an inaccuracy both the $E_i(q) - V_i(q)$ estimator and the $[E_i(q) - V_c(q)]^+$ estimator must make — can be aligned in sign with the inaccuracy accompanying the replacement of $V_c(q)$ with $V_i(q)$. In such cases, replacing $V_c(q)$ with $V_i(q)$ in our estimator may actually result in a smaller absolute value of the inaccuracy in our estimate of $E_c(q) - V_c(q)$.

This aligned-signs behavior is not a sufficient condition for the superiority of the $E_i(q) - V_i(q)$ estimator; $[E_i(q) - V_c(q)]^+$ may still be superior if $|V_c(q) - V_i(q)|$ is large compared to $|E_c(q) - E_i(q)|$. In fact, if for some particular i the random variable $\omega_V^i(q) \equiv V_c(q) - V_i$ were uncorrelated (as one varies over q) with the random variable $\omega_E^i(q) \equiv E_c(q) - E_i$, then even the naive $E_i(q) - V_c(q)$ estimator (which has no positivity constraint) would be more correlated with $Err_c(q) = E_c(q) - V_c(q)$ than would be the $E_i - V_i$ estimator. Intuitively speaking, if one takes two random steps in a random walk where the steps are uncorrelated with one another (and both have mean zero), then one must, on average, have traveled further than if one had only taken a single random step.

The fact that this is not the case (see experiments below) means that $\omega_E^i(q)$ and $\omega_V^i(q)$ are correlated. This is not unexpected; for example for fixed values of E_i and V_i, if $\omega_E^i(q)$ is quite small, then $\omega_V^i(q)$ cannot be too large, since $Err(q) = E_i - V_i + \omega_E^i(q) - \omega_V^i(q)$ is always strictly non-negative.

In the limit where $\omega_E^i(q)$ and $\omega_V^i(q)$ are perfectly correlated, $E_i - V_i = E_c(q) - V_c(q)$ exactly, and you can do no better than use the $E_i - V_i$ estimator. However we would never expect to have such perfect correlation. (In our experiments, we had quite strong but not perfect correlation — correlation coefficients were on the order of 0.9 for $i = 2$.) Accordingly there is room to try to improve upon the $E_i - V_i$ estimator. One way to do this would be a Bayesian approach, where one uses the data $\mathcal{D}$ provided by the off-replicate points in the training set to estimate the expectation value $E(E_c(q) - V_c(q) \mid \mathcal{D}, V_c(q), E_c(q) \geq V_c(q), E_2, V_2)$. An alternative approach, somewhat akin to empirical Bayesian analysis, is described in the next section.

3.4. *Using Stacking to Exploit Exact Knowledge of* $\boldsymbol{V_c(q)}$

Rather than use something as complicated as a full Bayesian approach to exploit the fact that we know $V_c(q)$ exactly, we can instead directly modify our estimate of $E_c(q)$ based on its observed relationship with $V_c(q)$. In particular, one can use the bootstrap information the replicates provide to estimate the relation between $V_c(q)$ and $E_c(q)$, with the estimation constrained to enforce the condition $E_c(q) \geq V_c(q)$. One would then use the resultant probability distribution over possible values of $E_c(q) - V_c(q)$ to estimate $E_c(q)$ from the observed value of $V_c(q)$. Note that in doing this one is positing that there is a strong relationship between $V_c(q)$ and $E_c(q)$.

More directly, one can use the bootstrap information to directly estimate $Err(q)$ from $V_c(q)$, and then combine that estimate "conservatively" with the $E_2 - V_2$ estimate (say by averaging). This in essence amounts to single-generalizer stacking with a bootstrap partition set (Wolpert, 1992), where rather than try to improve the generalization of one's system, here one is instead trying to improve one's estimate of the system's generalization accuracy. (See (Wolpert & Macready, 1996) for discussion of how one can also successfully use this same basic idea of combining bagging and stacking to get lower generalization error than that given by conventional bagging.)

To implement this idea we used a simple linear model, $Err(q) = aV_c(q) + b$, to approximate the mapping from $V_c(q)$ to $Err(q)$. The slope a and intercept b were estimated by minimizing the summed squared error between the line $Err(q) = aV_c(q) + b$ and the m pairs $\{V_c(x^i), Err(x^i)\}$ provided by the replicates and the training set. As stacking advises, for each x^i, to avoid overfitting we calculated $V_c(x^i)$ and $Err(x^i)$ by only using those replicates d'_j that do not contain the point x^i. (So in particular the value of ν for these calculations varied from one (x^i, y^i) pair to the next, in general.)

Given such a fit, one might simply estimate the accuracy of the bagged algorithm at point q as $aV_c(q) + b$. However following the advice of stacking (see (Wolpert, 1992)), we chose to be "conservative". The accuracy of the linear fit can be estimated by looking at its χ^2 residual error. If that error is large, we should be wary of our stacking-based estimate of $Err(q)$, and might prefer to instead use an estimator like $E_2 - V_2$. One way to implement this idea starts by defining a threshold c. Then if $\chi^2 > c$ (*i.e.*, if we had a poor linear fit) we estimated the error as $E_2 - V_2$ while if $\chi^2 < c$ (*i.e.*, a good fit) we estimated the average $Err(q)$ over the test set as

$$Err = \frac{1}{|\text{test set}|} \sum_{q \in \text{test set}} aV_c(q) + b$$

By setting the size of c we can tune how conservative we are in our estimate.

We have also investigated a "weighted" scheme combining both guesses rather than switching between them based on a threshold. In this scheme our estimate of the bagged algorithm's error is given by

$$Err = \frac{1}{|\text{test set}|} \sum_{q \in \text{test set}} \alpha(\chi^2)\big(aV_c(q) + b\big) + \beta(\chi^2)\big(E_2 - V_2\big)$$

where $\alpha(\chi^2) = 1/(1 + \chi^2)$ and $\beta(\chi^2) = \chi^2/(1 + \chi^2)$.

Note that both of these χ^2-based schemes ignore many issues. For example, the bagged algorithm's guess is based on the full ν replicates. However the number of replicates going into the calculations of each $(V_c(x^i), Err(x^i))$ pair varies. When that number of replicates is small, one would expect the values of the associated $(V_c(x^i), Err(x^i))$ pair to be a relatively poor indicator of the relationship between $V_c(q)$ and $Err_c(q)$ for the full (based on all ν replicates) bagged algorithm. There are a number of ways to take this into effect in estimating the values of the coefficients a and b; we plan to explore some of them in the near future.

4. Experiments

4.1. Experimental setup

Again following the lead of Breiman, the experiments reported in this paper involved simple linear models. Because measuring the leave-one-out cross-validation error estimate for a bagged learning algorithm (the error-estimation scheme to which we're comparing our techniques) is so time-intensive, we used rather simple experiments in this paper. There was noise and (unlike in Breiman's experiments) model-misspecification, but the input space was only one-dimensional. Future work involves extending these experiments to more elaborate domains.

As in the first set of experiments in (Wolpert & Macready, 1996), our input and output space were $\mathcal{R}^1$. The target function was a third-order polynomial with two Gaussians superimposed. The coefficients of the polynomial were randomly chosen from [-1, 1]. The centers of the two Gaussians were randomly chosen from [-1, 1], and the widths were randomly chosen from [0.1, 0.3]. The coefficient of the Gaussians was always 1. Given such a randomly generated target function Gaussian noise of width 0.05 was superimposed to create training sets. The X-components of training sets were chosen randomly from [-1.0, 1.0]. Test sets consisted of 100 points chosen the same way as training sets, with the same amount of noise added.

To understand our learning algorithm, first consider the following four different learning algorithms. These learning algorithms all work by forming the linear combination of six basis functions with the least-squared-error fit to the training set. (No regularization was used.) Each learning algorithm's set of six basis functions consisted of the cosines and sines of three wavelengths. The difference between the learning algorithms was simply in the choice of those wavelengths. All of the learning algorithms used wavelengths from the set $\{0.5, 1, 2, 4\}$; since there are four sets of triples in that set, we had four learning algorithms all told.

All four of these learning algorithms are stable. Accordingly, for them, the bagged learning algorithm does not outperform the underlying learning algorithm. (It performs slightly worse – see (Wolpert & Macready, 1996).) Accordingly, a single unstable learning algorithm is created from these four stable ones. This unstable learning algorithm again works by forming a least-squared-error fit to the training set, using a set of three cosines and associated sines. However to create instability, the choice of the three wavelengths from the set $\{0.5, 1, 2, 4\}$ depends sensitively on the training set. The choice was made by forming the sum of the input and output components of the training set, multiplying it by

Table 1. Summary statistics for some of the (V_i, E_j) variants. Any uncertainties in the results are in the last significant digit. "$|\Delta$ error$|$" is the average, over the 50000 targets, of the absolute difference between estimated and true errors for each such target. "Correlation coefficient" also is between those two errors. The "slope" and the "intercept" values refer to a least mean squares linear fit between those two errors, and are included simply to provide a crude characterization of the associated scatter plots. All statistics here and throughout are based on simple frequency counts.

statistic	(V_1, E_1)	(V_2, E_2)	(V_3, E_1)	(V_3, E_2)
Estimated error	0.142	0.281	0.02	0.392
Actual test set error	0.292	0.293	0.293	0.293
$\lvert\Delta$ error$\rvert$	0.152	0.117	0.274	0.290
Correlation coefficient	0.678	0.677	0.135	0.195
Best fit slope	0.288	0.601	0.03	0.94
Best fit intercept	0.058	0.105	0.01	0.116

10, and then evaluating the result modulo 4. The resultant number was either 0, 1, 2, or 3. Which of those four numbers arose determined which of the four triples from the set of four possible wavelength-sets to use.

Note there is no compelling reason to use such a learning algorithm in the real world — it was simply a convenient way of introducing instability, which is something that in the real world you will usually want in order for bagging to be of interest. (Indeed, as expected, whereas bagging does not help the performance of any one of our learning algorithms used by itself, it does improve the performance of the unstable combination of those algorithms.) Moreover, as far as the experiments in this paper on estimating generalization accuracy are concerned, each of the the four bagged variants of a stable learning algorithm gives results that are very similar to the experimental results described below for the unstable bagger.

4.2. Results of the Experiments

The bagged generalizer was formed by combining $\nu = 50$ bootstrap replicates with $m = 25$. To obtain our results we generated 50000 target functions as described above, one new set of 50000 for each estimator. For each of these target functions one training set was generated with noise, as described above. Then the test set error was estimated using one of the schemes described above and this estimate was compared with the actual error on the (randomly generated) test set.

The summary statistics of some of the variants are presented in Table 1. For those variants for which $E_j - V_i$ can be negative we replace any negative estimate with the value 0. Of the six simple (V_i, E_j) estimators, (V_2, E_2) was found to be the most effective.

For comparison we have also looked at the correlations in the error estimated by leave-one-out cross-validation for both the underlying (unstable) learning algorithm G and the bagged learning algorithm. These results are presented in Table 2.

Unsurprisingly, the (V_1, E_1) estimator performs rather poorly. Because E_1 is measured on points in the training set this estimation procedure consistently underestimates the true

Table 2. Summary statistics for leave-one-out cross-validation schemes. See the caption for Table 1 for definitions of the quantities reported.

statistic	x-val G	x-val bag
Estimated error	0.735	0.301
Actual test set error	0.739	0.299
$\lvert\Delta$ error$\rvert$	0.303	0.117
Correlation coefficient	0.694	0.699
Best fit slope	0.777	1.114
Best fit intercept	0.168	-0.032

Table 3. Summary statistics for the "conservative" and "weighted" stacking-based estimation scheme described above for test sets of size 100, as in Table 1. See the caption for Table 1 for definitions of the quantities reported.

statistic	"conservative" $c = 1$	"weighted"
Estimated error	0.291	0.294
Actual test set error	0.293	0.294
$\lvert\Delta$ error$\rvert$	0.115	0.111
Correlation coefficient	0.691	0.706
Best fit slope	0.607	0.623
Best fit intercept	0.113	0.112

Table 4. Summary statistics for $E_2 - V_2$, cross-validation and "weighted estimates" for error on test sets of size 1. See the caption for Table 1 for definitions of the quantities reported.

statistic	$E_2 - V_2$	x-val	"weighted"
Estimated error	0.280	0.296	0.294
Actual test set error	0.294	0.296	0.294
$\lvert\Delta$ error$\rvert$	0.314	0.306	0.295
Correlation coefficient	0.251	0.267	0.353
Best fit slope	0.08	0.10	0.12
Best fit intercept	0.26	0.27	0.26

error on the training set. (Although interestingly, the correlation between its guess and the true error is comparable to that of the (V_2, E_2) estimator.)

However the results from the (V_2, E_2) estimator are very encouraging. In particular, for this technique $\lvert\Delta$ error$\rvert$ is no worse than that of the (far more expensive) leave-one-out cross-validation estimator of the bagged algorithm's error. In addition there is almost as much correlation between true and estimated errors for this estimator as there is for the leave-one-out cross-validation estimator. Moreover, the correlation seems to improve further with larger training sets.[1]

In Tables 3 and 4 we present the results for the stacked estimates of the error. We find that both the conservative scheme with $c = 1$ and the weighted scheme where α and β

are determined by χ^2 improve upon the $E_2 - V_2$ estimate and perform as well as or better than cross-validation. Of the two stacking algorithms, the weighted one appears to be the better performing. (We have not explored other stacking algorithms — we have not even investigated other values of c— so further gains are possible). For smaller test sets we found even greater gains in using stacked estimates rather than simply $E_2 - V_2$. For test sets of size 1 we gained almost 40% in correlation between our estimate of error and actual error by using stacking.

In summary, for the problems investigated here, we have found that the (V_2, E_2) estimator works almost as well as cross-validation in estimating error. Moreover, the more sophisticated stacking-based schemes actually lead to estimates as accurate, and in some cases, more accurate than cross-validation. This provides a novel justification for using bagging — by bagging an algorithm, one can use an error estimator that is more accurate than leave-one-out cross-validation applied to the underlying algorithm.[2]

5. Discussion

Our experiments indicate that it is indeed possible to accurately estimate a bagged algorithm's generalization error without extra training of the underlying algorithm. This removes one of the major obstacles to real-world use of bagging. In fact, based on the results in this paper, one might argue that one reason to use bagging is that when its generalization error is estimated using the techniques described in this paper, the resultant estimate is often more accurate than the estimate of the generalization error of the underlying algorithm given by conventional cross-validation. (The details of this comparison where one tries different kinds of cross-validation besides leave-one-out – including in particular bootstrap-based variants — are the subject of future work.)

Simultaneously with our work, Tibshirani conducted a similar study (Tibshirani, 1996). In his study, he investigated the (V_2, E_2) estimator, for classification problems (as opposed to the regression problems studied in this paper). The results in (Tibshirani, 1996) are quite encouraging; they suggest that the basic idea of the (V_2, E_2) estimator also works well on classification problems. Combined with our results this suggests that the (V_2, E_2) estimator may be broadly applicable.

Our work goes on to complement Tibshirani's study in a number of ways. We have considered other estimators beyond those concurrently investigated by Tibshirani. In particular, we have presented an estimator that exploits stacking and that improves upon the already effective (V_2, E_2) estimator also investigated by Tibshirani, at least for the problems we have investigated to date. Moreover, we have made comparisons to cross-validation, comparisons that have demonstrated that these estimation schemes sometimes work better than cross-validation. This result potentially provides a novel reason to use bagging — improved estimation of generalization error.

In addition, along with results similar to those Tibshirani presented concerning the bias of using the (V_2, E_2) estimator to estimate generalization error, we have also considered other measures of the efficacy of that estimator (and of other estimators). Such results are important because bias alone can be highly misleading. For example, one could conceivably have zero bias, but perfect anti-correlation between one's estimator and the quantity being estimated. Our experiments examine that (and similar) possibilities; we have found such pernicious anti-correlation not to obtain, and instead observed strong correlation between estimated and true error.

Finally, we have also drawn attention to the importance of these estimation schemes in light of the computational cost of bagging. Moreover, we have placed the estimation problem in the context of the bias-variance decomposition.

In addition to this work of ours and the work of Tibshirani's, there has been subsequent work by Breiman that explores yet more ways to use the out-of-replicate instances generated in bagging to aid estimation of error rates (Breiman, 1996). That work by Breiman also touches on how to use the out-of-bag instances to improve the error, not just the estimate thereof, much as in our work on bootstrap stacking (Wolpert & Macready, 1996).

6. Future Work

In addition to the work mentioned in the text, there are many other ways that the investigations presented in this paper should be extended. Most obviously, it is important to investigate the performance of the stacking-based estimator on higher dimensional problems than those investigated here. That estimator's performance on classification problems should also be explored. In addition, we plan to explore stacking schemes where $Err(q)$ depends on other variables besides or in addition to $V_c(q)$ (e.g., have it depend on q). More generally, we plan to explore other kinds of stacking schemes, e.g., a more principled way of using the χ^2 misfit in the stacking to determine how best to combine the stacker's estimate of $Err(q)$ with the $E_2 - V_2$ estimate of $Err(q)$.

As mentioned above, other future work involves modifying the stacking so that the (varying) number of distinct elements in each replicate is taken into account. We also plan to investigate schemes for correcting for replicate-size-based effects with the $E_i - V_i$ estimators. For example, E_2 is the average over the inputs $\{x^i\}$ of the off-sample-estimated error at the $\{x^i\}$. If we define $I_{ij} \equiv 0$ if $x^i \in (d'_j)_X$ and 1 otherwise, then we can write this as

$$E_2 = \sum_{ij} \frac{I_{ij}[h_{d'_j}(x^i) - y^i]^2}{m \sum_{j'} I_{ij'}}$$

However this differs in general from another reasonable candidate to use to estimate E: the average over the replicate-generated hypotheses $h_{d'_j}$ of the off-sample-estimated error of the $h_{d'_j}$. (This alternative quantity involves division by $\nu \sum_{i'} I_{i'j}$ instead of $m \sum_{j'} I_{ij'}$.) Note that with this alternative estimator, it may make sense to weight each term in our sum so as to take into account the fact that the larger $\sum_{i'} I_{i'j}$ is, the fewer points went into the training that produced $h_{d'_j}$, and therefore the less we would expect the associated error to be like that of h_d. On the other hand though, under the same conditions, the error estimate we have formed for $h_{d'_j}$ is based on more points (namely, all the points x^i for which I_{ij} is nonzero). One might therefore presume that that error estimate more accurately reflects $h_{d'_j}$'s actual error. So there are at least two opposing effects as one varies from replicate to replicate.

To further complicate matters, one can consider yet other alternative estimators. For example, one might simply ask for the average over all i, j of $I_{ij}[h_{d'_j}(x^i) - y^i]^2$. In other words, one might replace the normalization factor $m \sum_{j'} I_{ij'}$ by $\sum_{i'j'} I_{i'j'}$. Future work involves disentangling all these effects to produce a better variant of E_2 that can then be compared to the stacking-based estimators.

Taking a different approach, in those cases where we can afford to do some cross-validation (though would prefer not to), whenever one or more of the (V_i, E_j) estimation schemes gives dubious results (*e.g.* when it estimates a negative squared error) we can resort to cross-validation. In other words, we can use such dubious results as a warning flag. Future work involves characterizing the performance of such a hybrid estimator.

It is worth noting that many of the concepts introduced in this paper can be extended to other problems besides the estimation of a bagged algorithm's generalization error. For example, the basic insight that one can measure variance exactly could perhaps be used to improve the accuracy of cross-validation schemes in which there is overlap between the validation sets so that the calculated cross-validation value can be viewed as a sum of (an estimate of) bias and (an estimate of) variance. (In the variant of such a scheme being suggested here, one would only estimate bias by means of partitions of the training set - — variance would be measured directly on the test set.) In particular, stacking could perhaps be used to effect this improvement. Similarly, the schemes explored in (Wolpert & Macready, 1996) use stacking to shrink the generalization error of bagging; the results of this paper suggest that having $V_c(q)$ be one of the variables involved in the stacking would result in even greater improvement in generalization error.

Another avenue of future work is to investigate scenarios where h_d is non-single-valued, and in fact is an estimate for f. Such scenarios are quite common, arising for example whenever a Bayesian learning algorithm is used. (For quadratic loss, for example, one's final generalization guess at q with such a learning algorithm would be $\sum_y yh_d(y|q)$.) For such scenarios, the replicates going into the bagging can be formed by sampling h_d rather than d. If h_d is a better estimate of f at the points $\{x^i\}$ than is d - — which one would hope is a common case — then such sampling of h_d to form bagging's guess may result in lower generalization error than does conventional bagging. (See (Breiman, 1994a) for related discussion.) An even more speculative issue is how best to estimate generalization accuracy for such a modification of bagging.

Acknowledgments

DHW was supported in part by TXN Inc. and the Santa Fe Institute, and WGM was supported by the Santa Fe Institute and Bios Group. DHW thanks Ronny Kohavi for interesting discussion. We thank Michael Pazzani for helpful comments.

Notes

1. We came to this conclusion based on an additional set of experiments for the (V_2, E_2) estimator that kept all parameters the same but increased the training set size to 60. Doing this resulted in the correlation coefficient increasing to 0.913. (The means of the estimated and true errors for those experiments are 0.219 and 0.227 respectively.)
2. Compare the "weighted" results of section (b) of Table (3), especially the results for the correlation coefficient, to the "x-val G" results of Table 2.

References

Breiman, L. (1994a). Bagging predictors. Univesity of California, Dept. of Statistics, TR 421.

Breiman, L. (1994b). Heuristics of instability and stabilization in model selection. University of California, Dept. of Statistics, TR 416.

Breiman, L. (1996). Out-of-bag estimation. University of California, Dept. of statistics.

Efron, B. (1979). Computers and the theory of statistics: thinking the unthinkable. *SIAM Review, 21*: 460.

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

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

Tibshirani, R. (1996). Bias, variance and prediction error for classification rules. University of Toronto Statistics Department Technical Report.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*: 241–249.

Wolpert, D. H. (1996). The bootstrap is inconsistent with probability theory. *In "Maximum Entropy and Bayesian Methods", K. Hanson and R. Silver (Eds)*, pages 69–76.

Wolpert, D. H. (1996). On bias plus variance. Neural Computation, in press.

Wolpert, D. H. & Macready, W. G. (1996). Combining stacking with bagging to improve a learning algorithm. Submitted.

Received August 14, 1996
Accepted August 24, 1998
Final Manuscript August 24, 1998