Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation

Michael Kearns
AT&T Labs Research, Florham Park, NJ 07932, U.S.A.

Dana Ron
Department of EE—Systems, Tel Aviv University, 69978 Ramat Aviv, Israel

In this article we prove sanity-check bounds for the error of the leave-one-out cross-validation estimate of the generalization error: that is, bounds showing that the worst-case error of this estimate is not much worse than that of the training error estimate. The name sanity check refers to the fact that although we often expect the leave-one-out estimate to perform considerably better than the training error estimate, we are here only seeking assurance that its performance will not be considerably worse. Perhaps surprisingly, such assurance has been given only for limited cases in the prior literature on cross-validation.

Any nontrivial bound on the error of leave-one-out must rely on some notion of algorithmic stability. Previous bounds relied on the rather strong notion of hypothesis stability, whose application was primarily limited to nearest-neighbor and other local algorithms. Here we introduce the new and weaker notion of error stability and apply it to obtain sanity-check bounds for leave-one-out for other classes of learning algorithms, including training error minimization procedures and Bayesian algorithms. We also provide lower bounds demonstrating the necessity of some form of error stability for proving bounds on the error of the leave-one-out estimate, and the fact that for training error minimization algorithms, in the worst case such bounds must still depend on the Vapnik-Chervonenkis dimension of the hypothesis class.

1 Introduction and Motivation

A fundamental problem in statistics, machine learning, neural networks, and related areas is that of obtaining an accurate estimate for the generalization ability of a learning algorithm trained on a finite data set. Many estimates have been proposed and examined in the literature, some of the most prominent being the training error (also known as the resubstitution estimate), the various cross-validation estimates (which include the leave-one-out or deleted estimate, as well as $k$-fold cross-validation), and the hold-out estimate. For each of these estimates, the hope is that for a fairly wide
class of learning algorithms, the estimate will usually produce a value \( \hat{\epsilon} \) that is close to the true (generalization) error \( \epsilon \).

There are surprisingly few previous results providing bounds on the accuracy of the various estimates (Rogers & Wagner, 1978; Devroye & Wagner, 1979a, b; Vapnik, 1982; Holden, 1996a, b; Kearns, Mansour, Ng, & Ron, 1995; Kearns, 1996) (see Devroye, Gyorfi, & Lugosi, 1996, for an excellent introduction and survey of the topic). Perhaps the most general results are those given for the (classification) training error estimate by Vapnik (1982), who proved that for any target function and input distribution, and for any learning algorithm that chooses its hypotheses from a class of VC dimension \( d \), the training error estimate is at most \( O(\sqrt{d/m}) \) away from the true error, where \( m \) is the size of the training sample. On the other hand, among the strongest bounds (in the sense of the quality of the estimate) are those given for the leave-one-out estimate by the work of Rogers and Wagner (1978), Devroye and Wagner (1979a, b), and Vapnik (1982).

The (classification error) leave-one-out estimate is computed by running the learning algorithm \( m \) times, each time removing one of the \( m \) training examples, and testing the resulting hypothesis on the training example that was deleted; the fraction of failed tests is the leave-one-out estimate. Rogers and Wagner (1978) and Devroye and Wagner (1979a, b) proved that for several specific algorithms, but again for any target function and input distribution, the leave-one-out estimate can be as close as \( O(1/\sqrt{m}) \) to the true error. The algorithms they consider are primarily variants of nearest-neighbor and other local procedures, and as such do not draw their hypotheses from a fixed class of bounded VC dimension, which is the situation we are primarily interested in here. Devroye et al. (1996) obtain a bound on the error of the leave-one-out estimate for another particular class of algorithms: that of histogram rules. Vapnik (1982) studies the leave-one-out estimate (which he refers to as the moving-control estimate) for a special case of linear regression. He proves bounds of order \( 1/\sqrt{m} \) on the error of the estimate under certain assumptions on the distribution over the examples, and their labels.

A tempting and optimistic intuition about the leave-one-out estimate is that it should typically yield an estimate that falls within \( O(1/\sqrt{m}) \) of the true error. This intuition derives from viewing each deleted test as an independent trial of the true error. The problem, of course, is that these tests are not independent. The results of Rogers and Wagner (1978) and Devroye and Wagner (1979a, b) demonstrate that for certain algorithms, the intuition is essentially correct despite the dependencies. In such cases, the leave-one-out estimate may be vastly preferable to the training error, yielding an estimate of the true error whose accuracy is independent of any

---

1 The \( \hat{O}(\cdot) \) notation hides logarithmic factors in the same way that \( O(\cdot) \) notation hides constants.
notion of dimension or hypothesis complexity (even though the true error itself may depend strongly on such quantities).

Despite such optimism, the prior literature leaves open a disturbing possibility for the leave-one-out proponent: that its accuracy may often be, for wide classes of natural algorithms, arbitrarily poor. We would like to have what we shall informally refer to as a sanity-check bound: a proof, for large classes of algorithms, that the error of the leave-one-out estimate is not much worse than the $\tilde{O}(\sqrt{d/m})$ worst-case behavior of the training error estimate. The name sanity check refers to the fact that although we believe that under many circumstances, the leave-one-out estimate will perform much better than the training error (and thus justify its computational expense) the goal of the sanity-check bound is simply to prove that it is not much worse than the training error. Such a result is of interest simply because the leave-one-out estimate is in wide experimental use (largely because practitioners expect it to outperform the training error frequently), so it behooves us to understand its performance and limitations.

A moment’s reflection should make it intuitively clear that in contrast to the training error, even a sanity-check bound for leave-one-out cannot come without restrictions on the algorithm under consideration: some form of algorithmic stability is required (Devroye & Wagner, 1979b; Holden, 1996b; Kohavi, 1995). If the removal of even a single example from the training sample may cause the learning algorithm to “jump” to a different hypothesis with, say, much larger error than the full-sample hypothesis, it seems hard to expect the leave-one-out estimate to be accurate. The precise nature of the required form of stability is less obvious.

Devroye and Wagner (1979b) first identified a rather strong notion of algorithmic stability that we shall refer to as hypothesis stability, and showed that bounds on hypothesis stability directly lead to bounds on the error of the leave-one-out estimate. This notion of stability demands that the removal of a single example from the training sample results in hypotheses that are “close” to each other, in the sense of having small symmetric difference with respect to the input distribution. For algorithms drawing hypotheses from a class of fixed VC-dimension, the first sanity-check bounds for the leave-one-out estimate were provided by Holden (1996b) for two specific algorithms in the realizable case (that is, when the target function is actually contained in the class of hypothesis functions).

However, in the more realistic unrealizable (or agnostic; Kearns, Schapire, & Sellie, 1994) case, the notion of hypothesis stability may simply be too strong to be obeyed by many natural learning algorithms. For example, if there are many local minima of the true error, an algorithm that managed always to minimize the training error might be induced to move to a rather distant hypothesis by the addition of a new training example (we shall elaborate on this example shortly). Many gradient descent procedures use randomized starting points, which may even cause runs on the same sample
to end in different local minima. Algorithms behaving according to Bayesian principles will choose two hypotheses of equal training error with equal probability, regardless of their dissimilarity. What we might hope remains relatively stable in such cases would not be the algorithm’s hypothesis itself but the error of the algorithm’s hypothesis.

The primary goal of this article is to give sanity-check bounds for the leave-one-out estimate that are based on the error stability of the algorithm. In section 2, we begin by stating some needed preliminaries. In section 3, we review the Devroye and Wagner notion of hypothesis stability, and generalize the results of Holden (1996b) by showing that in the realizable case, this notion can be used to obtain sanity-check bounds for any consistent learning algorithm, but we also discuss the limitations of hypothesis stability in the unrealizable case. In section 4, we define our new notion of error stability and prove our main results: bounds on the error of the leave-one-out estimate that depend on the VC-dimension of the hypothesis class and the error stability of the algorithm. The bounds apply to a wide class of algorithms meeting a mild condition that includes training error minimization and Bayesian procedures. Although we concentrate on boolean functions, we also discuss real-valued functions (in section 4.3). In section 5, we give a number of lower bound results showing, among other things, the necessity of some form of error stability for proving bounds on the error of the leave-one-out estimate, but also the absence of sufficiency of error stability (thus justifying the need of an additional condition). In section 6 we conclude with some open problems.

2 Preliminaries

Let \( f \) be a fixed target function from domain \( X \) to range \( Y \), and let \( P \) be a fixed distribution over \( X \). Both \( f \) and \( P \) may be arbitrary. We use \( S_m \) to denote the random variable \( S_m = (x_1, y_1), \ldots, (x_m, y_m) \), where \( m \) is the sample size, each \( x_i \) is drawn randomly and independently according to \( P \), and \( y_i = f(x_i) \). A learning algorithm \( A \) is given \( S_m \) as input, and outputs a hypothesis \( h = A(S_m) \), where \( h: X \rightarrow Y \) belongs to a fixed hypothesis class \( H \). If \( A \) is randomized, it takes an additional input \( r \in [0, 1]^k \) of random bits of the required length \( k \) to make its random choices. In this article we study mainly the case in which \( Y = \{0, 1\} \), and briefly the case in which \( Y = \mathbb{R} \). For now we restrict our attention to boolean functions.

For any boolean function \( h \), we define the generalization error of \( h \) (with respect to \( f \) and \( P \)) by

\[
\epsilon(h) = \epsilon_{f, P}(h) = \frac{1}{m} \sum_{x \in P} [h(x) \neq f(x)].
\]  

---

2 Our results directly generalize to the case in which we allow the target process to be any joint distribution over the sample space \( X \times Y \), but it will be convenient to think of there being a distinct target function.
For any two boolean functions $h$ and $h'$, the distance between $h$ and $h'$ (with respect to $P$) is

$$\text{dist}(h, h') = \text{dist}_P(h, h') \stackrel{\text{def}}{=} \Pr_{x \in P}[h(x) \neq h'(x)].$$

Since the target function $f$ may or may not belong to $H$, we define

$$\epsilon_{\text{opt}} \stackrel{\text{def}}{=} \min_{h \in H} \epsilon(h)$$

and let $h_{\text{opt}}$ be some function for which $\epsilon(h_{\text{opt}}) = \epsilon_{\text{opt}}$. Thus, the function $h_{\text{opt}}$ is a best approximation to $f$ (with respect to $P$) in the class $H$, and $\epsilon_{\text{opt}}$ measures the quality of this approximation. We define the training error of a boolean function $h$ with respect to $S_m$ by

$$\hat{\epsilon}(h) = \hat{\epsilon}_{S_m}(h) \stackrel{\text{def}}{=} \frac{1}{m} \cdot \left| \left\{ (x_i, y_i) \in S_m : h(x_i) \neq y_i \right\} \right|$$

and the (generalized) version space

$$VS(S_m) \stackrel{\text{def}}{=} \left\{ h \in H : \hat{\epsilon}(h) = \min_{h' \in H} \hat{\epsilon}(h') \right\},$$

consisting of all functions in $H$ that minimize the training error.

Throughout this article we assume that the algorithm $A$ is symmetric. This means that $A$ is insensitive to the ordering of the examples in the input sample $S_m$, so for every ordering of $S_m$ it outputs the same hypothesis. (In case $A$ is randomized, it should induce the same distribution on hypotheses.) This is a very mild assumption, as any algorithm can be transformed into a symmetric algorithm by adding a randomizing preprocessing step. Thus, we may refer to $S_m$ as an unordered set of labeled examples rather than as a list of examples. For any index $i \in [m]$, we denote by $S'_m$ the sample $S_m$ with the $i$th labeled example, $(x_i, y_i)$, removed. That is,

$$S'_m \stackrel{\text{def}}{=} S_m \setminus \{(x_i, y_i)\}.$$

The leave-one-out cross-validation estimate, $\hat{\epsilon}_{\text{cv}}(S_m)$, of the error of the hypothesis $h = A(S_m)$ is defined to be

$$\hat{\epsilon}_{\text{cv}}(S_m) \stackrel{\text{def}}{=} \frac{1}{m} \cdot \left| \left\{ i \in [m] : h'(x_i) \neq y_i \right\} \right|$$

where $h' = A(S'_m)$. We are thus interested in providing bounds on the error $|\hat{\epsilon}_{\text{cv}}(S_m) - \epsilon(A(S_m))|$ of the leave-one-out estimate.

The following uniform convergence bound, due to Vapnik (1982), will be central to this article.
Theorem 1. Let $H$ be a hypothesis class with VC-dimension $d < m$. Then for every $m > 4$ and for any given $\delta > 0$, with probability at least $1 - \delta$, for every $h \in H$,

$$|\hat{\epsilon}(h) - \epsilon(h)| < 2 \sqrt{\frac{d \left( \ln(2m/d) + 1 \right) + \ln(9/\delta)}{m}}. \quad (2.8)$$

Let us introduce the shorthand notation,

$$VC(d, m, \delta) \stackrel{\text{def}}{=} 2 \sqrt{\frac{d \left( \ln(2m/d) + 1 \right) + \ln(9/\delta)}{m}}. \quad (2.9)$$

Thus, theorem 1 says that for any learning algorithm $A$ using a hypothesis space of VC-dimension $d$, for any $\delta > 0$, with probability at least $1 - \delta$ over $S_m$, $|\hat{\epsilon}(A(S_m)) - \epsilon(A(S_m))| < VC(d, m, \delta)$.

3 Sanity-Check Bounds via Hypothesis Stability

It is intuitively clear that the performance of the leave-one-out estimate must rely on some kind of algorithmic stability (this intuition will be formalized in the lower bounds of section 5). Perhaps the strongest notion of stability that an interesting learning algorithm might be expected to obey is that of hypothesis stability: that small changes in the sample can only cause the algorithm to move to “nearby” hypotheses. The notion of hypothesis stability is due to Devroye and Wagner (1979b), and is formalized in a way that suits our purposes in the following definition.\(^3\)

Definition 1. We say that an algorithm $A$ has hypothesis stability $(\beta_1, \beta_2)$ if for every $m$

$$\Pr_{S_{m-1}, (x,y)} [\text{dist}(A(S_m), A(S_{m-1})) \geq \beta_2] \leq \beta_1, \quad (3.1)$$

where $S_m = S_{m-1} \cup \{(x, y)\}$, and both $\beta_1$ and $\beta_2$ may be functions of $m$.

Thus, we ask that with high probability, the hypotheses output by $A$ on $S_m$ and $S_{m-1}$ be similar.

We shall shortly argue that hypothesis stability is in fact too demanding a notion in many realistic situations. But first we state the elegant theorem of Devroye and Wagner (1979b) that relates the error of the leave-one-out estimate for an algorithm to the hypothesis stability.

\(^3\) Devroye and Wagner (1979b) formalized hypothesis stability in terms of the expected difference between the hypotheses; here we translate to the “high-probability” form for consistency.
Theorem 2. Let $A$ be any symmetric algorithm that has hypothesis stability $(\beta_1, \beta_2)$. Then for any $\delta > 0$, with probability at least $1 - \delta$ over $S_m$,

$$|\hat{\epsilon}_c^A(S_m) - \epsilon(A(S_m))| \leq \sqrt{\frac{1/(2m) + 3(\beta_1 + \beta_2)}{\delta}}.$$  

(3.2)

Thus, if we are fortunate enough to have an algorithm with strong hypothesis stability (that is, small $\beta_1$ and $\beta_2$), the leave-one-out estimate for this algorithm will be correspondingly accurate. What kind of hypothesis stability should we expect for natural algorithms? Devroye and Wagner (1979b) and Rogers and Wagner (1978) gave rather strong hypothesis stability results for certain nonparametric local learning algorithms (such as nearest-neighbor rules), and thus were able to show that the error of the leave-one-out estimate for such algorithms decreases like $1/m^a$ (for values of $a$ ranging from $1/4$ to $1/2$, depending on the details of the algorithm).

Note that for nearest-neighbor algorithms, there is no fixed “hypothesis class” of limited VC dimension; the algorithm may choose arbitrarily complex hypotheses. This unlimited complexity often makes it difficult to quantify the performance of the learning algorithm except in terms of the asymptotic generalization error (see Devroye et al., 1996, for a detailed survey of results for nearest-neighbor algorithms). For this and other reasons, practitioners often prefer to commit to a hypothesis class $H$ of fixed VC-dimension $d$ and use heuristics to find a good function in $H$. In this case, we gain the possibility of finite-sample generalization error bounds (where we compare the error to that of the optimal model from $H$). However, in such a situation, the goal of hypothesis stability may in fact be at odds with the goal of good performance in the sense of learning. To see this, imagine that the input distribution and target function define a generalization error “surface” over the function space $H$ and that this surface has minima at $h_{\text{opt}} \in H$, where $\epsilon(h_{\text{opt}}) = \epsilon_{\text{opt}} > 0$, and also at $h' \in H$, where $\epsilon(h') = \epsilon(h_{\text{opt}}) + \alpha$ for some small $\alpha > 0$. Thus, $h_{\text{opt}}$ is the “global” minimum, and $h'$ is a “local” minimum. Note that $\text{dist}(h_{\text{opt}}, h')$ could be as large as $2\epsilon_{\text{opt}}$, which we are assuming may be a rather large (constant) quantity. Now if the algorithm $A$ minimizes the training error over $H$, then we expect that as $m \to \infty$, algorithm $A$ will settle on hypotheses closer and closer to $h_{\text{opt}}$. But for $m \ll 1/\alpha$, $A$ may well choose hypotheses close to $h'$. Thus, as more examples are seen, at some point $A$ may need to move from $h'$ to the rather distant $h_{\text{opt}}$.

We do not know how to rule out such behavior for training error minimization algorithms, and so cannot apply theorem 2. Perhaps more important, for certain natural classes of algorithms (such as the Bayesian algorithms discussed later) and for popular heuristics such as C4.5 and backpropagation, it is far from obvious that any nontrivial statement about hypothesis stability can be made. For this reason, we would like to have bounds on the error of the leave-one-out estimate that rely on the weakest possible notion of stability. Note that in the informal example given above,
the quantity that we might hope would exhibit some stability is not the hypothesis itself, but the error of the hypothesis: even though \( h_{\text{opt}} \) and \( h' \) may be far apart, if \( A \) chooses \( h' \), then \( \alpha \) must not be “too large.” The main question addressed in this article is when this weaker notion of error stability is sufficient to prove nontrivial bounds on the leave-one-out error; we turn to this in section 4.

First, however, note that the instability of the hypothesis in the above discussion relied on the assumption that \( \epsilon_{\text{opt}} > 0 \)—that is, that we are in the unrealizable setting. In the realizable \( \epsilon_{\text{opt}} = 0 \) case, there is still hope for applying hypothesis stability. Indeed, Holden (1996b) was the first to apply uniform convergence results to obtain sanity-check bounds for leave-one-out via hypothesis stability, for two particular (consistent) algorithms in the realizable setting. Here we generalize Holden’s results by giving a sanity-check bound on the leave-one-out error for any consistent algorithm. The simple proof idea again highlights why hypothesis stability seems difficult to apply in the unrealizable case: in the realizable case, minimizing the training error forces the hypothesis to be close to some fixed function (namely, the target). In the unrealizable case, there may be many different functions, all with optimal or near-optimal error.

**Theorem 3.** Let \( H \) be a class of VC-dimension \( d \), and let the target function \( f \) be contained in \( H \) (realizable case). Let \( A \) be a symmetric algorithm that always finds an \( h \in H \) consistent with the input sample. Then for every \( \delta > 0 \) and \( m > d \), with probability at least \( 1 - \delta \),

\[
|\hat{e}_s(S_m) - \epsilon(A(S_m))| = O\left(\sqrt{\frac{(d/m) \log(m/d)}{\delta}}\right).
\]

**Proof.** By uniform convergence, with probability at least \( 1 - \delta' \),

\[
\epsilon(A(S_m)) = \text{dist}(f, A(S_m)) = O\left(\frac{d \log(m/d) + \log(1/\delta')}m\right) \tag{3.4}
\]

and

\[
\epsilon(A(S_{m-1})) = \text{dist}(f, A(S_{m-1})) = O\left(\frac{d \log(m-1/d) + \log(1/\delta')}{m-1}\right). \tag{3.5}
\]

Holden (1996a) has recently obtained sanity-check bounds, again for the realizable setting, for other cross-validation estimates.
(Here we are using the stronger $O(d/m)$ uniform convergence bounds that are special to the realizable case.) Thus by the triangle inequality, with probability at least $1-\delta'$,

$$\text{dist}(A(S_m), A(S_{m-1})) = O\left(\frac{d \log(m/d) + \log(1/\delta')}{m}\right).$$

(3.6)

The theorem follows from theorem 2, where $\delta'$ is set to $d/m$.

We should note immediately that the bound of theorem 3 has a dependence on $\sqrt{1/\delta}$, as opposed to the $\log(1/\delta)$ dependence for the training error given by theorem 1. Unfortunately, it is well-known (Devroye et al., 1996, chap. 24) (and demonstrated in section 5) that at least in the unrealizable setting, a $1/\delta$ dependence is in general unavoidable for the leave-one-out estimate. Thus, it appears that in order to gain whatever benefits leave-one-out offers, we must accept a worst-case dependence on $\delta$ that is inferior to that of the training error. This again is the price of generality. For particular algorithms, such as $k$-nearest neighbor rules, it is possible to show only logarithmic dependence on $1/\delta$ (Rogers & Wagner, 1978) (stated in Devroye et al., 1996, theorem 24.2). Also, we note in passing that theorem 3 can also be generalized (perhaps with a worse power of $d/m$) to the case where the target function lies in $H$ but is corrupted by random classification noise. Again, minimizing training error forces the hypothesis to be close to the target.

It is possible to give examples in the realizable case for which the leave-one-out estimate has error $O(1/\sqrt{m})$ while the training error has error $\Omega(d/m)$; such examples merely reinforce the intuition discussed in the introduction that leave-one-out may often be superior to the training error. Furthermore, there are unrealizable examples for which the error of leave-one-out is again independent of $d$, but for which no nontrivial leave-one-out bound can be obtained by appealing to hypothesis stability. It seems that a more general notion of stability is called for.

4 Sanity-Check Bounds via Error Stability

In this section, we introduce the notion of error stability and use it to prove our main results. We give bounds on the error of the leave-one-out estimate that are analogous to those given in theorem 2, in that the quality of the bounds is directly related to the error stability of the algorithm. However, unlike theorem 2, in all of our bounds there will be a residual $\tilde{O}(\sqrt{d/m})$ term that appears regardless of the stability; this is the price we pay for using a weaker, but more widely applicable, type of stability. In section 5, we will show that some form of error stability (which is slightly weaker than the
one defined below) is always necessary,\textsuperscript{5} and also that a dependence on $d/m$ cannot be removed in the case of algorithms that minimize the training error, without further assumptions on the algorithm.

For expository purposes, we limit our attention to deterministic algorithms for now. The generalization to randomized algorithms will be discussed shortly. Our key definition mirrors the form of definition 1.

**Definition 2.** We say that a deterministic algorithm $A$ has error stability $(\beta_1, \beta_2)$ if for every $m$

\[
\Pr_{S_{m-1}, (x,y)} \left[ |\epsilon(A(S_m)) - \epsilon(A(S_{m-1}))| \geq \beta_2 \right] \leq \beta_1
\]

where $S_m = S_{m-1} \cup \{(x, y)\}$, and both $\beta_1$ and $\beta_2$ may be functions of $m$.

Thus, we ask that with high probability, the hypotheses output by $A$ on $S_m$ and $S_{m-1}$ have similar error with respect to the target, while allowing them to differ from each other.

Our goal is thus to prove bounds on the error of the leave-one-out estimate that depend on $\beta_1$ and $\beta_2$. This will require an additional (and hopefully mild) assumption on the algorithm that is quantified by the following definition. We will shortly prove that some natural classes of algorithms do indeed meet this assumption, thus allowing us to prove sanity-check bounds for these classes.

**Definition 3.** For any deterministic algorithm $A$, we say that leave-one-out $(\gamma_1, \gamma_2)$ overestimates the training error for $A$ if for every $m$,

\[
\Pr_{S_{m-1}, (x,y)} \left[ \hat{\epsilon}^A(S_m) \leq \hat{\epsilon}(A(S_m)) - \gamma_2 \right] \leq \gamma_1
\]

where $S_m = S_{m-1} \cup \{(x, y)\}$, and both $\gamma_1$ and $\gamma_2$ may be functions of $m$.

Although we cannot claim that training error overestimation is in general necessary for obtaining bounds on the error of the leave-one-out estimate, we note that it is clearly necessary whenever the training error underestimates the true error, as is the case for algorithms that minimize the training error. In any case, in section 5 we show that some additional assumptions (beyond error stability) are required to obtain nontrivial bounds for the error of leave-one-out.

\textsuperscript{5} As we note in section 5, this lower bound also implies that for any reasonable learning algorithm (such that the probability that the error of its hypothesis increases when a sample point is added is very small) the notion of error stability we define below is in fact necessary.
Before stating the main theorem of this section, we give the following simple but important lemma. This result is well known (Devroye et al., 1996, Chap. 24), but we include its proof for the sake of completeness.

**Lemma 1.** For any symmetric learning algorithm $A$,

$$E_{S_m} [\hat{e}^A_{\text{cv}}(S_m)] = E_{S_{m-1}} [\epsilon(A(S_{m-1}))]. \quad (4.3)$$

**Proof.** For any fixed sample $S_m$, let $h^A_{S_m}$, and let $e_i \in \{0, 1\}$ be 1 if and only if $h^A_{S_m}(x_i) \neq y_i$. Then

$$E_{S_m} [\hat{e}^A_{\text{cv}}(S_m)] = E_{S_m} \left[ \frac{1}{m} \sum_i e_i \right] \quad (4.4)$$

$$= \frac{1}{m} \sum_{S_n} E_{S_n} [e_i] \quad (4.5)$$

$$= E_{S_n} [e_i] \quad (4.6)$$

$$= E_{S_{m-1}} [\epsilon(A(S_{m-1}))]. \quad (4.7)$$

The first equality follows from the definition of leave-one-out, the second from the additivity of expectation, the third from the symmetry of $A$, and the fourth from the definition of $e_i$.

The first of our main results follows.

**Theorem 4.** Let $A$ be any deterministic algorithm using a hypothesis space $H$ of VC-dimension $d$ such that $A$ has error stability $(\beta_1, \beta_2)$, and leave-one-out $(\gamma_1, \gamma_2)$ overestimates the training error for $A$. Then for any $\delta > 0$, with probability at least $1 - \delta$ over $S_m$,

$$|\hat{e}^A_{\text{cv}}(S_m) - \epsilon(A(S_m))| \leq \frac{3\sqrt{(d+1)(\ln(2m/d) + 1)}}{m} + 3\beta_1 + \beta_2 + \gamma_1 + \gamma_2. \quad (4.8)$$

Let us briefly discuss the form of the bound given in theorem 4. First, as we mentioned earlier, there is a residual $O(\sqrt{d/m})$ term that remains no matter how error stable the algorithm this. This means that we cannot hope to get something better than a sanity-check bound from this result. Our main applications of theorem 4 will be to show specific, natural cases in which $\gamma_1$ and $\gamma_2$ can be eliminated from the bound, leaving us with a bound that depends only on the error stability and the residual $O(\sqrt{d/m})$ term. We now turn to the proof of the theorem.
Proof. From Theorem 1 and the fact that leave-one-out $(\gamma_1, \gamma_2)$ overestimates the training error, we have that with probability at least $1 - \delta' - \gamma_1$ (where $\delta'$ will be determined by the analysis),

$$
\hat{\epsilon}_v^A(S_m) \geq \hat{\epsilon}(A(S_m)) - \gamma_2 \geq \epsilon(A(S_m)) - VC(d, m, \delta') - \gamma_2.
$$

(4.9)

Thus, the fact that leave-one-out does not underestimate the training error by more than $\gamma_2$ (with probability at least $1 - \gamma_1$) immediately lets us bound the amount by which leave-one-out could underestimate the true error $\hat{A}$. Sm/\hat{A}. Sm/VC.d;m;–0\geq\gamma_2$ (where the bound holds trivially). It remains to bound the amount by which leave-one-out could overestimate the true error.

Let us define the random variable $\chi(S_m)$ by

$$
\chi(S_m) \overset{\text{def}}{=} \hat{\epsilon}_v^A(S_m) - \epsilon(A(S_m));
$$

(4.10)

let

$$
\tau \overset{\text{def}}{=} VC(d, m, \delta') + \gamma_2
$$

(4.11)

and

$$
\rho \overset{\text{def}}{=} \delta' + \gamma_1.
$$

(4.12)

Then equation 4.9 says that the probability that $\chi(S_m) < -\tau$, is at most $\rho$. Furthermore, it follows from the error stability of $A$ that with probability at least $1 - \beta_1$,

$$
\chi(S_m) \leq \hat{\epsilon}_v^A(S_m) - \epsilon(A(S_m-1)) + \beta_2
$$

(4.13)

(where $S_{m-1} \cup \{x, y\} = S_m$). By lemma 1 we know that

$$
E_{S_{m-1},(x,y)}[\hat{\epsilon}_v^A(S_m) - \epsilon(A(S_m-1))] = 0.
$$

(4.14)

Hence, on those samples for which equation (4.13) holds (whose total probability weight is at least $1 - \beta_1$), the expected value of $\hat{\epsilon}_v^A(S_m) - \epsilon(A(S_m-1))$ is at most $\beta_1/(1 - \beta_1)$. Assuming $\beta_1 \leq 1/2$ (since otherwise the bound holds trivially) and using the fact that $|\chi(S_m)| \leq 1$, we have that

$$
E_{S_m}[\chi(S_m)] \leq 3\beta_1 + \beta_2.
$$

(4.15)

Let $\alpha$ be such that with probability exactly $\delta$, $\chi(S_m) > \alpha$. Then

$$
3\beta_1 + \beta_2 \geq E_{S_m}[\chi(S_m)]
$$

(4.16)

$$
\geq \delta\alpha + \rho(-1) + (1 - \delta - \rho)(-\tau)
$$

(4.17)

$$
\geq \delta\alpha - \rho - \tau
$$

(4.18)
where we have again used the fact that \(|\chi(S_m)| \leq 1\) always. Thus

\[
\alpha \leq \frac{3\beta_1 + \beta_2 + \rho + \tau}{\delta}. \tag{4.19}
\]

From the above we have that with probability at least 1 \(- \delta,

\[
\hat{\epsilon}^A_m(S_m) \leq \epsilon(A(S_m)) + \frac{3\beta_1 + \beta_2 + \rho + \tau}{\delta}, \tag{4.20}
\]

\[
= \epsilon(A(S_m)) + \frac{\text{VC}(d, m, \delta') + 3\beta_1 + \beta_2 + \gamma_1 + \gamma_2 + \delta'}{\delta}. \tag{4.21}
\]

If we set \(\delta' = d/m\), we get that with probability at least 1 \(- \delta,

\[
\hat{\epsilon}^A_m(S_m) \leq \epsilon(A(S_m)) + \frac{3\sqrt{\frac{(d+1)(\ln(9m/d)+1)}{m}} + 3\beta_1 + \beta_2 + \gamma_1 + \gamma_2}{\delta}, \tag{4.22}
\]

which together with equation (4.9) proves the theorem.

**4.1 Application to Training Error Minimization.** In this section, we give one of our main applications of Theorem 4, by showing that for training error minimization algorithms, a \(O(\sqrt{d/m})\) bound on the error of leave-one-out can be obtained from error stability arguments. We proceed by giving two lemmas, the first bounding the error stability of such algorithms and the second proving that leave-one-out overestimates their training error.

**Lemma 2.** Let \(A\) be any algorithm performing training error minimization over a hypothesis class \(H\) of VC-dimension \(d\). Then for any \(\beta_1 > 0\), \(A\) has error stability \((\beta_1, 2\text{VC}(d, m-1, \beta_1/2))\).

**Proof.** From uniform convergence (theorem 1), we know that with probability at least 1 \(- \beta_1, both

\[
\epsilon(A(S_{m-1})) \leq \epsilon_{opt} + 2\text{VC}(d, m-1, \beta_1/2), \tag{4.23}
\]

and

\[
\epsilon(A(S_m)) \leq \epsilon_{opt} + 2\text{VC}(d, m, \beta_1/2) \tag{4.24}
\]

hold, while it is always true that both \(\epsilon(A(S_m)) \geq \epsilon_{opt}\), and \(\epsilon(A(S_{m-1})) \geq \epsilon_{opt}\). Thus with probability at least 1 \(- \beta_1,

\[
|\epsilon(A(S_{m-1})) - \epsilon(A(S_m))| \leq 2\text{VC}(d, m-1, \beta_1/2)). \tag{4.25}
\]
Lemma 3. Let A be any algorithm performing training error minimization over a hypothesis class H. Then leave-one-out (0,0) overestimates the training error for A.

Proof. Let \( h = A(S_m) \) and \( h' = A(S_m') \). Let \( \text{err}(S_m) \) be the subset of examples in \( S_m \) on which \( h \) errs. We claim that for every \( (x_i, y_i) \in \text{err}(S_m) \), \( h' \) errs on \( (x_i, y_i) \) as well, implying that \( \hat{\epsilon}^A(S_m) \geq \hat{\epsilon}(A(S_m)) \). Assume, contrary to the claim, that for some \( i \), \( h(x_i) \neq y_i \) while \( h'(x_i) = y_i \). For any function \( g \) and sample \( S \), let \( e_g(S) \) denote the number of errors made by \( g \) on \( S \) (thus \( e_g(S) = \hat{\epsilon}(g) \cdot |S| \)). Since \( A \) performs training error minimization, for any function \( h' \in H \) we have \( e_{h'}(S_m) \geq e_g(S_m) \). Similarly, for any \( h' \in H \), we have \( e_{h'}(S_m') \geq e_{h}(S_m) \). In particular this must be true for \( h \), and thus \( e_h(S_m') \geq e_{h'}(S_m') \). Since \( h \) errs on \( (x_i, y_i) \), \( e_h(S_m') = e_h(S_m) - 1 \), and hence \( e_{h'}(S_m) \leq e_h(S_m) - 1 \). But since \( h' \) does not err on \( (x_i, y_i) \), \( e_{h'}(S_m) = e_{h'}(S_m') \leq e_h(S_m) - 1 < e_h(S_m) \), contradicting the assumption that \( h \) minimizes the training error on \( S_m \).

Theorem 5. Let \( A \) be any algorithm performing training error minimization over a hypothesis class \( H \) of VC-dimension \( d \). Then for every \( \delta > 0 \), with probability at least \( 1 - \delta \),

\[
|\hat{\epsilon}^A(S_m) - \epsilon(A(S_m))| \leq 8\sqrt{\frac{(d+1)(\ln(9m/d)+2)}{m \delta}}.
\]

Proof. Follows immediately from lemma 2 (where \( \beta_1 \) is set to \( 2d/m \)), lemma 3, and theorem 4.

Thus, for training error minimization algorithms, the worst-case behavior of the leave-one-out estimate is not worse than that of the training error (modulo the inferior dependence on \( 1/\delta \) and constant factors). We would like to infer that a similar statement is true if the algorithm almost minimizes the training error. Unfortunately, lemma 3 is extremely sensitive, forcing us to assume that leave-one-out overestimates the training error in the following theorem. We will later discuss how reasonable such an assumption might be for natural algorithms; in any case, we will show in section 5 that some assumptions beyond just error stability are required to obtain interesting bounds for leave-one-out.

Theorem 6. Let \( A \) be a deterministic algorithm that comes within \( \Delta \) of minimizing the training error over \( H \) (that is, on any sample \( S_m, \epsilon(A(S_m)) \leq \min_{h \in H} \{\hat{\epsilon}(h)\} + \Delta \)), and suppose that leave-one-out \((0,0)\) overestimates the training error for \( A \). Then with probability at least \( 1 - \delta \),

\[
|\hat{\epsilon}_e(S_m) - \epsilon(A(S_m))| \leq 8\sqrt{\frac{(d+1)(\ln(2m/d)+2)}{m \delta}} + \Delta.
\]
Thus, for the above bound to be meaningful, $\Delta$ must be relatively small. In particular, for $\Delta = O(\sqrt{d/m})$ we obtain a bound of the same order as the bound of theorem 5. It is an open question whether this is an artifact of our proof technique or the price of generality, as we are interested in a bound that holds for any algorithm that performs approximate training error minimization.

Proof. The theorem follows from the fact that any algorithm that comes within $D$ of minimizing the training error has error stability $(\beta_1, \Delta + 2VC (d, m - 1, \beta_1/2))$ (the proof is similar to that of lemma 2), and from theorem 4.

4.2 Application to Bayesian Algorithms. We have just seen that training error minimization in fact implies error stability sufficient to obtain a sanity-check bound on the error of leave-one-out. More generally, we might hope to obtain bounds that depend on whatever error stability an algorithm does possess. In this section, we show that this hope can be realized for a natural class of randomized algorithms that behave in a Bayesian manner.

To begin, we generalize definitions 2 and 3 to include randomization simply by letting the probability in both definitions be taken over both the sample $S_m$ and any randomization required by the algorithm. We use the notation $A(S, \bar{r})$ to denote the hypothesis output by $A$ on input sample $S$ and random string $\bar{r}$, and $\bar{\epsilon}_\alpha(A(S_m, \bar{r}_1, \ldots, \bar{r}_m))$ to denote the leave-one-out estimate when the random string $\bar{r}_i$ is used on the call to $A$ on $S_m$.

Definition 4. We say that a randomized algorithm $A$ has error stability $(\beta_1, \beta_2)$ if for every $m$

$$\Pr_{S_m-1,(x,y),\bar{r},\bar{r}'} \left[ |\epsilon(A(S_m, \bar{r})) - \epsilon(A(S_m-1, \bar{r}'))| \geq \beta_2 \right] \leq \beta_1$$

(4.28)

where $S_m = S_m-1 \cup \{(x,y)\}$.

Definition 5. For any randomized algorithm $A$, we say that leave-one-out $(\gamma_1, \gamma_2)$ overestimates the training error for $A$ if for every $m$

$$\Pr_{S_m-1,(x,y),\bar{r},\bar{r}_1,\ldots,\bar{r}_m} \left[ \hat{\epsilon}_\alpha(A(S_m, \bar{r}_1, \ldots, \bar{r}_m)) \leq \hat{\epsilon}(A(S_m, \bar{r})) \right] \leq \gamma_1$$

(4.29)

where $S_m = S_m-1 \cup \{(x,y)\}$.

The proof of the following theorem is essentially the same as the proof of theorem 4, where the only difference is that all probabilities are taken over the sample $S_m$ and the randomization of the algorithm.
Theorem 7. Let $A$ be any randomized algorithm using a hypothesis space $H$ of VC-dimension $d$ such that leave-one-out $(\gamma_1, \gamma_2)$ overestimates the training error for $A$, and $A$ has error stability $(\beta_1, \beta_2)$. Then for any $\delta > 0$, with probability at least $1 - \delta$, 

$$|\hat{\epsilon}_{cv}(S_m, \tilde{r}_1, \ldots, \tilde{r}_m) - \epsilon(A(S_m, \tilde{r}))|$$

$$= \frac{3\sqrt{(d+1)(\log \frac{2d^2}{\delta})}}{m} + 3\beta_1 + \beta_2 + \gamma_1 + \gamma_2. \quad (4.30)$$

Here the probability is taken over the choice of $S_m$, and over the coin flips $\tilde{r}_1, \ldots, \tilde{r}_m$ and $\tilde{r}$ of $A$ on the $S_m$ and $S$. 

We now apply theorem 4 to the class of Bayesian algorithms—that is, algorithms that choose their hypotheses according to a posterior distribution, obtained from a prior that is modified by the sample data and a temperature parameter. Such algorithms are frequently studied in the simulated annealing and statistical physics literature on learning (Seung, Sompolinsky, & Tishby, 1992; Geman & Geman, 1984). 

Definition 6. We say that a randomized algorithm $A$ using hypothesis space $H$ is a Bayesian algorithm if there exists a prior $P$ over $H$ and a temperature $T \geq 0$ such that for any sample $S_m$ and any $h \in H$, 

$$P_r[A(S_m, \tilde{r}) = h] = \frac{1}{Z} P(h) \exp \left( - \frac{1}{T} \sum_{i} I(h(x_i) \neq y_i) \right). \quad (4.31)$$

Here $Z = \sum_{h \in H} P(h) \exp (- \frac{1}{T} \sum_{i} I(h(x_i) \neq y_i))$ is the appropriate normalization and $I(\cdot)$ is the indicator function. 

Note that we still do not assume anything about the target function (for instance, it is not necessarily drawn according to $P$ or any other distribution); it is only the algorithm that behaves in a Bayesian manner. Also, note that the special case in which $T = 0$ and the support of $P$ is $H$ results in training error minimization.

We begin by giving a general lemma that identifies the only property about Bayesian algorithms that we will need; thus, all of our subsequent results will hold for any algorithm meeting the conclusion of this lemma.

Lemma 4. Let $A$ be a Bayesian algorithm. For any sample $S$ and any example $(x, y) \in S$, let $p$ be the probability over $\tilde{r}$ that $A(S, \tilde{r})$ errs on $(x, y)$, and let $p'$ be the probability over $\tilde{r}'$ that $A(S - (x, y), \tilde{r}')$ errs on $(x, y)$. Then $p' \geq p$.

Proof. Let $P$ be the distribution induced over $H$ when $A$ is called on $S$, and let $P'$ be the distribution over $H$ induced when $A$ is called on $S - (x, y)$. 

---

(Continued in the next page)
Then for any \( h \in H \), \( \mathcal{P}(h) = \tfrac{1}{2} \mathcal{P}^{*}(h) \) if \( h \) does not err on \( (x, y) \), and \( \mathcal{P}(h) = \tfrac{1}{2} \exp(-\tfrac{1}{2}) \mathcal{P}^{*}(h) \) if \( h \) does err on \( (x, y) \). Thus the only change from \( \mathcal{P}^{*} \) to \( \mathcal{P} \) is to decrease the probability of drawing an \( h \) that errs on \( (x, y) \).

The key result leading to a sanity-check bound for Bayesian algorithms follows. It bounds the extent to which leave-one-out overestimates the training error in terms of the error stability of the algorithm.

**Theorem 8.** Let \( A \) be a Bayesian algorithm (or any other algorithm satisfying the conclusion of lemma 4) that has error stability \((\beta_1, \beta_2)\). Then for any \( \alpha > 0 \), leave-one-out \((\gamma_1, \gamma_2)\) overestimates the training error for \( A \) for \( \gamma_1 = 2\alpha + 3\sqrt{\beta_1} \), and \( \gamma_2 = 2\sqrt{\beta_1} + 4\beta_2 + 4V C(d, m, \alpha) + \sqrt{\log(1/\alpha)/m} \).

In order to prove theorem 8, we first need the following lemma, which says that with respect to the randomization of a Bayesian algorithm, the leave-one-out estimate is likely to overestimate the expected training error.

**Lemma 5.** Let \( A \) be a Bayesian algorithm (or any randomized algorithm satisfying the conclusion of lemma 4). Then for any fixed sample \( S_m = S_{m-1} \cup \{(x, y)\} \), with probability at least \( 1 - \delta \) over \( \tilde{r}_1, \ldots, \tilde{r}_m \) and \( \tilde{r} \),

\[
\hat{\epsilon}_A(S_m, \tilde{r}_1, \ldots, \tilde{r}_m) \geq \mathbb{E}[\hat{\epsilon}(A(S_m, \tilde{r}))] - \sqrt{\log(1/\delta)/m}. \tag{4.32}
\]

**Proof.** For each \( (x_i, y_i) \in S_m \), let \( p_i \) be the probability over \( \tilde{r} \) that \( A(S_m, \tilde{r}) \) errs on \( (x_i, y_i) \) and let \( p'_i \) be the probability over \( \tilde{r}_i \) that \( A(S'_m, \tilde{r}_i) \) errs on \( (x_i, y_i) \). By lemma 4 we know that \( p'_i \geq p_i \). Then

\[
\mathbb{E}[\hat{\epsilon}(A(S_m, \tilde{r}))] = \sum_{h \in H} \mathbb{P}[A(S_m, \tilde{r}) = h] \cdot \hat{\epsilon}(h) \tag{4.33}
\]

\[= \sum_{h \in H} \mathbb{P}[A(S_m, \tilde{r}) = h] \cdot \frac{1}{m} \sum_{i} I(h(x_i) \neq y_i) \tag{4.34}
\]

\[= \frac{1}{m} \sum_{i} \sum_{h \in H} \mathbb{P}[A(S_m) = h] \cdot I(h(x_i) \neq y_i) \tag{4.35}
\]

\[= \frac{1}{m} \sum_{i} p_i. \tag{4.36}
\]

Denote \((1/m) \sum_i p_i \) by \( \bar{p} \), and \((1/m) \sum_i p'_i \) by \( \bar{p}' \). Let \( e_i \) be a Bernoulli random variable determined by \( \tilde{r}_i \), which is 1 if \( A(S'_m, \tilde{r}_i) \) errs on \( (x_i, y_i) \) and 0 otherwise. By definition, \( \hat{\epsilon}_A(S_m, \tilde{r}_1, \ldots, \tilde{r}_m) = (1/m) \sum_i e_i \), and

\[
\mathbb{E} \left[ \hat{\epsilon}_A(S_m, \tilde{r}_1, \ldots, \tilde{r}_m) \right] = \mathbb{E} \left[ (1/m) \sum_i e_i \right] \tag{4.37}
\]
By Chernoff’s inequality, for any $\alpha$,

$$
\Pr_{\tilde{r}_1, \ldots, \tilde{r}_m} \left[ \left( \frac{1}{m} \right) \sum_i e_i \leq \tilde{p}' - \alpha \right] < \exp(-2\alpha^2 m).
$$

(4.40)

By setting $\alpha = (1/2)\sqrt{\log(1/\delta)/m}$, we have that with probability at least $1 - \delta$ over the choice of $\tilde{r}_1, \ldots, \tilde{r}_m$,

$$
\hat{\epsilon}_m(A(S_m, \tilde{r}_1, \ldots, \tilde{r}_m)) \geq \mathbb{E}[\hat{\epsilon}(A(S_m, \tilde{r}))] - (1/2)\sqrt{\log(1/\delta)/m}.
$$

(4.41)

Now we can give the proof of theorem 8.

**Proof (Theorem 8).** Because $A$ has error stability $(\beta_1, \beta_2)$, if we draw $S_{m-1}$ and $(x, y)$ at random, we have probability at least $1 - \sqrt{\beta_1}$ of obtaining an $S_m$ such that

$$
\Pr_{\tilde{r}, \tilde{r}'} \left[ |\epsilon(A(S_m, \tilde{r})) - \epsilon(A(S_{m-1}, \tilde{r}''))| \geq \beta_2 \right] \leq \sqrt{\beta_1}.
$$

(4.42)

Equation 4.42 relates the error when $A$ is called on $S_m$ and $S_m'$. We would like to translate this to a statement relating the error when $A$ is called on $S_m$ twice. But if $S_m$ satisfies equation 4.42, it follows that

$$
\Pr_{\tilde{r}, \tilde{r}'} \left[ |\epsilon(A(S_m, \tilde{r})) - \epsilon(A(S_m, \tilde{r}'))| \geq 2\beta_2 \right] \leq 2\sqrt{\beta_1}.
$$

(4.43)

The reason is that if $|\epsilon(A(S_m, \tilde{r})) - \epsilon(A(S_m, \tilde{r}''))| \geq 2\beta_2$, then $\epsilon(A(S_m, \tilde{r}'))$ can be within $\beta_2$ of only one of $\epsilon(A(S_m, \tilde{r}))$ and $\epsilon(A(S_m, \tilde{r}''))$, and each is equally likely to result from a call to $A$ on $S_m$. From equation 4.43 and theorem 1, we have that with probability at least $1 - \alpha - \sqrt{\beta_1}$, $S_m$ will satisfy

$$
\Pr_{\tilde{r}, \tilde{r}'} \left[ |\hat{\epsilon}(A(S_m, \tilde{r})) - \hat{\epsilon}(A(S_m, \tilde{r}'))| \geq 2\beta_2 + 2VC(d, m, \alpha) \right] \\
\leq 2\sqrt{\beta_1}.
$$

(4.44)

If $S_m$ satisfies equation 4.44, it follows that there must be a fixed value $\hat{\epsilon}_0 \in [0, 1]$ such that

$$
\Pr_{\tilde{r}}[|\hat{\epsilon}(A(S_m, \tilde{r})) - \hat{\epsilon}_0| \geq 2\beta_2 + 2VC(d, m, \alpha)] \leq 2\sqrt{\beta_1}.
$$

(4.45)
Assuming that equation 4.45 holds, we get the following bounds on $E[$\hat{\epsilon}(A(S_m, \tilde{r}))$]:

$$E[\hat{\epsilon}(A(S_m, \tilde{r}))] \leq (1 - 2\sqrt{\beta_1})(\hat{\epsilon}_0 + 2\beta_2 + 2VC(d, m, \alpha)) + 2\sqrt{\beta_1} \cdot 1$$

(4.46)

and

$$E[\hat{\epsilon}(A(S_m, \tilde{r}))] \geq (1 - 2\sqrt{\beta_1})(\hat{\epsilon}_0 - 2\beta_2 - 2VC(d, m, \alpha)) + 2\sqrt{\beta_1} \cdot 0.$$  

(4.47)

In either case,

$$\left| E[\hat{\epsilon}(A(S_m, \tilde{r}))] - \hat{\epsilon}_0 \right| \leq 2\sqrt{\beta_1} + 2\beta_2 + 2VC(d, m, \alpha)$$

(4.48)

and thus by equation 4.45, with probability at least $1 - \alpha - \sqrt{\beta_1}$ over the draw of $S_m$, $S_m$ will be such that the probability over $\tilde{r}$ that

$$\left| \hat{\epsilon}(A(S_m, \tilde{r})) - E[\hat{\epsilon}(A(S_m, \tilde{r}'))] \right| \geq 2\beta_2 + 2VC(d, m, \alpha) + 2\sqrt{\beta_1} + 2\beta_2 + 2VC(d, m, \alpha)$$

(4.49)

is at most $2\sqrt{\beta_1}$. Combined with lemma 5, we obtain that with probability at least $1 - 2\alpha - 3\sqrt{\beta_1}$ over $S_m, \tilde{r}_1, \ldots, \tilde{r}_m$ and $\tilde{r}$,

$$\hat{\epsilon}_c^A(S_m, \tilde{r}_1, \ldots, \tilde{r}_m) \geq \hat{\epsilon}(S_m, \tilde{r}) - 2\sqrt{\beta_1} - 4\beta_2 - 4VC(d, m, \alpha) - \sqrt{\log(1/\alpha)/m}$$

(4.50)

as desired.

Now we can give the main result of this section.

**Theorem 9.** Let $A$ be a Bayesian algorithm (or any randomized algorithm satisfying the conclusion of lemma 4) that has error stability $(\beta_1, \beta_2)$. Then for any $\delta > 0$, with probability at least $1 - \delta$,

$$|\hat{\epsilon}_c^A(S_m, \tilde{r}_1, \ldots, \tilde{r}_m) - \epsilon(A(S_m, \tilde{r}'))| \leq 10\sqrt{(d+1)(\log(nm/d+1))m\delta}/m + 8\sqrt{\beta_1} + 5\beta_2.$$  

(4.51)

Here the probability is taken over the choice of $S_m$, and over the coin flips $\tilde{r}_1, \ldots, \tilde{r}_m$ and $\tilde{r}$ of $A$ on the $S_m$ and $S_m$.
Thus, theorem 9 relates the error of leave-one-out to the stability of a Bayesian algorithm: as $\beta_1, \beta_2 \to 0$, we obtain a $\tilde{O}(\sqrt{d/m})$ bound. Note that for Bayesian algorithms, we expect increasing error stability (i.e., $\beta_1, \beta_2 \to 0$) as the number of examples increases or as the temperature decreases.

4.3 Application to Linear Functions and Squared Error. In this section, we briefly describe an extension of the ideas developed so far to problems in which the outputs of both the target function and the hypothesis functions are real-valued and the error measure is squared loss. The importance of this extension is due to the fact that for squared error, there is a particularly nice case (linear hypothesis functions) for which empirical error minimization can be efficiently implemented, and the leave-one-out estimate can be efficiently computed.

Our samples $S_m$ now consist of examples $(x_i, y_i)$, where $x_i \in \mathbb{R}^d$ and $y_i \in [-1, 1]$. For any function $h: \mathbb{R}^d \to [-1, 1]$, we now define the generalization error by

$$\epsilon(h) \overset{\text{def}}{=} \mathbb{E}_{(x,y)} \left[ (h(x) - y)^2 \right].$$

and similarly the training error becomes

$$\hat{\epsilon}(h) = \sum_{(x_i, y_i) \in S} (h(x_i) - y_i)^2.$$ (4.53)

For any algorithm $A$, if $h^i$ denotes $A(S^i_m)$, the leave-one-out estimate is now

$$\hat{\epsilon}_{cv}^A(S_m) \overset{\text{def}}{=} \sum_{(x_i, y_i) \in S_m} (h^i(x_i) - y_i)^2.$$ (4.54)

It can be verified that in such situations, provided that a uniform convergence result analogous to theorem 1 can be proved, then the analog to theorem 5 can be obtained (with essentially the same proof), where the expression $VC(d, m, \delta)$ in the bound must be replaced by the appropriate uniform convergence expression. We will not state the general theorem here, but instead concentrate on an important special case. It can easily be verified that lemma 3 still holds in the squared error case: that is, if $A$ performs (squared) training error minimization, then for any sample $S_m$, $\hat{\epsilon}_{cv}^A(S_m) \geq \hat{\epsilon}(A(S_m))$. Furthermore, if the hypothesis space $H$ consists of only linear functions $w \cdot x$, then provided the squared loss is bounded for each $w$, nice uniform convergence bounds are known.

**Theorem 10.** Let the target function be an arbitrary mapping $\mathbb{R}^d \to [-B, B]$, where $B > 0$ is a constant, and let $P$ be any input distribution over $[-B, B]^d$. Let $A$
perform squared training error minimization over the class of all linear functions \( w \cdot x \) obeying \( \|w\| \leq B \). Then for every \( \delta > 0 \), with probability at least \( 1 - \delta \),

\[
|\hat{\epsilon}_m^A(S_m) - \epsilon(A(S_m))| = O\left(\sqrt{\frac{d}{m}}(\log\frac{d}{m})/\delta\right). \tag{4.55}
\]

Note that while the bound given in theorem 10 is weaker than that proved by Vapnik (1982, chap. 8) (for squared error minimization over the class of linear functions), it is much more general. We make no assumptions on the distribution according to which the examples are generated and the function labeling them.

Two very fortunate properties of the combination of linear functions and squared error make the sanity-check bound given in theorem 10 of particular interest: First, there exist polynomial-time algorithms for performing minimization of squared training error (Duda & Hart, 1973) by linear functions. These algorithms do not necessarily obey the constraint \( \|w\| \leq B \), but we suspect this is not an obstacle to the validity of theorem 10 in most practical settings. Second, there is an efficient procedure for computing the leave-one-out estimate for training error minimization of the squared error over linear functions (Miller, 1990). Thus, it is not necessary to run the error minimization procedure \( m \) times; there is a closed-form solution for the leave-one-out estimate that can be computed directly from the data much more quickly.

More generally, many of the results given in this article can be generalized to other loss functions via the proper generalizations of uniform convergence (Haussler, 1992).

4.4 Other Algorithms. We now comment briefly on the application of theorem 4 to algorithms other than error minimization and Bayesian procedures. As we have already noted, the only barrier to applying theorem 4 to obtain bounds on the leave-one-out error that depend only on the error stability and \( Q \).

\[
\hat{\epsilon}_m^A(S_m) = \epsilon(A(S_m)) \cdot \frac{d}{m} = \frac{1}{m} \cdot \frac{d}{m}.
\]

Note that in the deterministic case, leave-one-out will \((0, 0)\) overestimate the training error as long as \( A \) has the stronger property that if \( A(S_m) \) erred on an example \( (x, y) \in S_m \), then \( A(S_m - \{(x, y)\}) \) also erred on \( (x, y) \) as well. In other words, the removal of a point from the sample cannot improve the algorithm’s performance on that point. This stronger property is exactly what was proved in lemma 3 for training error minimization, and its randomized algorithm analog was shown for Bayesian algorithms in lemma 4. To see why this property may be plausible for a natural heuristic,
consider (in the squared error case) an algorithm that is performing a gradient descent on the training error over some continuous parameter space \( \tilde{w} \). Then the gradient with respect to \( \tilde{w} \) can be written as a sum of gradients, one for each example in \( S_m \). The gradient term for \((x, y)\) gives a force on \( \tilde{w} \) in a direction that causes the error on \((x, y)\) to decrease. Thus, the main effect on the algorithm of removing \((x, y)\) is to remove this term from the gradient, which intuitively should cause the algorithm’s performance on \((x, y)\) to degrade. (The reason that this argument cannot be turned into a proof of training error overestimation is that it technically is valid only for one step of the gradient descent.) It is an interesting open problem to verify whether this property holds for widely used heuristics.

5 Lower Bounds

In this section, we establish the following:

- That the dependence on \( 1/\delta \) is in general unavoidable for the leave-one-out estimate.

- That in the case of algorithms that perform error minimization, the dependence of the error of leave-one-out on the VC-dimension cannot be removed without additional assumptions on the algorithm.

- That for any algorithm, some form of error stability is necessary in order to provide nontrivial bounds on the leave-one-out estimate.

- That there exist algorithms with perfect error stability for which the leave-one-out estimate is arbitrarily poor, and furthermore, these algorithms use a hypothesis class with constant VC-dimension.

These last two points are especially important. Although we cannot prove that precisely our form of error stability is necessary, we can prove the necessity of a slightly weaker form of error stability. This implies that the leave-one-out estimate cannot provide very good bounds if no error-stability condition is met. On the other hand, we show that error stability by itself is not sufficient even when the hypothesis class has very small VC-dimension. Therefore, additional assumptions on the algorithm must be made. The additional assumptions made in theorem 4 were sufficient training error overestimation and bounded VC-dimension. In contrast, hypothesis stability alone is a sufficient condition for nontrivial bounds, but is far from necessary. We note that some of our lower bounds (as is often the case with such bounds) use quite singular distributions. It is an open question whether one can obtain improved bounds under certain continuity assumptions on the underlying distribution.

We begin with the lower bound giving an example where there is an \( \Omega(1/\sqrt{m}) \) chance of constant error for the leave-one-out estimate. Setting \( d = 1 \) in theorem 4 shows that the dependence on \( \delta \) given there is tight (up...
Bounds for Leave-One-Out Cross-Validation 1449

to logarithmic factors). This theorem has appeared elsewhere (Devroye et al., 1996, Chap. 24), but we include it here for completeness.

**Theorem 11.** There exists an input distribution $P$, a target function $f$, a hypothesis class $H$ of VC-dimension 1, and an algorithm $A$ that minimizes the training error over $H$ such that with probability $\tilde{p} = \frac{m}{\sqrt{m}}$, $|\hat{\epsilon}_o^A(S_m) - \epsilon(A(S_m))| = \Omega(1)$.

**Proof.** Let the input space $X$ consist of a single point $x$, and let the target function $f$ be the probabilistic function that flips a fair coin on each trial to determine the label to be given with $x$. Thus, the generalization error of any hypothesis is exactly $1/2$. The algorithm $A$ simply takes the majority label of the sample as its hypothesis. Now with probability $\tilde{p} = \frac{m}{\sqrt{m}}$, the sample $S_m$ will have a balanced number of positive and negative examples, in which case $\hat{\epsilon}_o^A(S_m) = 1$, proving the theorem.

The following theorem shows that in the case of algorithms that perform training error minimization, the dependence of the error of the leave-one-out estimate on the VC-dimension is unavoidable without further assumptions on the algorithm.

**Theorem 12.** For any $d$, there exists an input distribution $P$, a target function $f$, a hypothesis class $H$ of VC-dimension $d$, and an algorithm $A$ that minimizes the training error over $H$ such that with probability $\tilde{p} = \frac{m}{\sqrt{m}}$, $|\hat{\epsilon}_o^A(S_m) - \epsilon(A(S_m))| = \Omega(d/m)$.

**Proof.** Let $X = [0,1]$, and the underlying distribution be uniform. The hypothesis class $H$ consists of all $d$-switch functions over $[0,1]$ (that is, it consists of all functions defined by $d + 1$ disjoint intervals covering $[0,1]$, with a binary label associated with each interval). Let $h_d$ be the $d$-switch function over $[0,1]$ in which the switches are evenly spaced $1/d$ apart.

The algorithm $A$ behaves as follows: if the sample size $m$ is even, $A$ first checks if $h_d$ minimizes the training error on the sample. If so, it selects $h_d$ as its hypothesis. Otherwise, $S$ chooses the left-most hypothesis that minimizes the training error over $[0,1]$ (that is, the hypothesis that minimizes the training error and always chooses its switches to be as far to the left as possible between the two sample points where the switch occurs). If the sample size is odd, $A$ chooses the left-most hypothesis minimizing the training error over $[0,1]$. Thus, on even samples, $A$ has a strong bias toward choosing $h_d$ over $[0,1]$, but on odd samples, it has no such bias.

Now suppose that the target function labels $[0,1]$ according to $h_d$, and let $m$ be even. Then $A$ necessarily chooses $h_d$ (as it is consistent with the sample), and so $\epsilon(A(S_m)) = 0$. But when estimating this generalization error, for each point $x_i$ in the sample that is a left-most point in the interval it belongs to,
A(S_m^j) ε on x_i. Since there are d intervals, with high probability e^A_m(S_m) will be Ω(d/m), as desired.

We next show that some form of error stability is essential for providing upper bounds on the error of the leave-one-out estimate.

**Definition 7.** We say that a deterministic algorithm A has error stability β in expectation if for every m [E_{S_{m-1},(x,y)}[e(A(S_{m-1})) - e(A(S_m))] ≤ β, where S_m = S_{m-1} \cup \{(x,y)\} (and β may be a function of m).

Thus we are asking that the difference between the errors (of the hypotheses output by A when trained on S_m and on S_{m-1}, respectively) be small, in expectation. This is in general weaker than the requirement in definition 2, since the above expectation could be 0, while there is high probability that the error sometimes increases by much when a point is added, and sometimes decreases by much.6 We note however, that for any reasonable learning algorithm, in which there is very small probability that the error increases when a point is added, the above definition is not effectively weaker. For such reasonable algorithms, we are essentially showing that error stability as defined in definition 2 is necessary.

**Theorem 13.** Let A be any algorithm that does not have error stability β in expectation. Then there exists values of m such that for any τ > 0,

\[
Pr_{S_m}[|\tilde{e}_m^A(S_m) - e(A(S_m))| ≥ τ] > \frac{β - τ}{1 - τ}. \tag{5.1}
\]

**Proof.** Since A does not have error stability β in expectation, it is either the case that for some m E_{S_{m-1},(x,y)}[e(A(S_{m-1})) - e(A(S_m))] > β or that E_{S_{m-1},(x,y)}[e(A(S_{m-1})) - e(A(S_m))] < -β. Without loss of generality, assume the former is true. Let χ(S_m) be a random variable that is defined as follows: χ(S_m) = e^A_m(S_m) - e(A(S_m)). Thus, χ(S_m) = e^A_m(S_m) - e(A(S_{m-1})) - e(A(S_{m-1})) - e(A(S_m)) and

\[
E[\chi(S_m)] = E_{S_m, S_{m-1},(x,y)}[e^A_m(S_m) - e(A(S_{m-1}))]
+ E_{S_{m-1},(x,y)}[e(A(S_{m-1})) - e(A(S_m))], \tag{5.2}
\]

6 Precisely for this reason we were not able to show that the stronger notion of error stability is in fact necessary. We are not able to rule out the case that an algorithm is very unstable (according to definition 2) but that this instability averages out when computing the leave-one-out estimate.
where $S_m = S_{m-1} \cup \{(x, y)\}$. By lemma 1 and our assumption on $A$, we get that $E_{S_m}(X(S_m)) > \beta$. Let $\rho$ be the exact probability that $|X(S_m)| \leq \tau$. Then

$$\beta < E_{S_m}[X(S_m)]$$

$$\leq \rho \cdot \tau + (1 - \rho) \cdot 1$$

$$= 1 - \rho(1 - \tau).$$

Thus, $\rho < (1 - \beta)/(1 - \tau)$, and equivalently,

$$1 - \rho > \frac{\beta - \tau}{1 - \tau},$$

which means that with probability at least $(\beta - \tau)/(1 - \tau)$, $\epsilon_{cv}(S_m) - \epsilon(A(S_m)) > \tau$.

Finally, we show that unlike hypothesis stability, error stability alone is not sufficient to give nontrivial bounds on the error of leave-one-out even when the hypothesis class has very small VC-dimension, and hence additional assumptions are required.

**Theorem 14.** There exists an input distribution $P$, a target function $f$, a hypothesis class $H$ with constant VC-dimension, and an algorithm $A$, such that $A$ has error stability $(0, 0)$ with respect to $P$ and $f$, but with probability $1$, $|\epsilon_{cv}(S_m) - \epsilon(A(S_m))| = 1/2$.

**Proof.** Let $X = \{0, \ldots, N - 1\}$ where $N$ is even, $f$ the constant 0 function, $P$ the uniform distribution on $X$, and $H$ the following class of (boolean) threshold functions:

$$H \overset{\Delta}{=} \{ h_t : t \in \{0, \ldots, N - 1\},$$

$$\text{where } h_t(x) = 1 \text{ iff } (t + x) \mod N < N/2 \}.$$  \hfill (5.7)

Clearly, the VC-dimension of $H$ is 2. Furthermore, for every $h \in H$, the distance between $f$ and $h$ is exactly $1/2$, and hence any algorithm using hypothesis class $H$ is $(0, 0)$ stable with respect to $f$. It thus remains to show that there exists an algorithm $A$ for which the leave-one-out estimate always has large error.

For a given sample $S_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, let $t = (\sum_{i=1}^{m} x_i) \mod N$, and let $A(S_m) = h_t$, where $h_t$ is as defined in equation 5.7. Thus, the algorithm’s hypothesis is determined by the sum of the (unlabeled) examples. We next compute the leave-one-estimate of the algorithm on $S_m$. Assume first that $S_m$ is such that $(\sum_{i=1}^{m} x_i) \mod N < N/2$. Then, by definition of $A$, for
each \( x_i \), the hypothesis \( h_i^* = A(S_m^i) \) will label \( x_i \) by 1 whereas \( f(x_i) = 0 \). Similarly, if \( S_{\overline{m}} \) is such that \( (\sum_{i=1}^{m} x_i) \mod N \geq N/2 \), then for each \( x_i \), \( h_i^*(x_i) = 0 \) which is the correct label according to \( f \). In other words, for half of the samples \( S_m \) we have \( \epsilon_{ACV}(S_m) = 1 \), which means that leave-one-out overestimates \( \epsilon(A(S_m)) = 1/2 \) by 1/2, and for half of the sample it underestimates the error by 1/2.

6 Extensions and Open Problems

It is worth mentioning explicitly that in the many situations when uniform convergence bounds better than \( VC(d, m, \delta) \) can be obtained (Hausler, Kearns, Seung, & Tishby, 1996; Seung, Sompolinsky, & Tishby, 1992), our resulting bounds for leave-one-out will be correspondingly better as well.

There are a number of interesting open problems, both theoretical and experimental. On the experimental side, it would be interesting to determine the “typical” dependence of the leave-one-out estimate’s performance on the VC-dimension for various commonly used algorithms. It would also be of interest to establish the extent to which these algorithms possess error stability and leave-one-out overestimates the training error. On the theoretical side, it would be nice to prove sanity-check bounds for leave-one-out for popular heuristics like C4.5 and backpropagation. Also, it would be interesting to find additional properties other than training error overestimation, which together with error stability and bounded VC-dimension suffice for proving sanity-check bounds. Finally, there is almost certainly room for improvement in both our upper and lower bounds: our emphasis has been on the qualitative behavior of leave-one-out in terms of a number of natural parameters of the problem, not the quantitative behavior.

Acknowledgments

Thanks to Avrim Blum for interesting discussions on cross-validation, to Sean Holden for pointing out a mistake we had in theorem 13 and other discussions, and to Nabil Kahale for improving the construction in the proof of theorem 14. We also thank two anonymous referees for their comments.

This work was done at MIT and while visiting AT&T. It was supported by an NSF Postdoctoral Fellowship and an ONR Science Scholar Fellowship at the Bunting Institute.

References

Bounds for Leave-One-Out Cross-Validation


Received September 8, 1997; accepted November 2, 1998.