A Comparison of Tight Generalization Error Bounds

Matti Kääriäinen MATTI.KAARIAINEN@CS.HELSINKI.FI
International Computer Science Institute, 1947 Center St Suite 600, Berkeley, CA 94704, USA

John Langford JL@TTI-C.ORG
TTI-Chicago, 1427 E 60th Street, Chicago, IL 60637, USA

Abstract

We investigate the empirical applicability of several bounds (a number of which are new) on the true error rate of learned classifiers which hold whenever the examples are chosen independently at random from a fixed distribution.

The collection of tricks we use includes:

1. A technique using unlabeled data for a tight derandomization of randomized bounds.
2. A tight form of the progressive validation bound.
3. The exact form of the test set bound.

The bounds are implemented in the `semibound` package and are freely available.

1. Introduction

When we learn a classifier c, it is very natural to wonder "How often will c be wrong in the future?" This question cannot be answered in general, because any source of future examples might know c and either choose $c(x) = y$ or not. However, this question can be answered when assumptions are made about the data.

A basic principle of system design is that if a system has few components then it can break in fewer ways. For this problem, components are assumptions which may or may not be true on any individual problem. Learning theory shows us that using *only* the assumption that data is drawn IID, we can construct confidence intervals on the error rate of learned classifiers. Thus, we make only the IID assumption.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

There are several common techniques for attempting to predict the future error rate of a classifier. For example, sometimes people compute the standard deviation of the error rate on a test set. This is statistically questionable because 0/1 errors are inherently not drawn from a normal distribution. This leads to odd or misleading results like reporting "accuracy 0.9 ± 0.3" and "1.00 ± 0.0" (the last is especially difficult because it's a form of overconfidence). Another approach is to use k-fold cross-validation on a dataset and then use similar methods to transform the cross-validation estimate into an estimate or heuristic confidence interval of the error of the final classifier learned from all examples. This, again, is statistically questionable because the runs of cross validation are not independent, the error rates are not Gaussian, and nothing in general guarantees that the error rate of the final classifier learned from all data is close to the error rates of the classifiers learned in cross-validation.

In this paper, we test several different styles of error bounds that lead to confidence intervals which hold based only on the IID assumption. The baseline version of this approach is to simply use a test set and compute a Binomial confidence interval. This technique is sometimes unsatisfactory because it "wastes examples" which might be critical to successful learning, so we test other approaches, including:

1. Transforming cross-validation and bagging based error estimates into error bounds for randomized classifiers.
2. Using unlabeled data to get tight training set bounds (Kääriäinen, 2005) based on derandomization of bounds for randomized classifiers, including, e.g., the bounds in 1 and the PAC-Bayesian margin bound.
3. Using progressive validation (Blum et al., 1999) over a test set to improve performance so that a portion of the test set can be used for training and

validation.

4. Combinations of the above.

All of the bounds are stated with respect to a classification problem defined by a probability measure D over a space $X \times Y$, where X is some arbitrary feature space and Y a set of labels. The IID assumption we make means that we assume that all examples are drawn independently at random from D (for labeled examples) or its marginal D_X (for unlabeled examples). With this notation, the error rate $e(c, D)$ of a classifier c is $e(c, D) = \Pr_{(x,y)\sim D}(c(x) \neq y)$.

The organization of the paper is straightforward. We describe the techniques that we apply, the settings that they apply in, and then test them with a wide array of experiments.

2. Test Set Bound

When a classifier c is trained on some training set and then tested on an independent test set S sampled from D, the distribution of the test set error

$$\hat{e}(c, S) = \frac{1}{|S|} \sum_{(x,y)\in S} I(c(x) \neq y)$$

is Binomial with parameters $e(c, D)$ and $|S|$. The following inverse binomial tail defines and lets us compute an exact $1 - \delta$ confidence interval for $e(c, D)$.

Definition 1 *(Langford, 2005) The inverse binomial tail* $\overline{\mathrm{Bin}}\,(\hat{p}, m, \delta)$ *is the* q *for which*

$$\sum_{i=0}^{\lceil \hat{p}m \rceil} \binom{m}{i} q^i (1-q)^{m-i} = \delta.$$

The interpretation of the definition is that getting an error rate of $\hat{p}$ or smaller from a binomial distribution with bias larger than $\overline{\mathrm{Bin}}\,(\hat{p}, m, \delta)$ has probability less than δ. Therefore, the true error rate p has to be below $\overline{\mathrm{Bin}}\,(\hat{p}, m, \delta)$ with confidence $1 - \delta$. Besides stating this formally, the next lemma is a robust baseline and a frequently used building block in the bounds that follow.

Lemma 1 *(Test Set Bound) For all* D, c

$$\Pr_{S\sim D^m}\left(\overline{\mathrm{Bin}}\,(\hat{e}(c, S), m, \delta) \geq e(c, D)\right) \geq 1 - \delta.$$

3. Semi-Supervised Bounds

There is some tension between what we can prove and what we want to prove. For example, the (empirically) tightest learning theory bounds for many applications tend to be for randomized classifiers (see (Langford & Shawe-Taylor, 2002) for one example). However, a randomized classifier is inconvenient and counterintuitive for many practical purposes.

The safe derandomization technique we discuss here relieves this tension using a spare unlabeled dataset U drawn from D_X^m with m examples. This makes the bounds semi-supervised.

Randomized classifiers are metaclassifiers defined by a distribution on a set of deterministic classifiers. A randomized classifier is used by drawing an independent sample from this distribution of classifiers each time a classification is given. Standard deterministic classifiers are the special case in which the distribution concentrates on a single classifier. Analogous to deterministic classifiers, the generalization error of a randomized classifier f is its probability of making an error, that is $e(f, D) = \Pr_{(x,y)\sim D, c\sim f}(c(x) \neq y) = E_{c\sim f} e(c, D)$.

The basic idea behind safe derandomization is to use the disagreement probability $d(f, g) = d(f, g, D) = \Pr_{(x,y)\sim D}(f(x) \neq g(x))$ as a metric in the space of randomized classifiers. This quantity can be estimated in a semi-supervised setting by

$$\hat{d}(f, g, U) = \frac{1}{|U|} \sum_{x\in U} I(f(x) \neq g(x))$$

which can be computed by simply classifying the examples in U by f and g and counting the number of times they disagree. Note that $\hat{d}(f, g, U)$ depends both on the randomness in f and g and the randomness introduced by the choice of U from D_X^m. If neither f nor g depend on U, the distribution of $\hat{d}(f, g, U)$ is binomial with parameters m and $d(f, g)$ since $\hat{d}(f, g, U)$ is simply the proportion of times f and g disagree on the unlabeled sample. Thus, the same confidence intervals used for test sets in Section 2 can be reused here.

All of the semi-supervised bounds (there are several) use the next theorem as a starting point. Here, f is typically a randomized classifier for which we have a generalization bound, while c_{final} is the final deterministic classifier learned based on all the labeled data.

Theorem 1 *(Semisupervised Derandomization) (Kääriäinen, 2005) Let* f *be a randomized classifier and* c_{final} *the final deterministic classifier, both learned based on* $S \sim D^n$. *Suppose we have a generalization error bound* $\alpha = \alpha(S, \delta)$ *for* f *satisfying* $\Pr_{S\sim D^n}(e(f, D) \leq \alpha(S, \delta/2)) \geq 1 - \delta/2$. *Then with probability at least* $1 - \delta$ *over the draw of* $S \sim D^n$,

$U \sim D_X^m$, and the randomness in f, we have

$$e(c_{final}, D) \leq \alpha(S, \delta/2) + \overline{\mathrm{Bin}}\left(\hat{d}(f, c_{final}, U), m, \delta/2\right).$$

Next we present a resampling-related bound using the above theorem. The resampling methods are based on the following technique for obtaining f and α:

1. Generate k subsamples (multisets) of the labeled sample S. Denote these by $S_1, \ldots, S_k$ and their complements in S by $S'_1, \ldots, S'_k$. The only requirement is that the choice of which examples to include in the subsamples has to be independent of the contents of S.
2. For each S_i, apply the learning algorithm to get a classifier c_i.
3. Test each c_i on S'_i. By Lemma 1, with probability $1 - \delta/(2k)$ we have
$$e(c_i, D) \leq \overline{\mathrm{Bin}}\left(\hat{e}(c_i, S'_i), |S'_i|, \delta/k\right).$$
4. Let the randomized classifier f be the classifier obtained by uniform randomization over the set $\{c_i \mid 1 \leq i \leq k\}$. The union bound ($\Pr(\cup A_i) \leq \sum \Pr(A_i)$) tells us that all the approximations in step 3 hold simultaneously with probability at least $1 - \delta/2$, which implies
$$e(f, D) = E_{c \sim f} e(c, D) = \frac{1}{k} \sum_{i=1}^{k} e(c_i, D)$$
$$\leq \frac{1}{k} \sum_{i=1}^{k} \overline{\mathrm{Bin}}\left(\hat{e}(c_i, S'_i), |S'_i|, \delta/(2k)\right)$$
with the same probability. This gives us $\alpha = \frac{1}{k} \sum_{i=1}^{k} \overline{\mathrm{Bin}}\left(\hat{e}(c_i, S'_i), |S'_i|, \delta/(2k)\right)$.

After this, the learning algorithm is applied once more to the whole labeled sample S to get the final deterministic classifier c_{final}. Then the α obtained from step 4 above and the final hypothesis c_{final} are plugged into Theorem 1, which gives the following bound for c_{final}.

Theorem 2 *For all D, f, and c_{final} as above, with probability at least $1 - \delta$ over the choice of $S \sim D^n$, $U \sim D_X^m$, and the randomness in f, we have*

$$e(c_{final}, D) \leq \frac{1}{k} \sum_{i=1}^{k} \overline{\mathrm{Bin}}\left(\hat{e}(c_i, S'_i), |S'_i|, \delta/(2k)\right) + \overline{\mathrm{Bin}}\left(\hat{d}(f, c_{final}, U), m, \delta/2\right).$$

The only variation in the resampling based bounds is the way the samples S_i are chosen in step 1. We next list a few possibilities.

Cross-Validation In k-fold cross-validation, the labeled data is split into k equisized folds S'_i (for notational simplicity, we assume k divides n). The subsample generation process is then defined by $S_i = \cup_{j \neq i} S'_j$. This plugged into Theorem 2 gives the cross-validation bound. The special case $k = n$ is the leave-one-out bound, which is of no use as $\overline{\mathrm{Bin}}(0, 1, \delta/(2n)) = 1 - \frac{\delta}{2n}$.

Bagging Bagging is a bootstrapping method introduced by Leo Breiman (Breiman, 1996a). In it each of the k subsamples S_i is generated by choosing randomly with replacement n examples from S. For each i, about a $1 - 1/e \simeq .63$ fraction of the examples are represented in S_i, while a .37 fraction remain for testing purposes in S'_i. Applying Theorem 2 to this resampling scheme gives the bagging bound.

Bagging was originally introduced as an aggregation method that enhances classification accuracy by replacing c_{final} with the *voting classifier* $c_{\mathrm{vote}}(x) = \arg\max\{P(f(x) = y) \mid y \in Y\}$ (ties are broken arbitrarily). The bagging bound holds also with the substitution $c_{\mathrm{final}} \to c_{\mathrm{vote}}$, thus giving a generalization error bound for the aggregated hypothesis c_{vote}. Breiman has introduced out-of-bag *estimates* (different from the estimate behind α in step 4 above) of the generalization performance of c_{vote} (Breiman, 1996b), but the generalization error bounds for c_{final} and c_{vote} given by Theorem 2 are new.

Semi-Supervised Test Set Bound The special case $k = 1$ corresponds to transforming a test set bound for a (deterministic) classifier learned based on some fraction S_1 of the data into a semi-supervised training set bound for the classifier c_{final}.

This bound with the choice $|S'_1| = n/k$ should be compared with the semisupervised cross-validation bound above. Except for the difference in the confidence parameter (which should have only minor relevance for large n/k), the first term of the above bound and the sum in the cross-validation bound have the same expectation.

Other Variants Besides the above, one can think of bagging with different bootstrap sample sizes, bagging where a fraction $c < 1$ of data is sampled into each S_i without replacement, weighting the classifiers nonuniformly,... For each of these subsampling strategies Theorem 2 creates a generalization error bound.

4. Online Bounds

Online bounds for generalization error work in a setting where the learner is given the labeled examples

one by one. After seeing i labeled examples, $0 \leq i < n$, the learner outputs a classifier c_i that is used in predicting the label of the example number $i+1$. It is then given the correct label, and the process of prediction and correction repeats. It is also possible to start the online process after first seeing a batch of training examples that is used in learning c_0 and the subsequent classifiers.

This way, the learner produces a sequence of classifiers $c_0, \ldots, c_{n-1}$ whose performance is often measured by the cumulative error

$$\hat{e}_{A,S} = \sum_{i=1}^{n} I(c_{i-1}(X_i) \neq Y_i)$$

where $c_{i-1} = A((x,y)^{i-1})$ is the classifier learned on the first $i-1$ samples. The cumulative error is then used in providing a generalization error bound for the randomized classifier obtained by uniform randomization over the classifiers $\{c_i \mid 0 \leq i < n\}$. The generalization error is $\bar{e}_{A,S,D}$ (the notation tracks the dependencies).

This approach has been analyzed for classification error (Blum et al., 1999; Cesa-Bianchi et al., 2001) yielding a bound that was recently tightened (Cesa-Bianchi & Gentile, 2004) (for some values, the constants are worse). A simple application of Theorem 1 transforms these into semi-supervised bounds for $c_{\text{final}} = c_n$. However, neither of these results is maximally tight. A tighter bound can be constructed by thinking of the learning algorithm and the learning distribution as an adversary which can pick any probability of error given any history of past error/not error events in a round by round fashion. The goal of the adversary is to maximize the probability of achieving some deviation between the expected and the observed number of errors, where deviation simply means the first minus the second. This optimal strategy is defined recursively.

Let $g_n(x)$ be an upper bound for the maximum probability that an adversarial algorithm/distribution pair can achieve a deviation of size at least x in n rounds. The recurrence obeys the following property:

$$g_n(x) = \max_p p g_{n-1}(x+1-p) + (1-p) g_{n-1}(x-p)$$

with the base case $g_0(x) = I(x \leq 0)$.

Calculating this recurrence, we get the plots in Figure 1 for values of g at each deviation x. The optimal choice of p can be difficult to calculate analytically as shown by the plot on the bottom. In particular, note the nonmonotonic behavior.

This can still be improved by noting that the above recurrence does not take into account the number of errors observed as the test set bound (Lemma 1) suggests. We can state another recurrence based on both n and the number of errors $\hat{e}_{A,S}$.

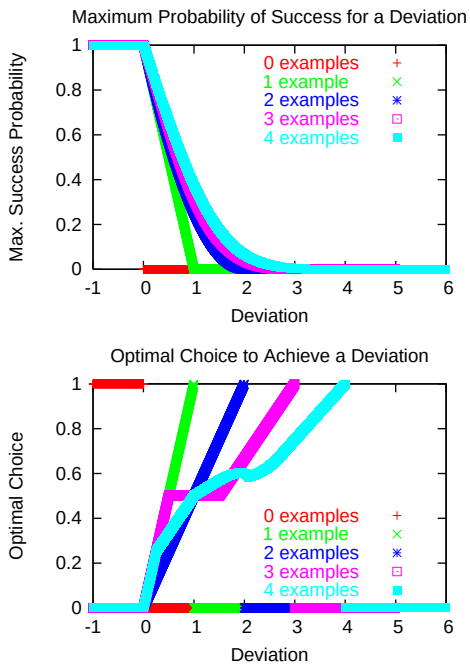

Figure 1. Top: maximum probability of a deviation x given 1, 2, 3, or 4 rounds of progressive validation. Bottom: Optimal choice of the adversary achieving these deviations.

$$g_{n,e}(x) = \max_p p g_{n-1,e-1}(x+1-p) + (1-p) g_{n-1,e}(x-p)$$

The base case is $g_{0,e}(x) = I(x \leq 0)$.

Solving this recurrence results in a bound which is somewhat worse than the Binomial confidence interval, but significantly better than other approaches as shown in Figure 2.

Many learning algorithms are designed for the online setting, and this approach is the most natural in combination with them. However, every batch mode learning algorithm can be used to define an online learning algorithm by the following simple strategy: Feed the batch algorithm the labeled examples seen so far, and use the classifier output by the algorithm to classify the next example before its label is revealed. Then, add the newly seen labeled example to the set of labeled examples, and continue till all n examples have been processed. A potential problem is that unless the batch learning algorithm is incremental in the sense it can efficiently update its hypothesis when a new example is seen, the learning algorithm has to be run n times. For large n this may take ages. To circumvent the time complexity, we can be lazy and update the

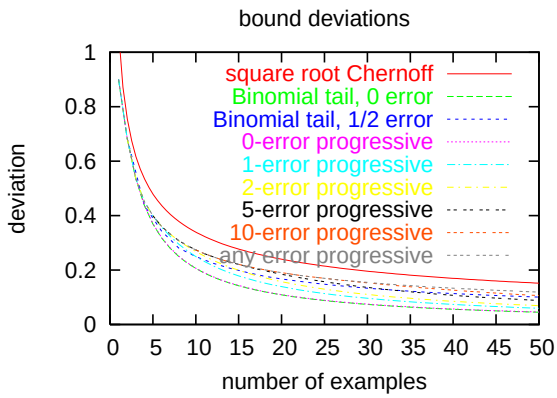

Figure 2. A comparison of bounds on the deviation between the observed and the true error rates with $\delta = 0.1$ and as the number of examples increases. "Square Root Chernoff" is the Chernoff approximation. Binomial Tail uses 1 with either 0 or 1/2 errors. The line "any error progressive" is the result of the first recursion which does not take the number of errors into account. All other lines use the second recursion which takes the errors into account.

hypothesis only every $0.1n$ samples (for example).

This strategy can be taken into account to tighten the bound calculation. Let k be the number in the batch for each update and $m = n/k$ be the number of batches. The new recurrence equations are:

$$g_{m,k,e}(x) = \max_p \sum_{i \leq min(e,k)} \binom{k}{i} (1-p)^{k-i} p^i g_{m-1,k,e-i}(x+i(1-p)-p(k-i))$$

with base case $g_{0,k,e}(x) = I(x \leq 0)$.

Computing the results of these recurrences is computationally intensive. We use an anytime approach which discretizes x at successive scales and carefully maintains the upper bound property. When this proves too intensive (as it does for very large datasets), we fall back on the looser analytic expressions above.

Theorem 3 *(Tight Progressive Validation) For all D, $n = m*k$, and all learning algorithms, with probability $1-\delta$ over the draw of $S \sim D^n$, we have:*

$$\bar{e}_{A,S,D} \leq \hat{e}_{A,S} + \max\{x : g_{m,k,\hat{e}_{A,S}}(x) \geq \delta\}$$

Proof. We start using induction to prove that for all A, x, k, m, d:

$$\Pr_{S \sim D^n} (\bar{e}_{A,S,D} \geq \hat{e}_{A,S} + x \text{ and } \hat{e}_{A,S} \leq d) \leq g_{m,k,d}(x)$$

The proof is done for any fixed A, x, k, d and inductively over m. The base case is $m = 0$ when no deviation above 0 can be achieved and A can do nothing.

For the inductive step, we have the following:

$$\Pr_{S' \sim D^{n+m}} (\bar{e}_{A,S',D} \geq \hat{e}_{A,S'} + x \text{ and } \hat{e}_{A,S} \leq d)$$

$$= \sum_{i \leq min(d,k)} \binom{k}{i} (1-p)^{k-i} p^i$$

$$\Pr_{S \sim D^n} (\bar{e}_{A,S,D} \geq \hat{e}_{A,S} + x + i(1-p) - p(k-i)$$

$$\text{and } \hat{e}_{A,S} \leq d - i)$$

where p is the probability of failure of the first classifier chosen by A (which depends on 0 examples). Using the inductive assumption, we have for all A:

$$\leq \sum_{i \leq min(d,k)} \binom{k}{i} (1-p)^{k-i} p^i g_{m,k,e-i}(x+i(1-p)-p(k-i))$$

$$\leq g_{m+1,k,d}(x)$$

Now, note that we can set $g_{m,k,d}(x) = \delta$ and solve for x to get:

$$\Pr_{S \sim D^n} (\bar{e}_{A,S,D} \geq \hat{e}_{A,S} + \max\{x : g_{m,k,d}(x) \geq \delta\}$$

$$\text{and } \hat{e}_{A,S} \leq d) \leq \delta$$

To complete the proof note that for any deviation $\max\{x : g_{m,k,d}(x) \geq \delta\}$, if $\hat{e}_{A,S} \in \{0, \ldots, d-1\}$ that also achieves deviation $\max\{x : g_{m,k,d}(x) \geq \delta\}$. Similarly, for any $\hat{e}_{A,S} \in \{d+1, \ldots, m\}$ the deviation is smaller than $\max\{x : g_{m,k,d}(x) \geq \delta\}$. □

5. PAC-Bayesian Bounds

The PAC-Bayesian theorems (McAllester, 2003a) are a relatively recent method of providing generalization error bounds for randomized classifiers. The idea is to control the generalization performance by the KL-divergence between a "prior" and a "posterior" on a set of classifiers. The "prior" may encode beliefs about which classifiers perform well, while the "posterior" describes the randomized classifier we learn.

5.1. A PAC-Bayesian Margin Bound

We work with *unbiased linear classifiers*. This representation is employed by many important learning algorithms, including, e.g., the (unbiased) support vector machines and boosting. We assume that $X = \{x \in \mathbb{R}^d \mid \|x\| = 1\}$ for some $d \in \mathbb{N}$ and that $Y = \{-1, +1\}$. A linear classifier is represented by a *weight vector* $w \in X$, which defines a classifier $f_w : X \to Y$ by $f_w(x) = sign(w \cdot x)$.

The "prior" used in the PAC-Bayesian margin bounds (Langford & Shawe-Taylor, 2002; Langford,

2005; McAllester, 2003b) is a unit-variance isotropic Gaussian on $\mathbb{R}^d$. The "posterior" $Q(w, \mu)$, where $\|w\| = 1$ and $\mu > 0$, is the normal distribution $N(\mu, 1)$ in the direction of w and standard normal in every orthogonal direction. Thus, $Q(w, \mu)$ is the prior shifted by μw.

Theorem 4 *(PAC-Bayesian margin bound) For all D, with probability at least $1 - \delta/2$ over the choice of the labeled data and the randomness in $Q(w, \mu)$, we have for all $w \in \mathbb{R}^d$, $\|w\| = 1$, $\mu \in [0, \infty)$ that*

$$KL\left(\hat{e}(Q(w,\mu), S) || e(Q(w,\mu), D)\right) \leq \frac{\frac{\mu^2}{2} + \ln \frac{m+1}{\delta/2}}{m}.$$

To derandomize the above, let us choose w to be the normalized weight vector learned by the learning algorithm, and let α be the smallest upper bound for $e(Q(w, \mu), D)$ that can be obtained from the above by the optimal choice μ^* for μ. Note that greedily picking μ^* for μ may not be the best choice for the derandomized bound, but we still get the following.

Theorem 5 *With probability at least $1 - \delta$ over the choice of $S \sim D^n$, $U \sim D_X^m$, and the randomness in $Q(w, \mu)$, we have*

$$e(f_w, D) \leq \alpha + \overline{\mathrm{Bin}}\left(\hat{d}(Q(w, \mu^*), f_w, U), m, \delta/2\right).$$

For details on how the bound of Theorem 4 can be evaluated and how μ^* can be found efficiently, the reader is referred to (Langford, 2005). The remaining difficulty in evaluating the bound of Theorem 5 is computing $\hat{d}(Q(w, \mu^*), f_w, U)$ which can be done with techniques similar to those presented in (Langford, 2005).

6. Empirical Results

In this section we summarize results of experiments obtained by `semibound` (available at `http://hunch.net/semibound`), a package of scripts that makes computing these bounds easy. The package currently supports `libsvm`, `C4.5`, `svmlight` (Joachims, 1999), and the classification algorithms in `Weka`, but can easily be extended to work with other learning algorithms as well. As datasets we used benchmark datasets from the UCI repository (Blake & Merz, 1998) (for C4.5) and from the libsvm tools page (Chang & Lin, 2005) (for `libsvm` and `svmlight`). The problem `mnist0` is the "0 versus rest" version of `mnist`, and `mnist0-10000` is a 10000 example subsample of it. Unlabeled data was obtained by forgetting the labels of 10% of the original labeled data set. This "unlabeled" data was also used in computing empirical error rates for the classifiers.

The results of the experiments are summarized in Tables 1 and 2. In Table 1, we report the observed error rates (on the "unlabeled" data set) and error bounds for the randomized classifiers, whereas Table 2 summarizes the corresponding semi-supervised bounds for c_{final}. All numbers in the tables are averages over ten runs of `semibound`, where the randomization of the split of the data to labeled and unlabeled parts as well as the randomization inside the bounds was done anew each time. In all bounds we chose $\delta = 0.01$.

The columns correspond to bounds presented in the text with the prefix R- indicating a randomized[1] and S- a semi-supervised bound. In the bounds Test and Prog we used a 90%/10% train/test split and in CV and Bag we chose $k = 10$. In Prog, the hypothesis was updated only 10 times during the validation phase to keep the computation time in control. For the same reason, the bound in the table is the minimum[2] of the analytic approximation and the upper bound to the solution of the recurrence obtained in one hour of computation. The PAC-Bayesian margin bound and algorithm are applicable to unbiased linear classifiers only, which explains the NAs in the P-B columns. With the exception of Prog, evaluating the bounds is relatively fast when compared to the time spent in the learning algorithm.

After committing ourselves to the above parameter combinations we experimented with others to see how stable the bounds are with respect to the parameters, e.g., the k in the bagging and cross-validation bounds. The randomized bounds perform best when k is small, because the slack introduced through the inverse binomial tail increases with k. In the semi-supervised bounds this increase is compensated by the decrease of $d(f, c_{\text{final}})$ with k, so the optimal value of the bound is attained with k around 5 or 10.

From Table 1 we see that R-Bag seems to produce the best bounds (even better than the baseline test set bound R-Test), whereas the randomized classifier related to the bound is seldom the one with best error rate. In bagging about 37% of the labeled data remains for testing purposes, so the test set bound on which the bagging bound built is close to the true error. However, the larger test set means less data for training, which shows up in increased error rates on the unlabeled data. In summary, R-Test and R-Prog appear to provide the best error rates, whereas

[1] The exception is R-Test which is the standard test set bound for a deterministic classifier.

[2] It is not, normally, valid to take the minimum of two bounds in this manner without increasing the value of δ. It is valid here because the recursion dominates the other bounds.

Table 1. Results on experiments with bounds for randomized classifiers on various (algorithm, data set) combinations. Each cell contains a "error rate"/"bound" combination.

Dataset/Alg	R-Test	R-Prog	R-Bag	R-CV	R-P-B
australian/svmlight	**13.04**/31.12	13.91/33.18	13.77/**23.29**	15.36/32.30	21.88/42.15
breast-cancer/svmlight	**1.45**/11.55	2.17/10.49	1.88/**8.60**	2.17/16.29	8.12/25.61
census-income/c4.5	4.61/**4.93**	4.60/5.92	4.82/5.03	**4.59**/5.01	NA/NA
covtype/c4.5	5.69/**5.91**	**5.61**/6.52	6.93/7.04	5.69/6.00	NA/NA
crx/c4.5	**10.14**/25.40	11.45/29.34	11.45/**24.83**	10.72/33.48	NA/NA
diabetes/svmlight	**19.22**/36.90	19.74/43.04	20.65/**33.63**	21.30/42.71	27.27/42.04
dna/c4.5	7.46/12.61	**7.08**/13.16	8.56/**11.29**	7.30/13.70	NA/NA
dna/libsvm	5.45/7.98	5.27/8.65	5.02/**6.86**	**4.76**/8.78	NA/NA
fourclass/svmlight	19.54/28.75	18.51/32.37	**15.40**/**22.98**	19.77/32.23	25.17/41.67
german.numer/svmlight	30.90/42.27	**30.60**/42.39	33.90/**38.20**	31.80/47.19	47.90/53.87
heart/svmlight	**20.37**/50.10	21.85/52.54	23.33/**36.56**	**20.37**/52.91	42.59/52.69
hypo/c4.5	0.74/1.70	**0.63**/1.79	0.98/**1.46**	**0.63**/3.03	NA/NA
letter/c4.5	12.50/14.29	**11.76**/**13.22**	14.57/16.02	12.37/14.87	NA/NA
letter/libsvm	2.33/**3.04**	**2.20**/3.93	2.91/3.66	2.21/3.48	NA/NA
mnist/libsvm	1.72/**2.09**	**1.70**/4.34	1.99/2.28	1.73/2.29	NA/NA
mnist0/svmlight	**0.29**/0.46	**0.29**/0.83	**0.29**/**0.43**	**0.29**/0.57	0.74/2.52
mnist0-10000/svmlight	**0.57**/1.21	0.58/1.16	0.60/**1.02**	0.62/1.75	1.62/5.75
monk1/c4.5	**0.00**/8.80	0.36/10.88	2.32/**7.50**	1.61/18.01	NA/NA
monk2/c4.5	**40.66**/58.49	41.31/59.74	41.64/**49.81**	41.64/58.17	NA/NA
monk3/c4.5	**1.79**/8.80	**1.79**/10.88	**1.79**/**5.82**	**1.79**/15.00	NA/NA
satimage/c4.5	**13.20**/**16.50**	13.82/18.50	15.19/17.68	14.13/18.72	NA/NA
satimage/libsvm	7.64/12.16	**7.55**/13.17	8.82/**10.80**	7.81/11.88	NA/NA
segment/c4.5	1.04/10.92	1.13/9.23	2.25/**7.65**	**1.00**/10.63	NA/NA
segment/libsvm	3.90/6.86	3.72/8.06	3.59/**5.60**	**3.42**/8.21	NA/NA
shuttle/c4.5	**0.04**/0.14	**0.04**/0.17	0.06/**0.12**	0.05/0.21	NA/NA
shuttle/libsvm	**0.05**/0.24	**0.05**/0.31	0.07/**0.20**	**0.05**/0.29	NA/NA
soybean/c4.5	**6.67**/**15.75**	**6.67**/18.83	9.57/18.94	8.70/26.44	NA/NA
usps/libsvm	1.84/**3.17**	1.75/3.42	1.99/3.32	**1.71**/4.03	NA/NA
vote/c4.5	4.55/22.99	4.55/30.40	**2.05**/**13.95**	2.95/25.49	NA/NA

R-Bag provides the best bounds. Still, it is hard to draw any definitive conclusions of the relative performance of the methods, with the exception of R-P-B being clearly worst.

The case of R-Bag is an example of a more general trade-off: For good prediction performance, we would like to use all data in training, while the easiest way to get good bounds is to leave a part of the data aside for testing. In case prediction accuracy is the most important thing, one can use unlabeled data to transform the bounds for randomized classifiers in Table 1 to bounds for c_{final}. The results of this transformation are presented in Table 2, in which the column Error is the error rate of c_{final}. The cost of the transformation is roughly the distance $d(f, c_{\text{final}})$ between f and c_{final}, which varies significantly from problem to problem. The results are impressive as compared to traditional training set bounds for c_{final}, but still not as good as that of the randomized bounds. Here S-Test appears to win, but the relative order of the other bounds is again not clear.

7. Discussion

What these experiments illustrate is that there is a natural trade-off between the goal of good prediction and the goal of good confidence about prediction ability. We can state and use bounds which operate over the range of this tradeoff, but no bound appears to dominate for both goals simultaneously.

The good news is that some of these nonstandard tradeoff points are entirely viable for common practice. Walking over the tradeoff, we get the following prescription for common practice:

1. Prediction ability marginally important, confidence very important: Use the a large test set bound or the randomized bagging bound.
2. Prediction ability somewhat important, confidence very important: Use some other randomized bound, e.g., the test set bound or the Progressive Validation bound.
3. Prediction ability very important, confidence somewhat important, unlabeled data available: use the semisupervised versions of these bounds.

If randomness in a classifier presents problems, then

Table 2. Results on experiments with semi-supervised bounds for c_{final} on various (algorithm, data set) combinations.

Dataset/Alg	Error	S-Test	S-Prog	S-Bag	S-CV	S-P-B
australian/svmlight	15.51	45.16	46.15	**38.64**	43.27	73.52
breast-cancer/svmlight	2.17	21.58	19.03	**17.91**	25.37	46.03
census-income/c4.5	4.60	**5.80**	6.70	7.12	5.84	NA
covtype/c4.5	5.45	11.98	**11.81**	14.87	11.91	NA
crx/c4.5	11.59	**36.93**	38.48	47.03	47.38	NA
diabetes/svmlight	21.04	**49.43**	54.21	50.52	54.49	71.03
dna/c4.5	6.58	20.55	**19.26**	22.07	22.26	NA
dna/libsvm	4.64	11.60	12.41	**11.19**	12.41	NA
fourclass/svmlight	18.39	**38.33**	40.39	41.67	43.46	77.05
german.numer/svmlight	30.90	55.65	**52.97**	62.18	59.42	113.74
heart/svmlight	20.74	78.75	75.02	**65.68**	77.85	113.20
hypo/c4.5	0.58	3.64	3.57	**3.55**	4.87	NA
letter/c4.5	11.88	26.81	**23.26**	34.47	27.03	NA
letter/libsvm	2.02	**4.44**	5.05	6.55	4.76	NA
mnist/libsvm	1.65	**2.84**	5.05	3.68	3.03	NA
mnist0/svmlight	0.28	**0.62**	0.97	0.63	0.70	3.43
mnist0-10000/svmlight	0.61	1.97	1.88	**1.78**	2.58	8.42
monk1/c4.5	0.00	**19.08**	22.15	22.24	30.36	NA
monk2/c4.5	45.08	82.44	**81.21**	100.83	82.10	NA
monk3/c4.5	1.79	19.08	21.45	**15.50**	25.27	NA
satimage/c4.5	13.98	**32.31**	32.99	37.95	35.73	NA
satimage/libsvm	7.66	16.71	17.01	18.26	**15.98**	NA
segment/c4.5	0.65	16.18	**14.47**	14.85	16.02	NA
segment/libsvm	3.46	**11.28**	11.91	12.14	12.88	NA
shuttle/c4.5	0.05	**0.29**	0.30	0.31	0.35	NA
shuttle/libsvm	0.04	**0.36**	0.44	0.38	0.42	NA
soybean/c4.5	10.58	**32.29**	34.98	41.82	43.70	NA
usps/libsvm	1.70	**4.38**	4.41	5.27	5.22	NA
vote/c4.5	2.27	40.54	48.30	**32.26**	40.59	NA

the test set bound (in case confidence is more important than accuracy) or the semi-supervised bounds (when tightness of the bound is less important than the accuracy of the final hypotheisis) should be preferred.

Acknowledgements

We would like to thank Tom Hayes who helped substantially with early discussions that turned into the improved progressive validation bound.

References

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Blum, A., Kalai, A., & Langford, J. (1999). Beating the hold-out: bounds for k-fold and progressive cross-validation. *COLT '99* (pp. 203–208).

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

Breiman, L. (1996b). Out-of-bag estimation. Manuscript.

Cesa-Bianchi, N., Conconi, A., & Gentile, C. (2001). On the generalization ability of on-line algorithms. *NIPS* (pp. 359–366).

Cesa-Bianchi, N., & Gentile, C. (2004). Improved risk tail bounds for on-line algorithms. A presentation in the (Ab)use of Bounds workshop.

Chang, C.-C., & Lin, C.-J. (2005). libsvm tools. `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/`.

Joachims, T. (1999). Making large-scale SVM learning practical. In B. Schölkopf, C. Burges and A. Smola (Eds.), *Advances in kernel methods – support vector learning*. MIT-Press.

Kääriäinen, M. (2005). Generalization error bounds using unlabeled data. *COLT '05*. To appear.

Langford, J. (2005). Practical prediction theory for classification. *Journal of Machine Learning Research*. To appear.

Langford, J., & Shawe-Taylor, J. (2002). PAC-Bayes & margins. *NIPS* (pp. 423–430).

McAllester, D. A. (2003a). PAC-Bayesian stochastic model selection. *Machine Learning*, *51*, 5–21.

McAllester, D. A. (2003b). Simplified PAC-Bayesian margin bounds. *COLT '03* (pp. 203–215).