Beating the Hold-Out: Bounds for K-fold and Progressive Cross-Validation

Avrim Blum[*]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
avrim+@cs.cmu.edu

Adam Kalai[†]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
akalai+@cs.cmu.edu

John Langford
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
jcl+@cs.cmu.edu

Abstract

The empirical error on a test set, the *hold-out estimate*, often is a more reliable estimate of generalization error than the observed error on the training set, the *training estimate*. K-fold cross validation is used in practice with the hope of being more accurate than the hold-out estimate without reducing the number of training examples. We argue that the k-fold estimate does in fact achieve this goal. Specifically, we show that for any nontrivial learning problem and learning algorithm that is insensitive to example ordering, the k-fold estimate is strictly more accurate than a single hold-out estimate on 1/k of the data, for $2 < k < n$ ($k = n$ is leave-one-out), based on its variance and all higher moments. Previous bounds were termed sanity-check because they compared the k-fold estimate to the training estimate and, further, restricted the VC dimension and required a notion of hypothesis stability [2]. In order to avoid these dependencies, we consider a k-fold hypothesis that is a randomized combination or average of the k individual hypotheses.

We introduce *progressive validation* as another possible improvement on the hold-out estimate. This estimate of the generalization error is, in many ways, as good as that of a single hold-out, but it uses an average of half as many examples for testing. The procedure also involves a hold-out set, but after an example has been tested, it is added to the training set and the learning algorithm is rerun.

1 INTRODUCTION

In many situations, a learning algorithm must simultaneously produce a hypothesis having low generalization error and a high-accuracy estimate of this error. We make the usual assumption that data is drawn independently from some fixed distribution, and the error of a hypothesis on examples from this distribution is called the generalization error or *true error*.

Several procedures exist for generating pairs of the form $\langle$hypothesis, estimated error$\rangle$. Such a procedure can be scored in two dimensions: the true error of its hypothesis, and the *error discrepancy*: $|(\text{estimated error}) - (\text{true error})|$.

The *resubstitution procedure* generates a hypothesis by training on all the data and generates an error estimate by measuring the number of mistakes of the learned hypothesis on the same data used for training. Since the training error can be a very optimistic estimate of the true error, quantities such as the VC dimension are used to bound the error discrepancy.

The *hold-out procedure* divides the data in two parts: the training set, on which the hypothesis is trained, and the hold-out set, on which its performance is measured. Among the nice properties that this procedure obeys are Hoeffding bounds guaranteeing, regardless of the learning algorithm, that with high probability the error discrepancy will be small.

The *k-fold procedure* divides the data into k equally sized *folds*. It then produces a hypothesis by training on $k-1$ folds and testing on the remaining fold. This is repeated for each fold, and the observed errors are averaged to form the *k-fold estimate*. It is not obvious what hypothesis to output along with this error estimate. In previous analysis [2], the final hypothesis was a new hypothesis trained on all the data. Because this hypothesis was constructed using more data than the hypotheses used for computing the error estimate, in order to argue for the accuracy of the estimate one needs some assumption that limits the effect of this extra training data. In particular, previous work gives *sanity-check bounds* which show that the k-fold estimate is almost as good as the training error estimate in the resubstitution procedure, under the assumption that the learning algorithm has some form of hypothesis stability.

Our k-fold procedure, instead, outputs the *k-fold hypothesis*, a meta-hypothesis that, given an example x, randomly chooses one of the k generated hypotheses h_i and outputs the prediction of that hypothesis, $h_i(x)$. Alternatively, if hypotheses are allowed to make predictions in $[0, 1]$ and we are using L_1 loss, this is equivalent, in terms of its true error, to outputting the average value of $h_1(x), \ldots, h_k(x)$ when the true labels are $0, 1$. We show that this k-fold procedure pro-

[*]Supported in part by NSF grant CCR-9732705
[†]Supported by an NSF Graduate Fellowship.

duces a better estimate than the hold-out procedure in sense that that the error discrepancy has smaller absolute moments, and that Hoeffding bounds still apply.[1]

The *progressive validation procedure*, like the hold-out procedure, first selects s examples for testing, and the remainder are for training only. It generates a sequence of s hypotheses, where the ith hypothesis is trained on all of training data *plus* the first $i-1$ examples of the test set, and tested on the ith example of the test set. Repeating this for $1 \leq i \leq s$, we count the number of mistakes to produce an error estimate. The hypothesis returned, as above, is a meta hypothesis which randomly selects among the s generated hypotheses to make its prediction. This procedure is very similar to methods used to convert online to batch learning algorithms [9, 7], but the kinds of guarantees we are looking for are somewhat different. In particular, we argue that the progressive validation procedure gives as good an estimate as the hold-out procedure with a hold-out of size s, while training on more examples.

2 PRELIMINARY DEFINITIONS

Let X be the instance space and let $\mathcal{D}$ be a fixed distribution over X. We also assume a fixed target function $f : X \longrightarrow \{0,1\}$. A learning algorithm produces a hypothesis $h : X \longrightarrow [0,1]$. We allow the range to be $[0,1]$, for convenience, so that we have a notion of averaging hypotheses. The error of this hypothesis on a particular example $x \in X$ is $e_h(x) = |h(x) - f(x)|$. The *true error* of this hypothesis is $\bar{e}_h = E_{x \in \mathcal{D}}[e_h(x)]$.

3 K-FOLD ANALYSIS

Imagine that we will flip an unfair coin ten times, and we want to estimate the probability of heads p. The full estimator "$\hat{p}_{10} = (\text{total number of heads})/10$" seems better than the one-flip estimator "$\hat{p}_1 = 1$ if the first flip is a head and $\hat{p}_1 = 0$ otherwise", but in what sense? For $p = 1/100$, the chance that $|\hat{p}_1 - p| > 0.05$ is 1/100, while the chance that $|\hat{p}_{10} - p| > 0.05$ is nearly 10/100, namely the chance that any of the flips were heads. Thus, $\hat{p}_{10}$ doesn't completely dominate $\hat{p}_1$ under every conceivable notion of "better". Instead, what *can* be said is that $E\left[|\hat{p}_{10} - p|^m\right] < E\left[|\hat{p}_1 - p|^m\right]$, for all $m \geq 1$. We make a similar statement about the k-fold procedure in comparison to a hold-out of size n/k.

Say we have a labelled data set of size n, and $1 < k \leq n$. We divide the data into k equally sized *folds*. Then we generate k hypotheses, $h_1, \ldots, h_k$, where h_i is trained on all the data except the ith fold. We let $\bar{e}_i = \bar{e}_{h_i}$ be the true error of h_i, and $\hat{e}_i$ be the measured error frequency of h_i on the ith fold. As discussed in the introduction, the *k-fold hypothesis*, h_{K}, makes a prediction on an example x by randomly choosing $1 \leq i \leq k$ and outputting $h_i(x)$ or, equivalently in terms of true error, by choosing $h_{\mathrm{K}}(x) = (h_1(x) + h_2(x) + \cdots + h_k(x))/k$. In either case, the true error of the k-fold hypothesis is the average of the true errors of its k hypotheses,

$$\bar{e}_{\mathrm{K}} = \frac{\bar{e}_1(x) + \bar{e}_2(x) + \cdots + \bar{e}_k(x)}{k}.$$

Finally, we let the k-fold error estimate be the average of the fold estimates, $\hat{e}_{\mathrm{K}} = (\hat{e}_1 + \hat{e}_2 + \cdots + \hat{e}_k)/k$.

Notice that the estimated and true errors of the k hypotheses and k-fold hypothesis, $\hat{e}_i, \bar{e}_i, \hat{e}_{\mathrm{K}}, \bar{e}_{\mathrm{K}}$, are random variables that are functions of the data set. We would like the error discrepancy $|\hat{e}_{\mathrm{K}} - \bar{e}_{\mathrm{K}}|$ to be small in absolute value.

We begin by showing that moments of the error discrepancy $|\hat{e}_{\mathrm{K}} - \bar{e}_{\mathrm{K}}|$ are *no larger than* those of a single hold-out of size n/k. Notice that the error discrepancy of a single hold-out is $|\hat{e}_1 - \bar{e}_1|$. The following theorem takes the trivial observation that the k-fold error is an unbiased estimate of the true error a step further. Expectations, unless otherwise noted, are over complete data sets drawn i.i.d. from $\mathcal{D}$.

Theorem 1 *For all $m \geq 1$, $E\left[(\text{error discrepancy})^m\right]$ is no larger for the k-fold procedure than for a hold-out of a $1/k$ fraction of the data, i.e.,*

$$E[|\hat{e}_K - \bar{e}_K|^m] \leq E[|\hat{e}_1 - \bar{e}_1|^m].$$

Proof. Jensen's inequality for any convex function f and reals x_i is,

$$f\left(\frac{x_1 + x_2 + \cdots + x_n}{n}\right) \leq \frac{f(x_1) + f(x_2) + \cdots + f(x_n)}{n}.$$

Because $|x|^m$ is convex for all $m \geq 1$,

$$\begin{aligned} |\hat{e}_{\mathrm{K}} - \bar{e}_{\mathrm{K}}|^m &= \left|\frac{\hat{e}_1 - \bar{e}_1 + \cdots + \hat{e}_k - \bar{e}_k}{k}\right|^m \\ &\leq \frac{|\hat{e}_1 - \bar{e}_1|^m + \cdots + |\hat{e}_k - \bar{e}_k|^m}{k}. \end{aligned}$$

Using linearity of expectation and that, for $1 \leq i \leq k$, $E[|\hat{e}_1 - \bar{e}_1|^m] = E[|\hat{e}_i - \bar{e}_i|^m]$ the expected value of the right-hand side is $E[|\hat{e}_1 - \bar{e}_1|^m]$, whereas the expected value of the left-hand side is $E[|\hat{e}_{\mathrm{K}} - \bar{e}_{\mathrm{K}}|^m]$. This completes the proof. ■

Now we wish to show that the k-fold error is a better estimate. However, it is possible that the hold-out error is a perfect estimate of the true error, if, for example, the learned hypothesis has true error equal to 0 or 1. To say something meaningful, we need to assume the learning algorithm has the property that $Pr[\hat{e}_1 \neq \bar{e}_1] > 0$ (all probabilities are taken over the draw of the full data set). In addition, our proof will need to assume that the instance space X is finite, and that the learning algorithm is insensitive to example ordering. This insensitivity can be enforced in our k-fold procedure simply by shuffling the training examples before giving them to the learning algorithm, on each of the k runs.

It is interesting to note that the k-fold estimate can be identical to the single hold-out estimate if $k = n$ or $k = 2$. In the case where $k = n$ (leave-one-out), Kearns and Ron [8] give several nice examples of poor performance. For instance, a learning algorithm that uses the rule "if I have seen an even number of positive examples then predict positive, else predict negative" will have the property that no matter

[1] Since our bounds compare an estimate to the hold-out estimate instead of the training error estimate, they are not sanity-check bounds, so they must be *insanity-check bounds*.

what the data, $\hat{e}_1 = \hat{e}_2 \ldots = \hat{e}_n$; thus the leave-one-out estimate will be exactly the same as a hold-out of size 1. Furthermore, if the underlying distribution has 50% positive examples, then the true errors will be the same as well. In the case where $k = 2$, an example is as follows. Suppose that we are to predict the label of integers drawn uniformly in some range $[1, \ldots, 2t]$, and the truth is that all labels are 0. Our hypotheses have a single parameter p, predicting p on even integers, and $1 - p$ on odd integers, thus having true error 50% regardless of p. Furthermore, our "learning" algorithm chooses p to be the fraction of even examples seen in the input. Now, if $k = 2$, we will have two hypotheses with p_1 and p_2, and $\hat{e}_1 = p_1 p_2 + (1 - p_1)(1 - p_2) = \hat{e}_2$. So the two-fold estimate, which is identical to the hold-out estimate, is no better an estimate of the 50% true error.

Theorem 2 *Suppose the example space is finite, our learning algorithm is insensitive to example ordering, and the hold-out estimate is not always perfect, i.e.* $Pr[\hat{e}_1 \neq \bar{e}_1] > 0$. *Then, for* $2 < k < n$ *and* $m \geq 2$,

$$E[|\hat{e}_K - \bar{e}_K|^m] < E[|\hat{e}_1 - \bar{e}_1|^m],$$

where, unlike the previous theorem, we now have strict inequality.

Proof. Without loss of generality, we assume that all examples in our finite example space have positive probability so that every dataset has positive probability. Now, for a strictly convex function, such as $|x|^m$, $m \geq 2$, Jensen's inequality holds with equality if and only if all the terms x_i are equal. Substituting $x_i = \hat{e}_i - \bar{e}_i$, we see that if $\hat{e}_i - \bar{e}_i \neq \hat{e}_j - \bar{e}_j$ for some dataset, then we are done. Otherwise, for contradiction, assume that

$$\hat{e}_i - \bar{e}_i = \hat{e}_j - \bar{e}_j, \text{ for all data sets, and } 1 \leq i, j \leq n. \quad (1)$$

Now, we consider several possible data sets. To describe these, let S_1 be a set of $\frac{n}{k} - 2$ examples, let S_2 be a set of $\frac{n}{k} - 1$ examples, and let $S_3, S_4, \ldots, S_k$ be sets of $\frac{n}{k}$ examples each. The basic idea is that we will be swapping the first element of the first fold with first element of the second fold. Specifically, the data sets we consider (using semicolons to separate the folds) are:

A. $z, x, S_1;\ z', S_2;\ S_3;\ S_4; \cdots$

B. $z', x, S_1;\ z, S_2;\ S_3;\ S_4; \cdots$

C. $z, y, S_1;\ z', S_2;\ S_3;\ S_4; \cdots$

D. $z', y, S_1;\ z, S_2;\ S_3;\ S_4; \cdots$

To distinguish between the hypotheses of different data sets, we'll refer to the errors by their letters, e.g. $\bar{e}_{Bi}$ refers to the true error of the hypothesis h_{Bi} trained on everything but the ith fold in dataset B.

By the assumption of insensitivity to example order, we see that $\hat{e}_{A3} - \bar{e}_{A3} = \hat{e}_{B3} - \bar{e}_{B3}$. By (1), we see that $\hat{e}_{A1} - \bar{e}_{A1} = \hat{e}_{B1} - \bar{e}_{B1}$. Similarly, insensitivity to example ordering implies that $\hat{e}_{C3} - \bar{e}_{C3} = \hat{e}_{D3} - \bar{e}_{D3}$ so we have $\hat{e}_{C1} - \bar{e}_{C1} = \hat{e}_{D1} - \bar{e}_{D1}$. Noting that $h_{A1} = h_{C1}$ and $h_{B1} = h_{D1}$, we subtract equations to get,

$$\begin{aligned} \hat{e}_{A1} - \bar{e}_{A1} - (\hat{e}_{C1} - \bar{e}_{C1}) &= \hat{e}_{B1} - \bar{e}_{B1} - (\hat{e}_{D1} - \bar{e}_{D1}) \\ \hat{e}_{A1} - \hat{e}_{C1} &= \hat{e}_{B1} - \hat{e}_{D1}. \end{aligned}$$

Now, again using the fact that $h_{A1} = h_{C1}$ and $h_{B1} = h_{D1}$ we have:

$$e_{A1}(x) - e_{A1}(y) = e_{B1}(x) - e_{B1}(y),$$

where $e_{A1}(x)$ denotes the error of h_{A1} on example x. Since this last equation holds for arbitrary z, z', and S_i, it means that changing a single training example (z to z') does not change the quantity $e(x) - e(y)$. Therefore, $e_h(x) - e_h(y)$ must be the same for any training set, because one training set can be changed to any other by a sequence of individual changes. Since this is also true for arbitrary y, this means the the function $f(x, y) = e_h(x) - e_h(y)$ is well-defined (i.e., it doesn't depend on the training data). In particular, we see that $e_h(x) - \bar{e}_h = E_{y \in D}[e_h(x) - e_h(y)]$ is a constant quantity across training sets for h.

This strict requirement that $e_h(x) - \bar{e}_h$ is constant leads us to conclude that $e_h(x) = e_h(y)$ always. To see this, consider the following data set:

E. $x, x, \ldots, x;\ y, y, \ldots, y;\ S_3;\ S_4;\ \cdots$

By applying (1) to data set E, we see that

$$\hat{e}_{E1} - \bar{e}_{E1} = e_{E1}(x) - \bar{e}_{E1} = e_{E2}(y) - \bar{e}_{E2}.$$

But, from the previous paragraph, we know these differences do not depend on the specific training data. Thus, $e_{E1}(x) - \bar{e}_{E1} = e_{E1}(y) - \bar{e}_{E1}$, $e_{E1}(x) = e_{E1}(y)$, and $e_h(x) = e_h(y)$ for any h learned from training data. This implies all individual fold error estimates are perfectly accurate, violating $Pr[\hat{e}_1 \neq \bar{e}_1] > 0$. ∎

It is interesting to consider when the k-fold estimate will be much better than the hold-out. It is sufficient that $\hat{e}_i - \bar{e}_i$ have a significant chance of different than $\hat{e}_j - \bar{e}_j$, i.e. that these variables are not completely correlated. One scenario in which this is the case is when you have a form of hypothesis stability, which could guarantee that $\bar{e}_j$ is close to $\bar{e}_i$.

Finally, we show a worst-case type of result, that Hoeffding bounds can still be used for the k-fold estimate, as if we had just a hold-out of size n/k:

Theorem 3 *Hoeffding bounds hold as if we used* n/k *testing examples. In particular,*

$$Pr[\hat{e}_K > \bar{e}_K + a] \leq e^{-2a^2 n/k} \text{ and } Pr[\hat{e}_K < \bar{e}_K - a] \leq e^{-2a^2 n/k}.$$

Proof (sketch). The proof of Hoeffding bounds for the standard hold-out case of $\hat{e}_1$ and $\bar{e}_1$ with a hold-out set of size $s = n/k$, e.g. [1], begins by bounding $E[e^{\lambda s(\hat{e}_1 - \bar{e}_1)}]$. Then they use Markov's inequality with this bound,

$$Pr[\hat{e}_1 > \bar{e}_1 + a] = Pr[e^{\lambda s(\hat{e}_1 - \bar{e}_1)} > e^{\lambda a}] \leq \frac{E[e^{\lambda s(\hat{e}_1 - \bar{e}_1)}]}{e^{\lambda s a}}.$$

However, since $e^{\lambda s x}$ is a convex function of x, Jensen's inequality implies that,

$$\begin{aligned} e^{\lambda s(\hat{e}_K - \bar{e}_K)} &= e^{\frac{\lambda s}{k}(\hat{e}_1 - \bar{e}_1 + \cdots + \hat{e}_k - \bar{e}_k)} \\ &\leq \frac{e^{\lambda s(\hat{e}_1 - \bar{e}_1)} + \cdots + e^{\lambda s(\hat{e}_k - \bar{e}_k)}}{k}. \end{aligned}$$

Thus $E[e^{\lambda(\hat{e}_K - \bar{e}_K)}] \leq E[e^{\lambda(\hat{e}_1 - \bar{e}_1)}]$, and the proof goes through. ∎

4 PROGRESSIVE VALIDATION ANALYSIS

Again, we suppose we have a data set of size n. This time we break it into two sets, a training set and a testing set, with the test set having s elements. In this section, we redefine h_i, $\hat{e}_i$, and $\bar{e}_i$. Hypothesis h_i is generated by training on the training set and the first $i-1$ elements of the testing set. It is tested on the ith element of the testing set to yield an estimate $\hat{e}_i$ of its true error, $\bar{e}_i$. The progressive hypothesis chooses randomly among the s hypotheses to label an example. Thus it has true error $\bar{e}_{\mathrm{P}}$, with

$$\bar{e}_{\mathrm{P}} = \frac{\bar{e}_1 + \bar{e}_2 + \cdots + \bar{e}_s}{s}.$$

Finally, we let the progressive error estimate be the average $\hat{e}_{\mathrm{P}} = (\hat{e}_1 + \hat{e}_2 + \cdots + \hat{e}_s)/s$.

We would like to show that the progressive error with a hold-out of size s is as good an estimate of the true error of the progressive hypothesis as the hold-out error is of the hold-out hypothesis. First we show that the same Hoeffding bounds apply:

Theorem 4 *Hoeffding bounds hold as if we used a hold-out set of size s. In particular,*

$$Pr[\hat{e}_P > \bar{e}_P + a] \leq e^{-2a^2 s} \text{ and } Pr[\hat{e}_P < \bar{e}_P - a] \leq e^{-2a^2 s}.$$

Littlestone [9], gives a quite detailed proof of the multiplicative (Chernoff-style) version of this theorem. The sketch below uses the same basic argument.

Proof (sketch). As before, we only need to make a slight modification to the standard proof of Hoeffding bounds. The standard proof [1] begins by bounding $E[e^{\lambda s(\hat{e}_{\mathrm{P}} - \bar{e}_{\mathrm{P}})}]$. This bound is achieved by writing $e^{\lambda s(\hat{e}_{\mathrm{P}} - \bar{e}_{\mathrm{P}})}$ as a product of s terms $e^{\lambda Y_i}$, with

$$E[e^{\lambda Y_i}] \leq e^{\lambda^2/8}. \tag{2}$$

The Y_i's, in the standard proof are independent variables, perhaps coin flips, but are adjusted so that they each have mean 0. In our setting, Y_i corresponds to $\hat{e}_i - \bar{e}_i$, the error discrepancy of the ith hypothesis with the ith measurement. In the ordinary setting these Y_i's are independent because the ith hypothesis doesn't depend on any previous data in the hold-out. In our setting, they are not independent. But, while the previous data in the hold-out may not be independent of the hypothesis, it is independent of the ith hold-out example. Since (2) holds regardless of the hypothesis, we still have that $E[e^{\lambda Y_i} | Y_1, Y_2, \ldots, Y_{i-1}] \leq e^{\lambda^2/8}$.

Now, for two non-negative random variables A and B, with $E[A] \leq c_1$ and $E[B|A] \leq c_2$, it is true that $E[AB] \leq E[Ac_2] \leq c_1 c_2$. Thus, by induction, even though the Y_i's aren't independent, $E[e^{\lambda s(\hat{e}_{\mathrm{P}} - \bar{e}_{\mathrm{P}})}] = E[\prod e^{\lambda Y_i}] \leq e^{\lambda^2 s/8}$, which is all that is needed for the proof. ∎

In fact, if we consider just the variance (the second moment) we can make a stronger statement. In particular, the variance of the progressive validation estimate, with respect to the true error of the progressive validation hypothesis, is no worse than the variance of an estimate produced by testing the progressive validation hypothesis on a new, extra, hold-out of size s.

Theorem 5 *Let $\hat{e}_P{}'$ be an estimate of the progressive validation hypothesis's error measured on a new, independently chosen hold-out of size s. Then,*

$$E[(\hat{e}_P - \bar{e}_P)^2] \leq E[(\hat{e}_P' - \bar{e}_P)^2].$$

Proof. Both quantities above are averages of s terms. The RHS is the variance of the sum of independent terms, which is the sum of the variances. Each of these i.i.d. terms has a $1/s$ chance of being distributed like $\hat{e}_i$, for each i. Thus the RHS is

$$\frac{E[\sum_{i=1}^{s}(\hat{e}_i - \bar{e}_{\mathrm{P}})^2]}{s^2} = \frac{E[\sum_{i=1}^{s}(\hat{e}_i^2 - \bar{e}_{\mathrm{P}}^2)]}{s^2}.$$

The LHS is $E[(\hat{e}_{\mathrm{P}} - \bar{e}_{\mathrm{P}})^2] = E[(\sum(\hat{e}_i - \bar{e}_i)^2]/s^2$. We would like to again use the fact that the variance of a sum of independent terms is the sum of variances. While these terms are not independent, we do have the martingale-like property that $E[\hat{e}_j - \bar{e}_j | \hat{e}_i - \bar{e}_i] = 0$ for $i < j$. Now, $E[A|B] = 0 \implies E[AB] = 0$, so that $E[(\hat{e}_j - \bar{e}_j)(\hat{e}_i - \bar{e}_i)] = 0$ for $i \neq j$. This means that even though the terms aren't independent, the variance of the sum is still the sum of the variances. Thus the LHS is,

$$\frac{E[\sum(\hat{e}_i - \bar{e}_i)^2]}{s^2} = \frac{E[\sum(\hat{e}_i^2 - \bar{e}_i^2)]}{s^2}.$$

Thus the LHS and RHS are quite similar. They only differ in that the RHS has $\bar{e}_{\mathrm{P}}$'s and the LHS has $\bar{e}_i$'s. Because $(\bar{e}_1^2 + \cdots + \bar{e}_s^2)/s \geq ((\bar{e}_1 + \cdots + \bar{e}_s)/s)^2 = \bar{e}_{\mathrm{P}}^2$, we're done. Unfortunately, this argument does not work on the higher moments. ∎

5 EXPERIMENTS

The motivation behind progressive validation is that it allows one to train on more examples than the hold-out estimate. With the extra examples training algorithms should be able to choose a better hypothesis. Many learning problems exhibit thresholding where a small increase in the number of examples dramatically improves the accuracy of the hypothesis. Consider an N dimensional feature space in the boolean setting where it is known that one feature is an exact predictor. Consider the learning algorithm: cross off features inconsistent with the training data and output the hypothesis that takes a majority vote over all features remaining. If the example distribution is uniform over $\{0,1\}^N$, then this example exhibits a thresholding behavior because the accuracy of the current hypothesis is almost 50% until the number of consistent features is reduced to a constant, at which point it quickly increases to 100%. In expectation, $\frac{1}{2}$ of the features will be eliminated with each example, leading us to expect a threshold near $\lg N$.

In our experiments, we built a synthetic data generator which picks a feature uniformly at random then produces some number of correctly-labeled examples consisting of $N = 1000$ boolean features, with $Pr(\text{true}) = .5$. The output of this generator was given to the learning algorithm.

In the first test, we trained on $n - 10$ examples and tested on 10 examples. In the second test, we trained on $n - 10$ examples and applied progressive validation to the next 10

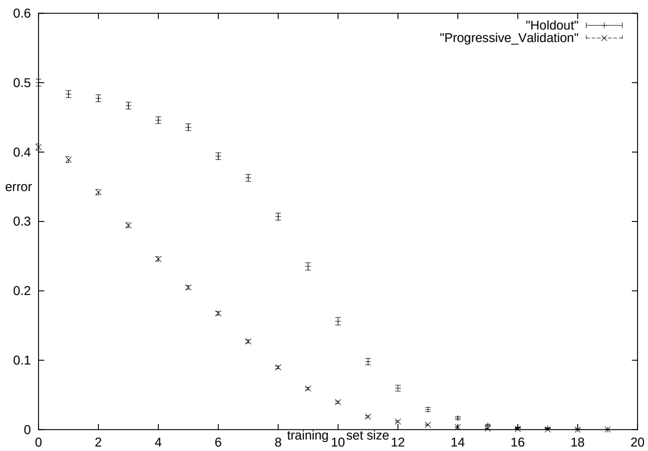

Figure 1: True error vs. training size for hold-out and progressive validation

examples. We repeated this experiment 1000 times for $10 \leq n \leq 30$ and averaged the results in order to get an empirical estimate of the true error of all hypotheses produced, shown in Figure 1. Error bars in the figure are at one standard deviation.

As expected, the hold-out's performance was much worse than that of progressive validation. In general, the degree of improvement in empirical error due to the progressive validation depends on the learning algorithm. The improvement can be large if the data set is small or the learning problem exhibits thresholding behavior at some point past the number of training examples.

In order to compare the quality of error estimation, we did another set of runs calculating the error discrepancy |true error$-$estimated error|. Five training examples were used followed by either progressive validation on ten examples or evaluation on a hold-out set of size ten. The "true error" was calculated empirically by evaluating the resulting hypothesis for each case on another hold-out set of 10000 examples. The hold-out estimate on five examples has larger variance then the progressive validation estimate. One might suspect that this is not due to a good estimation procedure but due to the fact that it is easier to estimate a lower error. To investigate this further, we performed a hold-out test which was trained on nine examples, because the true error of the progressive validation hypothesis with five training examples and ten progressive validation examples was close to the true error of a hypothesis trained on nine examples, as shown in the following table:

	true error	|true error $-$ est.|
Prog. Val. $(5, 10)$	$.205 \pm .003$	$.088 \pm .011$
Hold-out $(5, 10)$	$.436 \pm .005$	$.120 \pm .015$
Hold-out $(9, 10)$	$.235 \pm .005$	$.109 \pm .015$

Averages of the true error and estimate accuracy favor progressive validation in this experiment with a hold-out set of size 10. In fact, the progressive estimate and hypothesis on a data set of size 15 were better than the hold-out estimate and hypothesis on a data set of size 19.

6 RELATED WORK, FUTURE WORK, AND CONCLUSIONS

Leave-one-out cross-validation, which is also common in practice, corresponds to $k = n$, and the bounds [8] depend on the VC dimension and hypothesis stability. Restrictions of some kind seem unavoidable, as there are interesting examples of situations where the leave-one-out estimate is always off by 50% [8]. These terrible-case examples do not exist for k-fold cross-validation with small k, because it is better than a hold-out set of a reasonable size, which is a good estimator. In addition, certain algorithms, such as nearest neighbor, have been shown to have good performance with leave-one-out [4]. Our bounds, however, are not very informative in the leave-one-out case, because we would be comparing it to a hold-out of a single element.

Anthony and Holden [2] extend the analysis of Kearns and Ron [8] to the k-fold setting. They judge the k-fold error as an estimate of the true error of the hypothesis trained on all the data. This is a natural formulation of the problem, because in practice the hypothesis often chosen is this untested hypothesis. However, because the new hypothesis is untested, their performance guarantees depend on VC dimension, and their results are sanity-check bounds which relate the k-fold error to the training error. For large k, leaving a small number out, the training error may be a better estimate than the corresponding hold-out, and their bounds may bridge the gap between leave-one-out ($k = n$) and typical k-fold (k is a small constant).

On another note, if the k-fold hypothesis is chosen as an average of the k generated hypotheses rather than the randomizing hypothesis, it is similar to bagging[3]. In that situation, the goal is to reduce the generalization error, which Breiman claims can be achieved by reducing the variance in the hypothesis. On the other hand, we are concerned more with the variance in our error discrepancy. Thus decreasing the generalization error of the final hypothesis would make the k-fold error a worse estimate. It would also be interesting to explore the connection between hypothesis instability, which Breiman discusses for the purposes of reducing generalization error, to hypothesis stability, which Kearns and Ron [8] trace back to Devroye and Wagner [5] for the purposes of accurate error estimation.

In conclusion, we have shown that the k-fold estimate of generalization error is better than testing on a hold-out of $1/k$ of the data. In future work, it would be nice to analyze how much better the k-fold estimate is. We have also introduced progressive validation. We provide theoretical and experimental evidence that it does not reduce our error estimate accuracy, while providing more examples for training than a simple hold-out set.

References

[1] N. Alon and J. Spencer. *The Probabilistic Method.* Wiley, 1991.

[2] M. Anthony and S. B. Holden Cross-Validation for Binary Classification by Real-Valued Functions: Theoret-

ical Analysis In *Proc. Eleventh Annual Conference on Computational Learning Theory*, 1998.

[3] L. Brieman. Bagging predictors. *Machine Learning*, 24(2):123-140, 1996.

[4] L. Devroye, L. Gyrofi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer-Verlag, 1996.

[5] L. Devroye and T. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IT-25(5):601-604, 1979.

[6] Y. Freund and R. Shapire. Discussion of the paper ”Arcing classifiers” by Leo Breiman. *Annals of Statistics*, 26(3): 824-832, 1998.

[7] D.P. Helmbold and M.K. Warmuth. On Weak Learning. *JCSS*, 50(3): 551-573, 1995.

[8] M. J. Kearns and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. In *Proc. Tenth Annual Conference on Computational Learning Theory*, 1997.

[9] N. Littlestone. From on-line to batch learning. In *Proceedings of the 2nd Annual Workshop on Computational Learning Theory*, pp. 269–284, 1989.