# TUTORIAL ON PRACTICAL PREDICTION THEORY FOR CLASSIFICATION

JOHN LANGFORD, TTI-CHICAGO

ABSTRACT. We discuss basic prediction theory and its impact on classification success evaluation, implications for learning algorithm design, and uses in learning algorithm execution. This tutorial is meant to be a comprehensive compilation of results which are both theoretically rigorous and practically useful.

There are a two important implications of the results presented here:

(1) Common practices for reporting results in classification should change to use the test set bound.

(2) Train set bounds can sometimes be used to directly motivate learning algorithms.

## 1. INTRODUCTION

Classifiers are functions which partition a set into two classes (the set of rainy days and the set of sunny days). Classifiers appear to be the most simple nontrivial decision making element so their study often has implications for other learning systems. Classifiers are sufficiently complex that many phenomena observed in machine learning (theoretically or experimentally) can be observed in the classification setting. Yet, classifiers are simple enough to make their analysis easy to understand. This combination of sufficient yet minimal complexity for capturing phenomena makes the study of classifiers especially fruitful.

The goal of this paper is an introduction to the theory of prediction for classification. Here "prediction theory" means statements about the future error rate of learned classifiers. A typical statement has the form, "With probability $1 - \delta$ over an i.i.d. draw of some sample, the expected future error rate of a classifier is bounded by $f(\delta, \text{error rate on sample})$". These statements are confidence intervals on the error rate of a learned classifier. Many of these results have been presented elsewhere, although the style, tightness, and generality of the presentation are often new here (and particularly oriented towards practical use). This is a tutorial, so we limit our presentation to those results which are both theoretically sound and practically useful.

There are several important aspects of learning which the theory here casts light on. Perhaps the most important of these is the problem of performance reporting for classifiers. Many people use some form of empirical variance to estimate upper and lower bounds. This is an error-prone practice, and the test set bound in section 3 implies a better method by nearly any metric. Hopefully, this will become common practice.

After discussing the test set bound we cover the Occam's Razor bound, the simplest train set bound, which explains (and quantifies) the common phenomena

of overfitting. We also prove that the Occam's Razor bound cannot be improved without incorporating extra information and apply the bound to decision trees.

Next, we discuss two train set bounds, the PAC-Bayes bound and the Sample Compression bound, which have proved to give practical results for more general classifiers, such as Support Vector Machines and Neural Networks.

All of the results here should be easily approachable and understandable. The proofs are simple, and examples are given. Pointers to related work are also given.

It is important to note that all of the results presented here fall in the realm of classical statistics. In particular, all randomizations are over draws of the data, and our results have the form of confidence intervals.

The layout of this document is as follows:

- Section 2 presents the formal model
- Section 3 presents the test set bound
- Section 4 presents the Occam's Razor bound
- Section 5 presents the PAC-Bayes bound
- Section 6 presents the Sample Compression bound

The formal model and test set bound must be understood in order to appreciate all later results. There is no particular dependency between the various train set bounds we present.

## 2. Formal Model

There are many somewhat arbitrary choices of learning model. The one we use can (at best) be motivated by its simplicity. Other models such as the online learning model [10], PAC learning [22], and the uniform convergence model [23] differ in formulation, generality, and in the scope of addressable questions. The strongest motivation for studying the prediction theory model here is simplicity and corresponding generality of results. Appendix section 7.3 discusses the connections between various models.

### 2.1. Basic quantities.

We are concerned with a learning model in which examples of (input, output) pairs come independently from some unknown distribution. The goal is to find a function capable of predicting the output given the input. There are several mathematical objects we work with.

| Object | Description |
|---|---|
| $X$ | The (arbitrary) space of the input to a classifier |
| $Y = \{-1, 1\}$ | The output of a classification. |
| $D$ | An (unknown) distribution over $X \times Y$ |
| $S$ | A set of examples drawn independently from $D$. |
| $m$ | $= \|S\|$ the number of examples |
| $c$ | A function mapping $X$ to $Y$ |

There are several distinctions between this model and other (perhaps more familiar) models. There is no mention of a classifier space, because the results do not depend upon a classifier space. Also, the notion of a distribution on $X \times Y$ is strictly more general than the "target concept" model which assumes that there exists some function $f : X \to Y$ used to generate the label [22]. In particular we can model noisy learning problems which do not have a particular $Y$ value for each

$X$ value. This generalization is essentially "free" in the sense that it does not add to the complexity of presenting the results.

It is worth noting that the *only* unverifiable assumption we make is examples are drawn independently from $D$. The strength of all the results which follow rests upon the correctness of this assumption.

Sometimes, we decorate these objects with labels like $S_{\text{train}}$ (a train set) or $S_{\text{test}}$ (a test set). These decorations should always be clear.

**Example 2.1.** Weather prediction: Will it rain today or not? In this case $X =$ barometric pressure, observations of cloud cover or other sensory input and $Y = 0$ if the prediction is "no rain" and 1 otherwise. The distribution $D$ is over sensory inputs and outcomes. The sample set $S$, might consist of $m = 100$ (observation, outcome) pairs such as (pressure low, cloudy, rain), (pressure high, cloudy, not rain), etc. A classifier, $c$, is any function which predicts "rain" or "not rain" based upon the observation.

Note that the independence assumption here is not perfectly satisfied although it seems to be a reasonable approximation for well-seperated days. In any application of this theory, it must be carefully judged whether the independence assumption holds or not.

2.2. **Derived quantities.** There are several derived quantities which the results are stated in terms of.

**Definition 2.2.** (True Error) The true error $c_D$ of a classifier $c$ is defined as the probability that the classifier errs:

$$c_D \equiv \Pr_{(x,y)\sim D}(c(x) \neq y)$$

The true error is sometimes called the "generalization error". Unfortunately, the true error is not an observable quantity in our model because the distribution, $D$, is unknown. However, there is a related quantity which is observable.

**Definition 2.3.** (Empirical Error) Given a sample set $S$, the *empirical error*, $\hat{c}_S$ is the observed rate of errors:

$$\hat{c}_S \equiv \Pr_{(x,y)\sim S}(c(x) \neq y) = \frac{1}{m}\sum_{i=1}^{m} I(c(x_i) \neq y_i)$$

where $I()$ is a function which maps "true" to 1 and "false" to 0. Also, $\Pr_{(x,y)\sim S}(...)$ is a probability taken with respect to the uniform distribution over the set of examples, $S$.

The empirical error is sometimes called the "training error", "test error", or "observed error" depending on whether it is the error rate on a training set, test set, or a more general set.

**Example.** (continued) The classifier $c$ which always predicts "not rain" might have an empirical error of $\frac{38}{100}$ and an unknown true error rate (which might in fact be 0.5).

2.3. **Addressable questions.** Given the true error, $c_D$ of a classifier $c$ we can precisely describe the distribution of success and failure on future examples drawn according to $D$. This quantity is derived from the unknown distribution $D$, so our

effort is directed toward upper and lower bounding the value of $c_D$ for a classifier $c$.

The variations in all of the bounds that we present are related to the method of choosing a classifier, $c$. We cover two types of bounds:

(1) Test: Use examples in a test set which were not used in picking $c$.
(2) Train: Use examples for both choosing $c$ and evaluating $c$.

These methods are addressed in the next two sections.

It is worth noting that one question that *cannot* be addressed in this model is "Can learning occur for my problem?". Extra assumptions (as in [22] [23]) are inherently necessary.

## 3. The Test Set Method

The simplest bound arises for the classical technique of using $m$ fresh examples to evaluate a classifier. This section is organized into two subsections:

- Subsection 3.1 presents the basic upper bound on the true error rate, handy approximations, and a lower bound
- Subsection 3.2 discusses the implications of the test set bound on error reporting practice. A better method for error reporting is applied to several datasets and the results are shown.

### 3.1. The Bound.

Before stating the bound, we note a few basic observations which make the results less surprising. The principal observable quantity is the empirical error $\hat{c}_S$ of a classifier. What is the distribution of the empirical error for a fixed classifier? For each example, our independence assumption implies the probability that the classifier makes an error is given by the true error, $c_D$. This can be modeled by a biased coin flip: heads if you are right and tails if you are wrong.

What is the probability of observing $k$ errors (heads) out of $m$ examples (coin flips)? This is a very familiar distribution in statistics called the Binomial and so it should not be surprising that the bounds presented here are fundamentally dependent upon the cumulative distribution of a Binomial. For the following definition $\mathcal{B}(p)$ is the distribution of a Bernoulli coin flip.

**Definition 3.1.** (Binomial Tail Distribution)

$$\text{Bin}\left(\frac{k}{m}, c_D\right) \equiv \Pr_{Z_1,\ldots Z_m \sim \mathcal{B}(c_D)^m}\left(\sum_{i=1}^{m} Z_i \leq k\right) = \sum_{j=0}^{k} \binom{m}{j} c_D^j (1-c_D)^{m-j}$$

$=$ the probability that $m$ examples (coins) with error rate (bias) $c_D$ produce $k$ or fewer errors (heads).

A depiction of the Binomial distribution is given in figure 3.1.

For the learning problem, we always choose a bias of $c_D$ and $X_i$ =error or not on the $i$th example. With these definitions, we can interpret the Binomial tail as the probability of an empirical error greater than or equal to $\frac{k}{m}$.

Since we are interested in calculating a bound on the true error given a confidence $\delta$, and an empirical error $\hat{c}_S$, it is handy to define the inversion of a Binomial tail.

**Definition 3.2.** (Binomial Tail Inversion)

$$\overline{\text{Bin}}\left(\frac{k}{m}, \delta\right) \equiv \max_p \left\{p : \text{ Bin}\left(\frac{k}{m}, p\right) \geq \delta\right\}$$

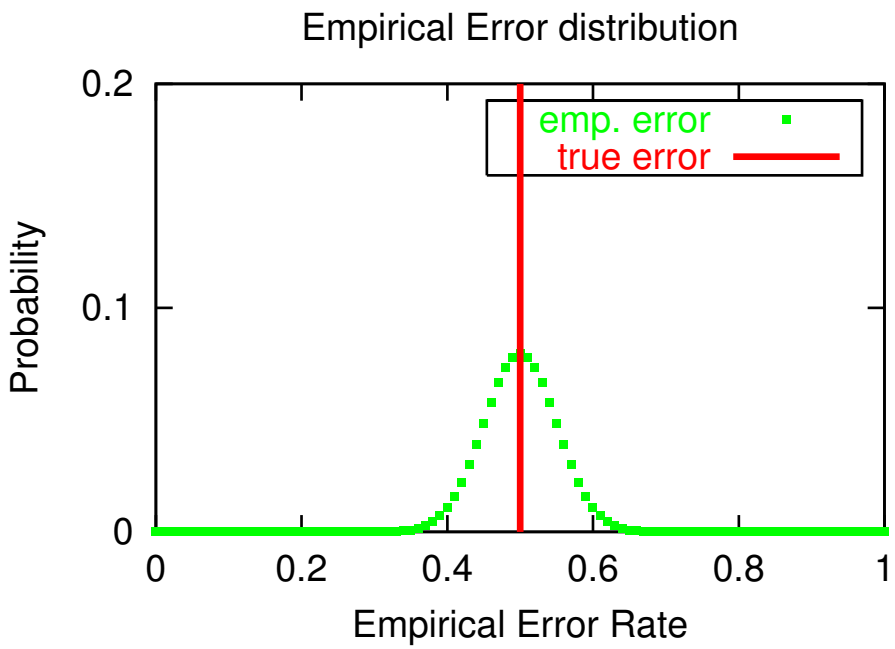


FIGURE 3.1. A depiction of the Binomial distribution. The cumulative of the Binomial is the area under the curve up to some point on the horizontal axis.

= the largest true error such that the probability of observing $\frac{k}{m}$ or more "heads" is at least $\delta$.

With these definitions finished, the results are all very simple statements.

**Theorem 3.3.** *(Test Set Bound) For all classifiers, $c$, for all $\delta \in (0, 1]$*

$$\Pr_{S \sim D^m}\left(c_D \leq \overline{Bin}(\hat{c}_S, \delta)\right) \geq 1 - \delta$$

Note that $m$ in this equation is $m_{\text{test}} = |S_{\text{test}}|$, the size of the test set.

*Proof.* (pictorially in 3.2) The proof is just a simple identification with the Binomial. For any distribution over $(x, y)$ pairs and any classifier, $c$, there exists some probability, $c_D$, that the classifier predicts incorrectly. We can regard this event as a coin flip with bias $c_D$. Since each example is picked independently, the distribution of the empirical error is a Binomial distribution.

Whatever our true error $c_D$ is, with probability $1 - \delta$ the observation $\hat{c}_S$ will not fall into a tail of size $\delta$. Assuming (correctly with probability $1 - \delta$) that the empirical error is not in the Binomial tail, we can constrain (and therefore bound) the value of the true error $c_D$. □

The test set bound is, essentially, perfectly tight. For any classifier with a sufficiently large true error, the bound is violated exactly a $\delta$ portion of the time.

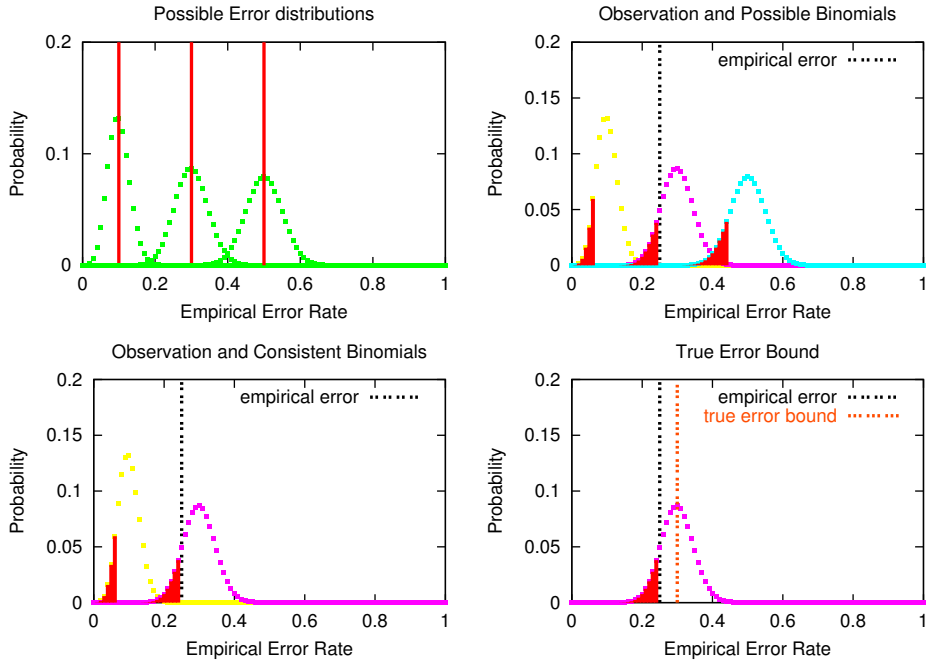


FIGURE 3.2. A graphical depiction of the test set bound. The first graph depicts several possible Binomials given their true error rates. The second depicts several Binomials, each with a tail cut. The third figure shows the Binomials consistent with the tail cut and observed test error. The worst case over all true error rates is the consistent Binomial with the largest bias.

3.1.1. *Approximations.* There are several immediate corollaries of the test set bound (3.3) which are more convenient when a computer is not handy. The first corollary applies to the limited "realizable" setting where you happen to observe 0 test errors.

**Corollary 3.4.** *(Realizable Test Set Bouund) For all classifiers, $c$, for all $\delta \in (0, 1]$*

$$\Pr_{S \sim D^m} \left( \hat{c}_S = 0 \Rightarrow c_D \leq \frac{\ln \frac{1}{\delta}}{m} \right) \geq 1 - \delta$$

*Proof.* Specializing theorem 3.3 to the zero empirical error case, we get:

$$\text{Bin}\left(\frac{0}{m}, \epsilon\right) = (1-\epsilon)^m \leq e^{-\epsilon m}$$

Setting this equal to $\delta$ and solving for $\epsilon$ gives us the result. The last inequality can be most simply motivated by comparing graphs as in figure 3.3.

□

Approximations which hold for arbitrary (nonzero) error rates rely upon the Chernoff bound which we state next, for completeness.

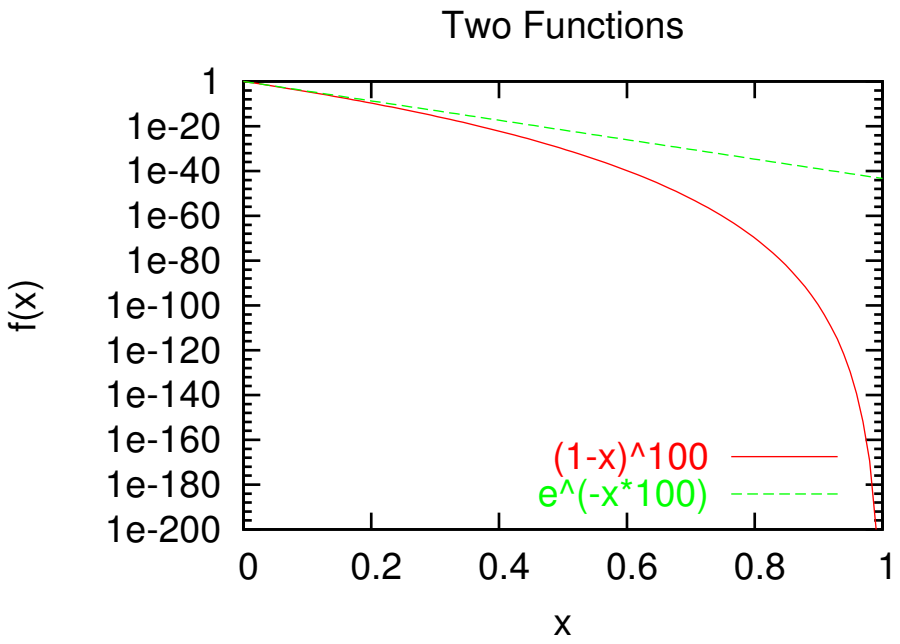


FIGURE 3.3. A graph suggesting $e^{-\epsilon m} \geq (1-\epsilon)^m$.

**Lemma 3.5.** *(Relative Entropy Chernoff Bound)*[1] *For* $\frac{k}{m} < p$*:*

$$Bin\left(\frac{k}{m}, p\right) \leq e^{-mKL\left(\frac{k}{m}||p\right)}$$

*where* $KL\left(\frac{k}{m}||p\right) = \frac{k}{m}\log\frac{\frac{k}{m}}{p} + \left(1-\frac{k}{m}\right)\log\frac{1-\frac{k}{m}}{1-p}$ *for* $p > \frac{k}{m}$ *and* $0$ *otherwise.*

*Proof.* (Originally from [3]. The proof here is based on [21]) For all $\lambda > 0$, we have:

$$\text{Bin}\left(\frac{k}{m}, p\right) = \Pr_{X^m \sim p^m}\left(\frac{1}{m}\sum_{i=1}^{m} X_i \leq \frac{k}{m}\right) = \Pr_{X^m \sim p^m}\left(e^{-m\lambda\frac{1}{m}\sum_{i=1}^{m} X_i} \geq e^{-m\lambda\frac{k}{m}}\right)$$

Using Markov's inequality ($X \geq 0$, $EX = \mu$, $\Rightarrow \Pr(X \geq \delta) \leq \frac{\mu}{\delta}$):

$$\leq \frac{E_{X^m \sim p^m} e^{-\lambda\sum_{i=1}^{m} X_i}}{e^{-\lambda k}}$$

Using independence, we get:

$$= e^{\lambda k}\left(pe^{-\lambda} + (1-p)\right)^m$$

and rewriting, we get:

$$= e^{mf(\lambda)}$$

where $f(\lambda) = \lambda\frac{k}{m} + \ln\left(pe^{-\lambda} + 1 - p\right)$.

[1]The closely related Hoeffding bound[7] makes the same statement for sums of $[0,1]$ random variables.

$\lambda$ is a free parameter which can be optimized to find the tightest possible bound. To find the optimal value, find $\lambda^*$ so that $f'(\lambda^*) = 0$.

$$\begin{aligned} 0 = f'(\lambda^*) &= \frac{k}{m} - \frac{pe^{-\lambda^*}}{pe^{-\lambda^*} + 1 - p} \\ \Rightarrow \frac{\frac{k}{m}}{p}\left(pe^{-\lambda^*} + 1 - p\right) &= e^{-\lambda^*} \\ \Rightarrow \frac{\frac{k}{m}}{p}(1-p) &= \left(1 - \frac{k}{m}\right)e^{-\lambda^*} \\ \Rightarrow e^{\lambda^*} &= \frac{p\left(1-\frac{k}{m}\right)}{\frac{k}{m}(1-p)} \end{aligned}$$

which is valid for $p > \frac{k}{m}$. Using this, we get:

$$\begin{aligned} f(\lambda^*) &= \frac{k}{m}\ln\frac{p\left(1-\frac{k}{m}\right)}{\frac{k}{m}(1-p)} + \ln\left(\frac{1-p}{1-\frac{k}{m}}\right) \\ &= \frac{k}{m}\ln\frac{p}{\frac{k}{m}} + \left(1-\frac{k}{m}\right)\ln\left(\frac{1-p}{1-\frac{k}{m}}\right) \\ &= -\mathrm{KL}\left(\frac{k}{m}||p\right) \end{aligned}$$

□

Using the Chernoff bound, we can loosen the test set bound to achieve a more analytic fom.

**Corollary 3.6.** *(Agnostic Test Set Bound) For all classifiers, c, for all $\delta \in (0,1]$*

$$\Pr_{S\sim D^m}\left(KL\left(\hat{c}_S||c_D\right) \leq \frac{\ln\frac{1}{\delta}}{m}\right) \geq 1-\delta$$

*where $KL(q||p) = q\ln\frac{q}{p} + (1-q)\ln\frac{1-q}{1-p}$ is the Kullback-Leibler divergence between two coins of bias $q,p$ with $q < p$.*

*Proof.* Loosening theorem 3.3 with the Chernoff approximation for $\frac{k}{m} < c_D$ we get:

$$\mathrm{Bin}\left(\frac{k}{m}, c_D\right) \leq e^{-m\mathrm{KL}\left(\frac{k}{m}||c_D\right)}$$

setting this equal to $\delta$, and solving for $\epsilon$ gives the result. □

The agnostic test set bound can be further loosened by bounding the value of $\mathrm{KL}(q||p)$.

**Corollary 3.7.** *(Agnostic Test Set Bound II) For all classifiers, c, for all $\delta \in (0,1]$*

$$\Pr_{S\sim D^m}\left(c_D \leq \hat{c}_S + \sqrt{\frac{\ln\frac{1}{\delta}}{2m}}\right) \geq 1-\delta$$

*Proof.* Use the approximation:

$$\mathrm{KL}(\frac{k}{m}||c_D) \geq 2(c_D - \frac{k}{m})^2$$

with the Chernoff bound and Test set bounds to get the result. □

The differences between the agnostic and realizable case are fundamentally related to the decrease in the variance of a Binomial as the bias (i.e. true error) approaches 0. Note that this implies using the exact Binomial tail calculation can result in *functional* (rather than merely constant) improvements on the above corollary.

3.1.2. *A Test Set Lower Bound.* The true error can be lower bounded using a symmetric application of the same techniques.

**Theorem 3.8.** *(Test Set Lower Bound) For all classifiers, c, for all* $\delta \in (0, 1]$

$$\Pr_{S \sim D^m} \left( c_D \geq \min_p \{p : \ 1 - Bin(\hat{c}_S, p) \geq \delta\} \right) \geq 1 - \delta$$

The proof is completely symmetric. Note that both bounds hold with probability $1 - 2\delta$ since $\Pr(A \text{ or } B) \leq \Pr(A) + \Pr(B)$. This is particularly convenient when the square-root version of the Chernoff approximation is used in both directions to get:

$$\forall c \ \Pr_{S \sim D^m} \left( |c_D - \hat{c}_S| \leq \sqrt{\frac{\ln \frac{2}{\delta}}{2m}} \right) \geq 1 - \delta$$

**Example.** (continued) let $\delta = 0.1$. Using the square root Chernoff bound with $\hat{c}_S = \frac{38}{100}$, we get the confidence interval $c_D \in [0.26, 0.50]$. Using an exact calculation for the Binomial tail, we get: $c_D \in [0.30, 0.47]$. In general, as the observed error moves toward 0, the exact calculation provides a tighter confidence interval than the agnostic approximation.

3.1.3. *The state of the art.* Although the test set bound is very well understood, the same cannot be said of other testing methods. Only weak general results in this model are known for some variants of cross validation (see [2]). For specific learning algorithms (such as nearest neighbor), stronger results are known (see [4]). There are a wide range of essentially unanalyzed methods and a successful analysis seems particularly tricky although very worthwhile if completed.

3.2. **Test Set Bound Implications.** There are some common practices in machine learning which can be improved by application of the test set bound. When attempting to calculate a confidence interval on the true error rate given the test set, many people follow a standard statistical prescription:

(1) Calculate the empirical mean $\hat{\mu} = \hat{c}_{S_{\text{test}}} = \frac{1}{m} \sum_{i=1}^{m} I(h(x_i) \neq y_i)$.
(2) Calculate the empirical variance $\hat{\sigma}^2 = \frac{1}{m-1} \sum_{i=1}^{m} (I(c(x_i) = y_i) - \hat{\mu})^2$.
(3) Pretend that the distribution is Gaussian with the above variance and construct a confidence interval by cutting the tails of the Gaussian cumulative distribution at the $2\hat{\sigma}$ (or some other) point.

This approach is motivated by the fact that for any *fixed* true error rate, the distribution of the observed accuracy behaves like a Gaussian *asymptotically.* Here, asymptotically means "in the limit as the number of test examples goes to infinity".

The problem with this approach is that it leads to fundamentally misleading results as shown in figure 3.4. To construct this figure, a collection of discrete (aka "nominal) feature datasets from the UCI machine learning database were split into a training and test sets. A decision tree classifier was learned on each training set and then evaluated on the held-out test set.

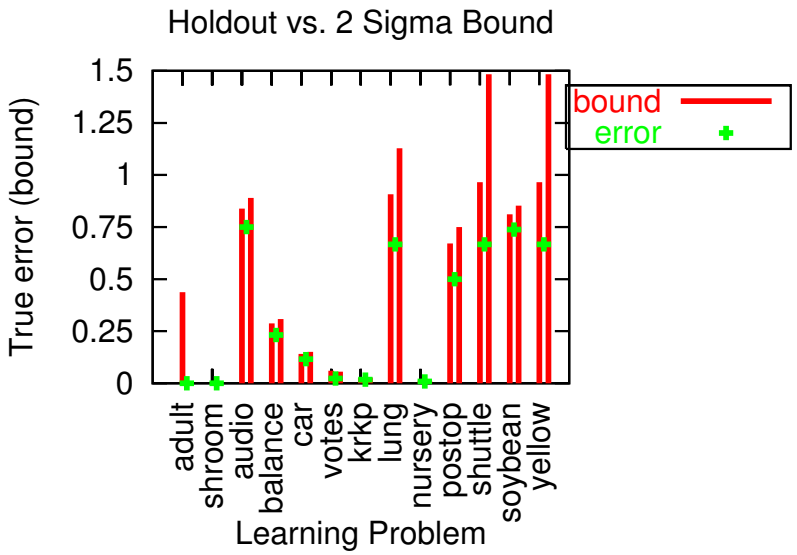


FIGURE 3.4. This is a graph of the confidence intervals implied by the test set bound (3.3) on the left, and the approximate confidence intervals implied using the common two sigma rule motivated by asymptotic normality on the right. The upper bounds of the test set bound have $\delta = 0.025$ failure rate, so as to be comparable with the 2-sigma approach. The test set bound is better behaved as the confidence interval is confined to the interval $[0, 1]$ and is never over-optimistic.

This "misleading" is both pessimistic and (much worse) optimistic. The pessimism can be seen by intervals with boundaries less than 0 or greater than 1 and the optimism by observing what happens when the test error rate is 0. When we observe perfect classification, our confidence interval should *not* have size 0 for any finite $m$.

The basic problem with this approach is that the Binomial distribution is not similar to a Gaussian when the error rate is near 0. Since our goal is finding a classifier with a small true error, it is essential that the means we use to evaluate classifiers work in this regime. The test set bound can satisfy this requirement (and, in fact, operates well for all true error regimes).

(1) The test set bound approach is *never* optimistic.
(2) The test set bound based confidence interval always returns an upper and lower bound in $[0, 1]$.

The $2\hat{\sigma}$ method is a relic of times when computational effort was expensive. It is now simple and easy to calculate a bound based upon the cumulative distribution of the Binomial (see [11] for a program which does this).

The test set bound can be thought of as a game where a "Learner" attempts to convince a reasonable "Verifier" of the amount of learning which has occurred. Pictorially we can represent this as in figure 3.5.

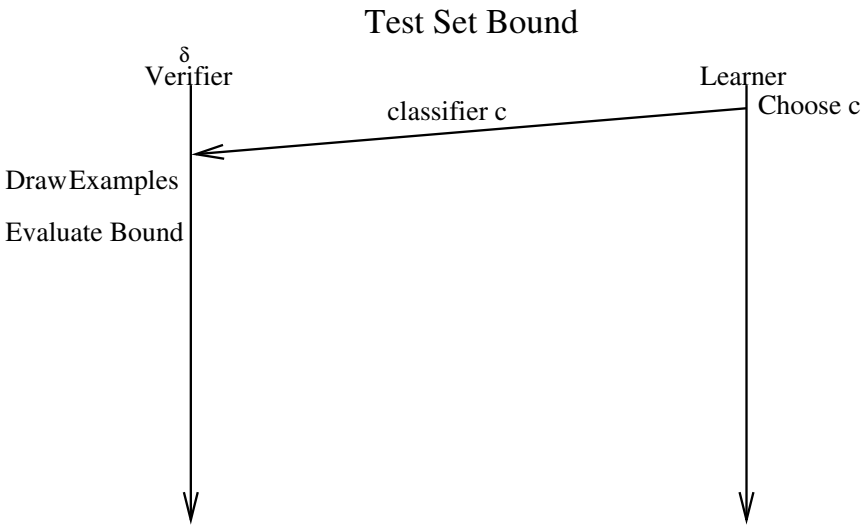


FIGURE 3.5. For this diagram "increasing time" is pointing downwards. The only requirement for applying this bound is that the learner must commit to a classifier without knowledge of the test examples. A similar diagram for train set bounds is presented later (and is somewhat more complicated). We can think of the bound as a technique by which the "Learner" can convince the "Verifier" that learning has occurred (and the degree to which it has occurred). Each of the proofs can be thought of as a communication protocol for an interactive proof of learning by the Learner.

## 4. The Occam's Razor Bound

Given that the bounds for the simple test set bound work well, why do we need to engage in further work? There is one serious drawback to the test set technique—it requires $m_{\text{test}}$ otherwise unused examples. This can strongly degrade the value of the learned hypothesis because an extra $m_{\text{test}}$ examples for the training set increases the true error of the learned hypothesis from 0 to 0.5 for some natural learning algorithm/learning problem pairs.

There is another reason why training set based bounds are important. Many learning algorithms implicitly assume that the train set accuracy "behaves like" the true error in choosing the hypothesis. With an inadequate number of training examples, there may be very little relationship between the behavior of the train set accuracy and the true error. Training set based bounds can be used *in* the training algorithm and can provide insight into the learning problem itself.

This section is organized into three subsections.

(1) Subsection 4.1 states and proves the Occam's Razor bound.
(2) Subsection 4.2 proves that the Occam's Razor bound cannot be improved in general.
(3) Subsection 4.3 discusses implications of the Occam's Razor bound and shows results for its application.

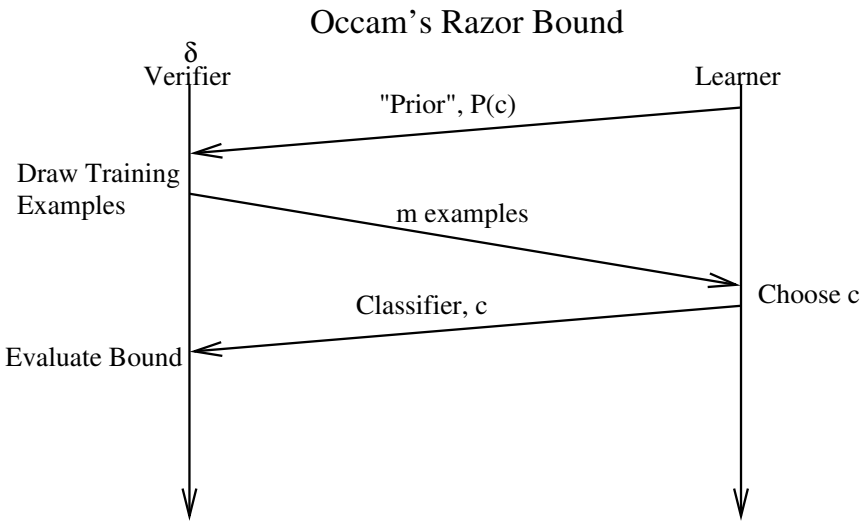


FIGURE 4.1. In order to apply the Occam's Razor bound it is necessary that the choice of "prior" be made before seeing any training examples. Then, the bound is calculated based upon the chosen classifier. Note that it *is* "legal" to chose the classifier based upon the prior $P(c)$ as well as the empirical error $\hat{c}_S$.

4.1. **The Occam's Razor bound.** This bound in more approximate forms has appeared elsewhere [1][18]. We use "prior" (with quotes) here because it is an arbitrary probability distribution over classifiers and not necessarily a Bayesian prior. The distinction is important, because the theory holds regardless of whether or not a Bayesian prior is used.

**Theorem 4.1.** *(Occam's Razor Bound) For all "priors" $P(c)$ over the classifiers, $c$, for all $\delta \in (0,1]$:*

$$\Pr_{S\sim D^m}\left(\forall c:\ c_D \leq \overline{Bin}\left(\hat{c}_S, \delta P(c)\right)\right) \geq 1-\delta$$

The application of the Occam's Razor bound is somewhat more complicated than the application of the test set bound. Pictorially, the protocol for bound application is given in figure 4.1. It is very important to notice that the "prior" $P(c)$ must be selected *before* seeing the training examples.

*Proof.* (pictorially in figure 4.2) The proof starts with the test set bound:

$$\forall c \quad \Pr_{S\sim D^m}\left(c_D \leq \overline{\mathrm{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right) \geq 1-\delta P(c)$$

Negating this statement, we get:

$$\forall c \quad \Pr_{S\sim D^m}\left(c_D > \overline{\mathrm{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right) < \delta P(c)$$

then, we apply the union bound in a nonuniform manner. The union bound says that $\Pr(A \text{ or } B) \leq \Pr(A) + \Pr(B)$. Applying the union bound to every classifier with a positive measure gives a total probability of failure of

$$\sum_c \delta P(c) = \delta \sum_c P(c) = \delta$$

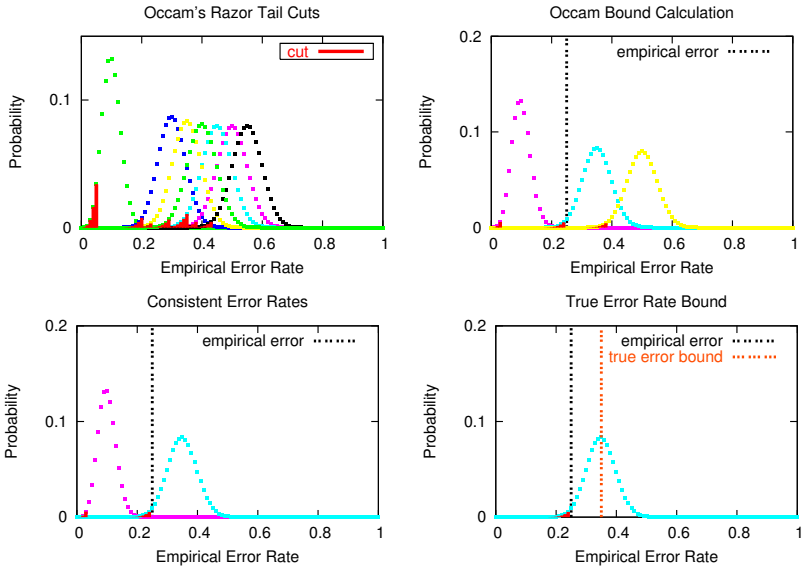


FIGURE 4.2. The sequence of pictures is the pictorial representation of the proof of the Occam's Razor Bound. The first figure shows a set of classifiers, each with a tail cut of some varying depth. The second picture shows an observed training error and the possible Binomial distributions for a chosen classifier. The third picture shows the true errors which are consistent with the observation and the tail cuts. The fourth picture shows the true error bound.

which implies

$$\Pr_{S\sim D^m}\left(\exists c:\ c_D > \overline{\text{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right) < \delta$$

Negating this again completes the proof. □

4.1.1. *Occam's Razor Corollaries.* Just as with the test set bound, we can relax the Occam's Razor bound (theorem 4.1) with the Chernoff approximations to get a somewhat more tractable expression.

**Corollary 4.2.** *(Chernoff Occam's Razor Bound) For all "priors" $P(c)$ over classifiers, for all $\delta \in (0, 1]$:*

$$\Pr_{S\sim D^m}\left(\forall c:\ c_D \leq \hat{c}_S + \sqrt{\frac{\ln\frac{1}{P(c)} + \ln\frac{1}{\delta}}{2m}}\right) \geq 1 - \delta$$

*Proof.* approximate the Binomial tail with the Chernoff Bound (lemma 3.5). □

Many people are more familiar with a degenerate form of this bound where $P(c) = \frac{1}{|H|}$ and $H$ is some set of classifiers. In that case, simply replace $\ln\frac{1}{P(c)}$ with $\ln|H|$. The form presented here is both more general and necessary if the bound is to be used in practice.

Other corollaries as in section 3.1.1 exist for the Occam's Razor bound. In general, just substitute $\delta \to \delta P(c)$.

4.1.2. *Occam's Razor Lower bound.* Just as for the test set bound, a lower bound of the same form applies.

**Theorem 4.3.** *(Occam's Razor Lower Bound) For all "priors" $P(c)$ over the classifiers, $c$, for all $\delta \in (0,1]$:*

$$\Pr_{S\sim D^m}\left(\forall c:\ c_D \geq \min_p\{p:\ 1 - Bin(\hat{c}_S, p) \geq \delta P(c)\}\right) \geq 1-\delta$$

**Example.** (continued) Suppose that instead of having 100 test examples, we had 100 train examples. Also suppose that before seeing the train examples, we committed to $P(c) = 0.1$ for $c$ the constant classifier which predicts "no rain". Then, the Chernoff approximations of the upper and lower bound give the interval, $c_D \in [0.22, 0.54]$. With an exact calculation, we get $c_D \in [0.26, 0.51]$.

4.1.3. *The state of the art.* A very large amount of work has been done on train set bounds. In addition to those included here, there is:

(1) Reinterpretations of uniform convergence [23] results for continuously parameterized classifiers.
(2) Reinterpretations of PAC convergence [22] results.
(3) Shell bounds [13] which take advantage of the distribution of true error rates on classifiers.
(4) Train and Test bounds [16] which combine train set and test set bounds.

Of this large amount of work only a small fraction has been shown to be useful on real-world learning algorithm/learning problem pairs. The looseness of train set based bounds often precludes analytical use.

4.2. **The Occam's Razor Bound is sometimes Tight.** The question of tightness for train set bounds is important to address, as many of them have been extremely loose. The simplest method to address this tightness is constructive: exhibit a learning problem/algorithm pair for which the bound is almost achieved. For the test set bound, this is trivial as any classifier with a large enough true error will achieve the bound. For the train set bound, this is not so trivial.

How tight is the Occam's Razor bound (4.1)? The answer is *sometimes* tight. In particular, we can exhibit a set of learning problems where the Occam's Razor bound can not be made significantly tighter as a function of the observables, $m$, $\delta$, $P(c)$, and $\hat{c}_S$. After fixing the value of these quantities we construct a learning problem exhibiting this near equivalence to the Occam's Razor bound.

**Theorem 4.4.** *(Occam's Razor tightness) For all $P(c)$, $\frac{k}{m}$, $\delta$ there exists a learning problem and algorithm such that:*

$$\Pr_{S\sim D^m}\left(\exists c:\ \hat{c}_S \leq \frac{k}{m} \text{ and } c_D \geq \overline{Bin}(\hat{c}_S, \delta P(c))\right) \geq \delta - \delta^2$$

*Furthermore, if $c^*$ is the classifier with minimal training error, then:*

$$\Pr_{S\sim D^m}\left(c^*_D \geq \overline{Bin}(\hat{c}^*_S, \delta P(c))\right) \geq \delta - \delta^2$$

Intuitively, this theorem implies that we can not improve significantly on the Occam's Razor bound (theorem 4.1) without using extra information about our learning problem.

*Proof.* The proof is constructive: we create a learning problem on which large deviations are likely. We start with a prior $P(c)$, probability of error $\delta$, $m$, and a targeted empirical error rate, $\frac{k}{m}$. For succinctness we assume that $P(c)$ has support on a finite set of size $n$.

To define the learning problem, let: $X = \{0,1\}^n$ and $Y = \{0,1\}$.

The distribution $D$ can be drawn by first selecting $Y$ with a single unbiased coin flip, and then choosing the $i$th component of the vector $X$ independently, $\Pr((X_1, ..., X_n)|Y) = \Pi_{i=1}^n \Pr(X_i|Y)$ . The individual components are chosen so $\Pr(X_i = Y|Y) = \overline{\text{Bin}}\left(\frac{k}{m}, \delta P(c)\right)$.

The classifiers we consider just use one feature to make their classification: $c_i(x) = x_i$. The true error of these classifiers is given by: $c_D = \overline{\text{Bin}}\left(\frac{k}{m}, \delta P(c)\right)$.

This particular choice of true errors implies that if any classifier has a too-small train error rate, then the classifier with minimal train error must have a too-small train error.

Using this learning problem, we know that:

$$\forall c, \forall \delta \in (0,1]: \Pr_{S \sim D^m}\left(c_D \geq \overline{\text{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right) = \delta P(c)$$

(negation)

$$\Rightarrow \forall c, \forall \delta \in (0,1]: \Pr_{S \sim D^m}\left(c_D < \overline{\text{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right) = 1 - \delta P(c)$$

(independence)

$$\Rightarrow \forall \delta \in (0,1]: \Pr_{S \sim D^m}\left(\forall c\ c_D < \overline{\text{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right) < \prod_c \left(1 - \delta P(c)\right)$$

(negation)

$$\Rightarrow \forall \delta \in (0,1]: \Pr_{S \sim D^m}\left(\exists c\ c_D \geq \overline{\text{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right)$$

$$= \sum_{i=1}^n \delta P(c_i)\left(1 - \Pr_{S \sim D^m}\left(\exists c \in \{c_1, ..., c_{i-1}\}\ c_D \geq \overline{\text{Bin}}\left(\hat{c}_S, \delta P(c)\right)\right)\right)$$

$$\geq \sum_{i=1}^n \delta P(c_i)(1 - \delta)$$

$$= \delta - \delta^2$$

□

The lower bound theorem implies that we can not improve an Occam's Razor like statement. However, it is important to note that large improvements are possible if we use other sources of information. To see this, just note the case where every single classifier happens to be the same. In this case the "right" bound would the be the *test* set bound, rather than the train set bound. The PAC-Bayes bound and the Sample Compression bound presented in the next sections use other sources of information.

### 4.3. **Train set bound implications.**

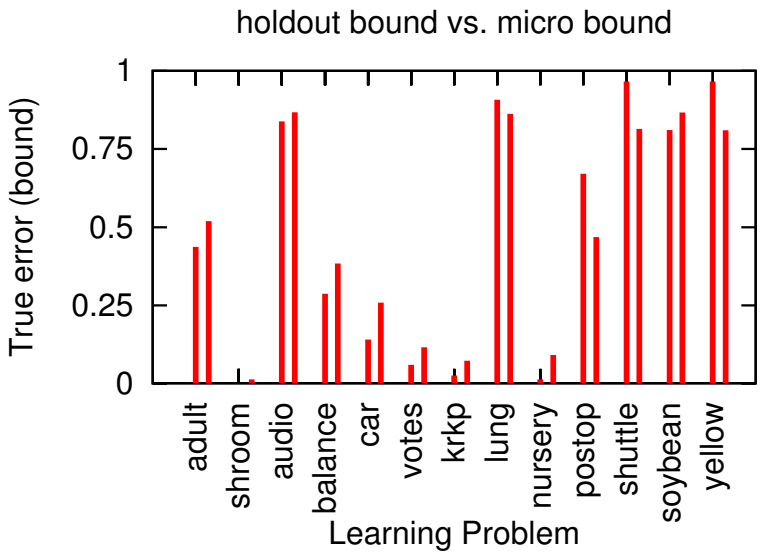


FIGURE 4.3. This is a plot comparing confidence intervals built based upon the test set bound (3.3) with an 80%/20% train/test split on the left and the Occam's Razor bound (4.1) with all data in the training set on the right. The Occam's razor bound is sometimes superior on the smaller data sets and always nonvacuous (in contrast to many other train set bounds).

4.3.1. *results.* The Occam's Razor bound is strongly related to compression. In particular, for any self-terminating description language, $d(c)$, we can associate a "prior" $P(c) = 2^{-|d(c)|}$ with the property that $\sum_c P(c) \leq 1$. Consequently, short description length classifiers tend to have a tighter convergence and the penalty term, $\ln \frac{1}{P(c)}$ is the number of "nats" (bits base e). For any language fixed before seeing the train set, classifiers with shorter description lengths have tighter bounds on the true error rate.

One particularly useful description language to consider is the execution trace of a learning algorithm. If we carefully note the sequence of data-dependent choices which a learning algorithm makes, then the output classifier can be specified by a sequence such as "2nd choice, third choice, first choice, etc..." This is the idea behind microchoice bounds [12]. Results for this approach are reported in Figure 4.3 and are strong enough to act as an empirical existence proof that Occam's Razor bounds can be made tight enough for useful application.

## 5. PAC-BAYES BOUND

The PAC-Bayes bound[18] is particularly exciting because it can provide quantitatively useful results for classifiers with *real valued* parameters. This includes

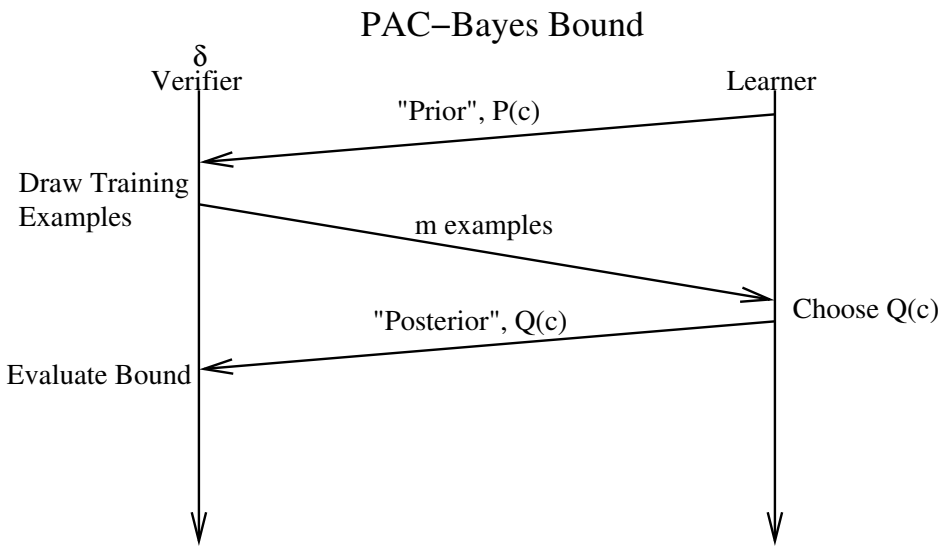


FIGURE 5.1. The "interactive proof of learning" associated with the PAC-Bayes bound. The figure is the same as for the Occam's razor bound, except that instead of committing to a single classifier, the PAC-Bayes bound applies to any distribution over classifiers.

such commonly used classifiers as Support Vector Machines and Neural Networks.[2] This section is divided into three parts:

(1) Subsection 5.1 States and proves the PAC-Bayes Bound.
(2) Subsection 5.2 shows that the PAC-Bayes Bound is nearly as tight as possible given the observations.
(3) Subsection 5.3 discusses results from the application of the PAC-Bayes bound to support vector machines.

5.1. **The PAC-Bayes Bound.** The PAC-Bayes bound has been improved by tightening [14] and then with a much simpler proof [20] since it was originally stated. The statement and proof presented here incorporate these improvements and improve on them slightly.

The PAC-Bayes bound is dependent upon two derived quantities, an average true error:

$$Q_D \equiv E_{c\sim Q} c_D$$

and an average train error:

$$\hat{Q}_S \equiv E_{c\sim Q} \hat{c}_S$$

These quantities can be interpreted as the train error and true error of the meta-classifier which chooses a classifier according to $Q$ every time a classification is made. If we refer to this meta-classifier as $Q$, the notation for error rates is consistent with our earlier notation.

[2]There is a caveat here—the bound only applies to stochastic versions of the classifiers. However, the probability that the stochastic classifier differs from the classifier can be made very small.

The "interactive proof of learning" viewpoint of the PAC-Bayes bound is shown in figure 5.1. It is essentially the same as for the Occam's Razor bound except for the commitment to the metaclassifier $Q$ rather than the classifier $c$.

**Theorem 5.1.** *(PAC-Bayes Bound) For all "priors" $P(c)$ over the classifiers, $c$, for all $\delta \in (0,1]$:*

$$\Pr_{S\sim D^m}\left(\forall Q(c):\ KL\left(\hat{Q}_S||Q_D\right) \leq \frac{KL(Q||P) + \ln\frac{m+1}{\delta}}{m}\right) \geq 1-\delta$$

*Here $KL(q||p) = q\ln\frac{q}{p} + (1-q)\ln\frac{1-q}{1-p}$ for $q < p$ and $KL(Q||P) = E_{c\sim Q}\ln\frac{Q(c)}{P(c)}$ is the Kullback-Leibler divergence between two distributions.*

Note that the PAC-Bayes bound applies to any *distribution* over classifiers. When $Q$ is concentrated on one classifier, we have $\mathrm{KL}(Q||P) = \ln\frac{1}{P(c)}$, just as in the Occam's razor bound[3], with the only distinction being the additive $\frac{\ln(m+1)}{m}$ term. It is somewhat surprising that the bound holds for *every* distribution $Q$ with only the slight worsening by $\frac{\ln(m+1)}{m}$.

Since the KL-divergence applies to distributions over continuous valued parameters, the PAC-Bayes bound can be nontrivially tight in this setting as well. This fact is used in the application section.

We first state a couple simple lemmas that are handy in the proof. The intuition behind this lemma is that the expected probability of an event is not too small.

**Lemma 5.2.** *For all $P(c)$, For all $\delta \in (0,1]$:*

$$\Pr_{S\sim D^m}\left(E_{c\sim P}\frac{1}{\Pr_{S\sim D^m}(\hat{c}_S)} \leq \frac{m+1}{\delta}\right) \geq 1-\delta$$

*Proof.* Note that:

$$\begin{aligned}\forall c\ \ E_{S\sim D^m}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})} &= \sum_{\frac{k}{m}} \Pr_{S\sim D^m}\left(\hat{c}_S = \frac{k}{m}\right)\frac{1}{\Pr_{S'\sim D^m}\left(\hat{c}_{S'} = \frac{k}{m}\right)} \\ &= m+1\end{aligned}$$

Taking the expectation over classifiers according to $P$ and switching the order of expectation, we get:

$$E_{S\sim D^m}E_{c\sim P}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})} = m+1$$

and using the Markov inequality ($X \geq 0$, $EX = \mu$, $\Rightarrow \Pr(X > \frac{\mu}{\delta}) \leq \delta$), we get:

$$\forall P\ \ \Pr_{S\sim D^m}\left(E_{c\sim P}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S = \hat{c}_{S'})} \geq \frac{m+1}{\delta}\right) \leq \delta$$

□

The next lemma shows that a certain expectation is bounded by the Kullback-Leibler distance between two coin flips, just as for the relative entropy Chernoff bound (Lemma 3.5).

**Lemma 5.3.** *For all $Q(c)$:* $\frac{E_{c\sim Q}\ln\frac{1}{\Pr_{S\sim D^m}(\hat{c}_S)}}{m} \geq KL(\hat{Q}_S||Q_D)$

[3]As weakened with the relative entropy Chernoff bound (Lemma 3.5) on the Binomial.

*Proof.* $\frac{1}{m} E_{c\sim Q} \ln \frac{1}{\binom{m}{m\hat{c}_S} c_D^{m\hat{c}_S} (1-c_D)^{m(1-\hat{c}_S)}}$

$$= E_{c\sim Q}\left[\hat{c}_S \ln \frac{1}{c_D} + (1-\hat{c}_S)\ln\frac{1}{1-c_D}\right] - \frac{E_{c\sim Q}\ln\binom{m}{m\hat{c}_S}}{m}$$

Jensen's inequality ($\log \frac{1}{x}$ is convex $\Rightarrow E\log\frac{1}{x} \geq \log\frac{1}{Ex}$):

$$E_{c\sim Q}\left[\hat{c}_S \ln \frac{1}{c_D} + (1-\hat{c}_S)\ln\frac{1}{1-c_D}\right] \geq \hat{Q}_S \ln\frac{1}{Q_D} + \left(1-\hat{Q}_S\right)\ln\frac{1}{1-Q_D}$$

$$\text{and} \quad \frac{E_{c\sim Q}\ln\binom{m}{m\hat{c}_S}}{m} \leq \frac{E_{c\sim Q}\ln e^{mH(\hat{c}_S)}}{m} = \frac{E_{c\sim Q} mH(\hat{c}_S)}{m} \leq H\left(\hat{Q}_S\right)$$

□

With these two lemmas, the PAC-Bayes theorem is easy to prove.

*Proof.* (Of the PAC-Bayes theorem) Let

$$P_G(c,\hat{c}_S) = \frac{1}{\Pr_{S\sim D^m}(\hat{c}_S)\, E_{c\sim P}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S=\hat{c}_{S'})}} P(c)$$

$$\Rightarrow 0 \leq \mathrm{KL}(Q||P_G) = E_{c\sim Q}\ln\frac{Q(c)}{P(c)} \Pr_{S\sim D^m}(\hat{c}_S)\, E_{c\sim P}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S=\hat{c}_{S'})}$$

$$= \mathrm{KL}(Q||P) - E_{c\sim Q}\ln\frac{1}{\Pr_{S\sim D^m}(\hat{c}_S)} + \ln E_{c\sim P}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S=\hat{c}_{S'})}$$

$$\Rightarrow E_{c\sim Q}\ln\frac{1}{\Pr_{S\sim D^m}(\hat{c}_S)} \leq \mathrm{KL}(Q||P) + \ln E_{c\sim P}\frac{1}{\Pr_{S'\sim D^m}(\hat{c}_S=\hat{c}_{S'})}$$

Now, use lemma 5.2 on the right hand term and lemma 5.3 on the left hand term to finish the proof. □

Note that one of the last inequalities in the proof is not necessary, and so a slightly tighter bound can be proved.

5.2. **The PAC-Bayes Bound is sometimes Tight.** Since the PAC-Bayes bound is (almost) a generalization of the Occam's Razor bound, the tightness result for Occam's Razor also applies to PAC-Bayes bounds.

5.3. **Application of the PAC-Bayes Bound.** Applying the PAC-Bayes bound requires some significant specialization [15]. Here, we specialize to classifiers of the form:

$$c(x) = \mathrm{sign}(\vec{w}\cdot\vec{x})$$

Note that via the kernel trick, Support Vector Machines also have this form.

The specialization is naturally expressed in terms of a few derived quantities:

(1) The cumulative distribution of a Gaussian. Let $\bar{F}(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$. Here we use $\bar{F}$ rather than $F$ to denote the fact that we integrate from $\infty$ to $x$ rather than $-\infty$ to $x$.

(2) A "posterior" distribution $Q(\vec{w},\mu)$ which is $N(\mu,1)$ for some $\mu > 0$ in the direction of $\vec{w}$ and $N(0,1)$ in all perpendicular directions.

(3) The normalized margin of the examples

$$\gamma(\vec{x}, y) = \frac{y\vec{w} \cdot \vec{x}}{||\vec{w}||||\vec{x}||}$$

(4) A stochastic error rate, $\hat{Q}(\vec{w}, \mu)_S = E_{\vec{x},y \sim S} \bar{F}(\mu\gamma(\vec{x}, y))$

This last quantity in particular is very important to understand. Consider the case as $\mu$ approaches $\infty$. When the margin is negative (indicating an incorrect classification), $\bar{F}(\mu\gamma(\vec{x}, y))$ approaches 1. When the margin is positive $\bar{F}(\mu\gamma(\vec{x}, y))$ approaches 0. Thus, $\hat{Q}(\vec{w}, \mu)_S$ is a softened form of the empirical error $\hat{c}_S$ which takes into account the margin.

**Corollary 5.4.** *(PAC-Bayes Margin Bound) For all distributions $D$, for all $\delta \in (0, 1]$, we have:*

$$\Pr_{S \sim D^m} \left( \forall \vec{w}, \mu : \ KL\left(\hat{Q}(\vec{w}, \mu)_S || Q(\vec{w}, \mu)_D\right) \leq \frac{\frac{\mu^2}{2} + \ln \frac{m+1}{\delta}}{m} \right) \geq 1 - \delta$$

*Proof.* The proof is very simple. We just chose the prior $P = N(0, 1)^n$ and work out the implications.

Since the Gaussian distribution is the same in every direction, we can reorient the coordinate system of the prior to have one dimension parallel to $w$. Since the draws in the parallel and perpendicular directions are independent, we have:

$$\begin{aligned} \mathrm{KL}(Q||P) &= \mathrm{KL}(Q_\perp||P_\perp) + \mathrm{KL}(N(\mu, 1)||N(0, 1)) \\ &= \frac{\mu^2}{2} \end{aligned}$$

as required.

All that remains is calculating the stochastic error rate $\hat{Q}(\vec{w}, \mu)_S$. Fix a particular example, $(\vec{x}, y)$. This example has a natural decomposition $\vec{x} = \vec{x}_{||} + \vec{x}_\perp$ into a component, $\vec{x}_{||}$ parallel to the weight vector $\vec{w}$ and a component $\vec{x}_\perp$ perpendicular to the weight vector.

To classify, we draw weight vector $\vec{w}'$ from $\hat{Q}(\vec{w}, \mu)$. This $\vec{w}'$ consists of 3 components, $\vec{w}' = \vec{w}'_{||} + \vec{w}'_\perp + \vec{w}'_{\perp\perp}$. Here $\vec{w}'_{||} \sim N(\mu, 1)$ is parallel to the original weight vector, $\vec{w}'_\perp \sim N(0, 1)$ which is parallel to $\vec{x}_\perp$ and $\vec{w}'_{\perp\perp}$ is perpendicular to both $\vec{w}$ and $\vec{x}$. We have:

$$\begin{aligned} \hat{Q}(\vec{w}, \mu)_S &= E_{\vec{x},y \sim S, \vec{w}' \sim Q(\vec{w},\mu)} I\left(y \neq \mathrm{sign}\left(\vec{w}' \cdot \vec{x}\right)\right) \\ &= E_{\vec{x},y \sim S, \vec{w}' \sim Q(\vec{w},\mu)} I\left(y\vec{w} \cdot \vec{x} \leq 0\right) \end{aligned}$$

If we let $w'_{||} = ||w'_{||}||$, $w'_\perp = ||w'_\perp||$, $x_{||} = ||\vec{x}_{||}||$, and $x_\perp = ||\vec{x}_\perp||$, and assume (without loss of generality) that $y = 1$ we get:

$$\begin{aligned} &= E_{\vec{x},y \sim S, w'_{||} \sim N(\mu,1), w'_\perp \sim N(0,1)} I\left(y(w'_{||}x_{||} + w'_\perp x_\perp) \leq 0\right) \\ &= E_{\vec{x},y \sim S} E_{w'_{||} \sim N(\mu,1)} E_{w'_\perp \sim N(0,1)} I\left(y(w'_{||}x_{||} + w'_\perp x_\perp) \leq 0\right) \\ &= E_{\vec{x},y \sim S} E_{z' \sim N(0,1)} E_{w'_\perp \sim N(0,1)} I\left(y\mu \leq -yz' - yw'_\perp \frac{x_\perp}{x_{||}}\right) \end{aligned}$$

Using the symmetricity of the Gaussian, this is:

$$= E_{\vec{x},y\sim S} E_{z'\sim N(0,1)} E_{w'_\perp \sim N(0,1)} I\left(y\mu \leq yz' + yw'_\perp \frac{x_\perp}{x_{||}}\right)$$

Using the fact that the sum of two Gaussians is a Gaussian:

$$= E_{\vec{x},y\sim S} E_{v\sim N\left(0,1+\frac{x_\perp^2}{x_{||}^2}\right)} I\left(y\mu \leq yv\right)$$

$$= E_{\vec{x},y\sim S} E_{v\sim N\left(0,\frac{1}{\gamma(\vec{x},y)^2}\right)} I\left(y\mu \leq yv\right)$$

$$= E_{\vec{x},y\sim S} \bar{F}\left(\mu\gamma(\vec{x},y)\right)$$

finishing the proof. □

Using the corollary, the true error bound $\bar{Q}(\vec{w},\mu)_D$ satisfies the equation:

$$\mathrm{KL}\left(\hat{Q}(\vec{w},\mu)_S || \bar{Q}(\vec{w},\mu)_D\right) = \frac{\frac{\mu^2}{2} + \ln\frac{m+1}{\delta}}{m}$$

This is an implicit equation for $\bar{Q}$ which can be easily solved numerically.

The bound is stated in terms of dot products here, so naturally it is possible to kernelize the result using methods from [6]. In kernelized form, the bound applies to classifiers (as output by SVM learning algorithms) of the form:

$$c(x) = \mathrm{sign}\left(\sum_{i=1}^m \alpha_i k(x_i,x)\right) \tag{5.1}$$

Since, by assumption, $k$ is a kernel, we know that $k(x_i,x) = \vec{\Phi}(x_i)\cdot\vec{\Phi}(x)$ where $\vec{\Phi}(x)$ is some projection into another space. In kernelized form, we get $\vec{w}\cdot\vec{x} = \sum_{i=1}^m \alpha_i k(x_i,x)$, $\vec{x}\cdot\vec{x} = k(x,x)$, $\vec{w}\cdot\vec{w} = \sum_{i,j}\alpha_i\alpha_j k(x_i,x_j)$, defining all of the necessary quantities to calculate the normalized margin,

$$\gamma(x,y) = \frac{\sum_{i=1}^m \alpha_i k(x_i,x)}{\sqrt{k(x,x)\sum_{i,j=1,1}^{m,m}\alpha_i\alpha_j k(x_i,x_j)}}$$

One element remains, which is the value of $\mu$. Unfortunately the bound can be nonmonotonic in the value of $\mu$, but it turns out that for classifiers learned by support vector machines on reasonable datasets, there is only one value of $\mu$ which is (locally, and thus globally) minimal. A binary search over some reasonable range of $\mu$ (say from 1 to 100) can find the minima quickly, given the precomputation of the margins. It is worth noting again here that we are not "cheating"—the bound holds for all values of $\mu$ simultaneously.

The computational time of the bound calculation is dominated by the calculation of the margins which is $O\left(m^2\right)$ where $m$ is the number of support vectors with a nonzero associated $\alpha$. This computational time is typically dominated by the time of the SVM learning algorithm.

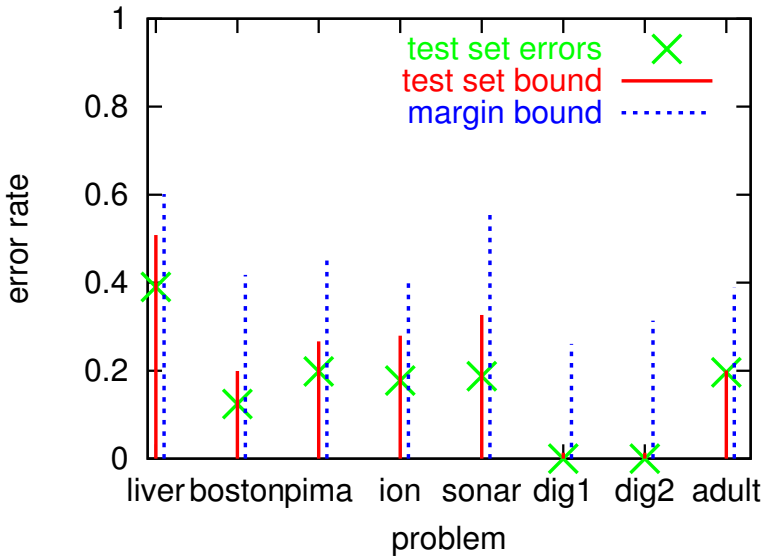


FIGURE 5.2.
This figure shows the results of applying SVMlight to 8 datasets with a Gaussian kernel and a 70/30 train/test split. The observed test error rate is graphed as an X. On the test set, we calculate a Binomial confidence interval (probability of bound failure = 0.01) which upper bounds the true error rate. On the training set we calculate the PAC-Bayes margin bound for an optimized choice of $\mu$.

5.3.1. *Results.* Application of this bound to support vector machines is of significant importance because SVMs are reasonably effective and adaptable classifiers in common and widespread use. A SVM learns a kernelized classifier as per equation 5.1[4].

We apply the support vector machine to 8 UCI database problems chosen to fit the criteria "two classes" and "real valued input features". The problems vary in size over an order of magnitude from 145 to 1428 examples. In figure 5.2 we use a 70/30 train/test split of the data.

In all experiments, we use SVMlight[8] with a Gaussian kernel and the default bandwidth. Results for other reasonable choices of the "C", bandwidth[5], and kernel appear to be qualitatively similar (although of course they differ quantitatively).

It is important to note that the PAC-Bayes margin bound is *not* precisely a bound (or confidence interval) on the true error rate of the learned classifier. Instead, it is a true error rate bound on an associated stochastic classifier chosen so as to have a similar test error rate. These bounds can be regarded as bounds for the original classifier only under an additional assumption: that picking a classifier according to the majority vote of this stochastic distribution does not worsen the true error rate. This is not true in general, but may be true in practice.

[4]Some SVM learning algorithms actually learn a classifier of the form: $c(x) = \text{sign}\left(b + \sum_{i=1}^{m} \alpha_i k(x_i, x)\right)$. We don't handle this form here.

[5]Note that the bandwitdh of a gaussian kernel used by an SVM is not directly related to the optimized value of $\mu$ we find.

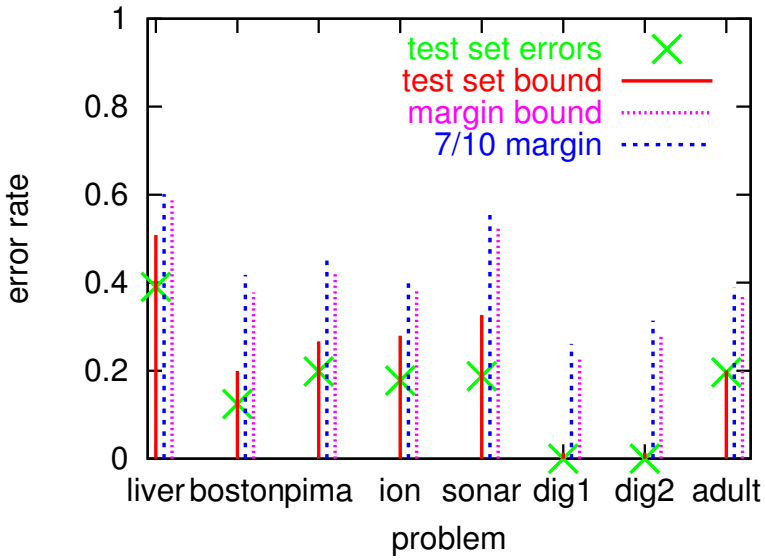


FIGURE 5.3.
In addition to comparing with everything in figure 5.2, we graph the margin bound when *all* of the data is used for the train set. Note that it improves somewhat on the margin bound calculated using the 70% train set (7/10 margin bound), but not enough to compete with the test set bound.

It is of course unfair to compare the train set bound with the test set bound on a 70/30 train/test split because a very tight train set bound would imply that it is unnecessary to even have a test set. In figure 5.3 we compare the true error bounds on all of the data to the true error bounds generated from the 70/30 train/test split.

The results show that the PAC-Bayes margin bound is tight enough to give useful information, but still not competitive with the test set bounds. This is in strong contrast with a tradition of quantitatively impractical margin bounds. There are several uses available for bounds which provide some information but which are not fully tight.

(1) They might be combined with a train/test bound [16].
(2) The train set bound might easily become tighter for smaller sample sizes. This was observed in [16].
(3) The train set bound might still have the right "shape" for choosing an optimal parameter setting, such as "C" in a Support Vector Machine.

## 6. Sample Compression Bound

The sample compression bound [17], [5] is like the PAC-Bayes bound in that it applies to arbitrary precision continuous valued classifiers. Unlike the PAC-Bayes bound, it applies meaningfully to nonstochastic classifiers. Mainstream learning algorithms do not optimize the sample compression metric, so the bound application is somewhat rarer. Nonetheless, there do exist some reasonably competitive learning algorithms for which the sample compression bound produces significant results.

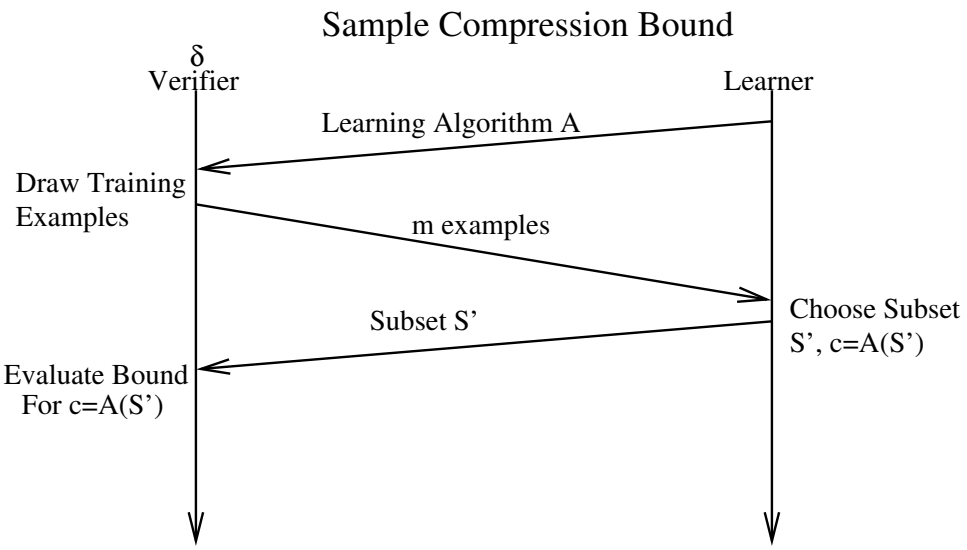


FIGURE 6.1. The interactive proof of learning for the sample compression bound. Note that the learning algorithm is arbitrary here, similar to the test set bound.

The section is organized as follows:

(1) Subsection 6.1 states and proves the sample compression bound.
(2) Subsection 6.2 shows that the sample compression bound is nearly as tight as possible given the observations.
(3) Subsection 6.3 discusses results from the application of the sample compression bound to support vector machines.

6.1. **The Sample Compression Bound.** The Sample Compression bound stated here is differs somewhat from other results ([17][5] and others) by generalization and simplification but the bound behavior is qualitatively identical.

Suppose we have a learning algorithm, $A(S)$ whose training is "sparse"[6] in the sense that the output classifier is dependent upon only a subset of the data, $A(S) = A(S')$ for $S' \subseteq S$. The sample compression bound is dependent on the error rate, $\hat{c}_{S-S'}$ on the subset $S - S'$. The motivation here is that the examples which the learning algorithm does *not* depend upon are "almost" independent and so we can "almost" get a test set bound. In general, the bound becomes tighter as the dependent subset $S'$ becomes smaller and as the error rate on the nondependent subset $S - S'$ becomes smaller.

Viewed as an interactive proof of learning (in figure 6.1), the sample compression bound is unique amongst training set bounds because it does not require *any* initial commitment to a measure over the classifiers[7].

[6]This is satisfied, for example, by the Support Vector Machine algorithm which only depends upon the set of support vectors.

[7]However, we can regard the commitment to a learning algorithm as an implicit commitment to a measure over classifiers which is dependent on the learning algorithm and the distribution generating the data.

**Theorem 6.1.** *(Sample Compression Bound) For all $\delta \in (0,1]$, $D$, $A$:*

$$\Pr_{S\sim D^m}\left(\forall S' \subseteq S \text{ with } c = A(S') : \ c_D \leq \overline{Bin}\left(\hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)\right) \geq 1-\delta$$

*Proof.* Suppose we knew in advance that the learning algorithm will not depend upon some subset of the examples. Then, the "undependent" subset acts like a test set and gives us a test set bound:

$$\forall S' \subseteq S \text{ with } c = A(S') : \Pr_{S\sim D^m}\left(c_D \leq \overline{\text{Bin}}\left(\hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)\right) \geq 1-\frac{\delta}{m\binom{m}{|S-S'|}}$$

(Note that, technically, it is possible to refer to $S'$ unambiguously before randomizing over $S$ by specifying the indices of $S$ contained in $S'$.) Negating this, we get:

$$\forall S' \subseteq S \text{ with } c = A(S') : \Pr_{S\sim D^m}\left(c_D > \overline{\text{Bin}}\left(\hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)\right) < \frac{\delta}{m\binom{m}{|S-S'|}}$$

and using the union bound ($\Pr(A \text{ or } B) \leq \Pr(A)+\Pr(B)$ over each possible subset, $S'$, we get:

$$\Pr_{S\sim D^m}\left(\exists S' \subseteq S \text{ with } c = A(S') : \ c_D > \overline{\text{Bin}}\left(\hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)\right) < \delta$$

Negating this again gives us the proof. □

6.2. **The Sample Compression Bound is Sometimes Tight.** We can construct a learning algorithm/learning problem pair such that the Sample compression bound is provably near optimal, as a function of its observables.

**Theorem 6.2.** *(Sample Compression Tightness) For all $\delta \in (0,1]$, $\frac{k}{m}$, there exists a distribution $D$ and learning algorithm $A$ s.t.*

$$\Pr_{S\sim D^m}\left(\exists S' \subseteq S \text{ with } c = A(S') : \ c_D > \overline{Bin}\left(\hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)\right) > \delta - \delta^2$$

*furthermore, if $S^*$ minimizes $\overline{Bin}\left(\hat{c}_{S-S'}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)$, then*

$$\Pr_{S\sim D^m}\left(c^* = A(S^*) : \ c^*_D > \overline{Bin}\left(\hat{c}^*_{S-S^*}, \frac{\delta}{m\binom{m}{|S-S^*|}}\right)\right) > \delta - \delta^2$$

*Proof.* The proof is constructive and similar to the Occam's Razor tightness result. In particular, we show how to construct a learning algorithm which outputs classifiers that err independently depending on the subset $S'$ used.

Consider an input space $X = \{0,1\}^{2^m}$. Each variable in the input space $x_{S'}$ can be thought of as indexing a unique subset $S' \subseteq S$ of the examples. In the rest of the proof, we index variables by the subset they correspond to.

Draws from the distribution $D$ can be made by first flipping an unbiased coin to get $y = 1$ with probability 0.5 and $y = -1$ with probability 0.5. The distribution

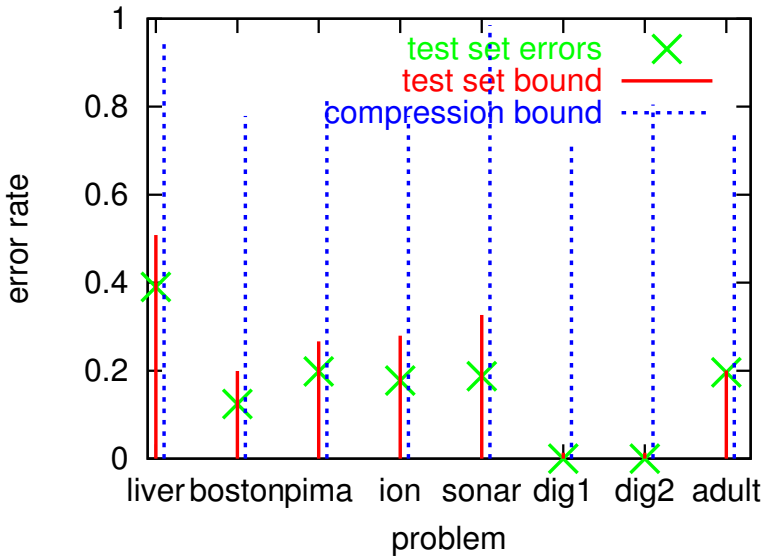


FIGURE 6.2. The sample compression bound applied to the output of a support vector machine with a Gaussian kernel. Here we use $\delta = 0.01$

on $X$ consists of a set of independent values after conditioning on $y$. Choose

$$\Pr(x_{S'} \neq y) = \overline{\text{Bin}}\left(\frac{k}{m}, \frac{\delta}{m\binom{m}{|S-S'|}}\right)$$

Now, the learning algorithm $A(S')$ is very simple—it just outputs the classifier $c(x) = x_{S'}$. On the set $S - S'$, we have:

$$\forall S' \quad \Pr_{S \sim D^m}\left(\hat{c}_{S-S'} \geq \frac{k}{m}\right) = 1 - \frac{\delta}{m\binom{m}{|S-S'|}}$$

Using independence, we get:

$$\Pr_{S \sim D^m}\left(\forall S' \;\; \hat{c}_{S-S'} \geq \frac{k}{m}\right) = \prod_{S'}\left(1 - \frac{\delta}{m\binom{m}{|S-S'|}}\right)$$

Negating, we get:

$$\Pr_{S \sim D^m}\left(\forall S' \;\; \hat{c}_{S-S'} < \frac{k}{m}\right) = 1 - \prod_{S'}\left(1 - \frac{\delta}{m\binom{m}{|S-S'|}}\right)$$

and doing some algebra, we get the result. □

6.3. **Application of the Sample Compression Bound.** One obvious application of the Sample Compression bound is to support vector machines, since the learned classifier is only dependent on the set of support vectors. If $S'$ is the set of support vectors then $S - S'$ is the set of nonsupport vectors. Unfortunately, it turns out that this does not work so well, as observed in figure 6.2.

There are other less common learning algorithms for which the sample compression bound works well. The Set Covering machine [19] has an associated bound which is a variant of the Sample Compression Bound.

## 7. Discussion

7.1. **Learning algorithm design.** *Every* train set bound implies a learning algorithm: choose the classifier which minimizes the true error bound. This sounds like a rich source of learning algorithms, but there are some severe caveats to that statement.

(1) It is important to note that the form of a train set bound does *not* imply that this minimization is a good idea. Choosing between two classifiers based upon their true error bound implies a better worst-case bound on the true error. It does not imply an improved true error. In many situations, there is some other metric of comparison (such as train error rate) which in fact creates better behavior.
(2) Another strong caveat is that, historically, train set bounds have simply not been tight enough on real datasets for a nonvacuous application. This is changing with new results, but more progress is necessary.
(3) Often the optimization problem is simply not very tractable. In addition to sample complexity, learning algorithms must be concerned with run time and space usage.

7.2. **Philosophy.** Train set bounds teach us about ways in which verifiable learning is possible, a subject which borders on philosophy. The train set bound presented here essentially shows that a reasonable person will be convinced of learning success when a short-description classifier does well on train set data. The results here do *not* imply that this is the only way to convincingly learn. In fact, the (sometimes large) looseness of the Occam's Razor bound suggests that other methods for convincing learning processes exist. This observation is partially shown by other train set bound results which are presented next.

7.3. **Conclusion.** This introduction to prediction theory covered two styles of bound: the test set bound and the train set bound. There are two important lessons here:

(1) Test set bounds provide a better way to report error rates and confidence intervals on future error rates than some current methods.
(2) Train set bounds can provide useful information.

It is important to note that the train set bound and test set bound techniques are not mutually exclusive. It is possible to use both simultaneously [16], and doing so is often desirable. Test set bounds are improved by the "free" information about the training error and train set bounds become applicable, even when not always tight.

*Acknowledgement* 7.1. I am indebted to many people for helpful suggestions assembling the content. This includes some anonymous reviewers, several coauthors on previous papers, Imre Kondor, Arindam Banerjee, and Alina Beygelzimer.

## Appendix

For those interested in comparing models, the uniform convergence model [23] requires the additional assumption of the axiom of choice (to achieve empirical risk minimization) and a bound on the hypothesis space complexity. Typical theorems are of the form "after $m$ examples, all training errors are near to true errors".

The PAC learning model[22] requires a polynomial time complexity learning algorithm and the assumption that the learning problem comes from some class. Theorems are of the form "after $m$ examples learning will be achieved".

Both of these models can support stronger statements than the basic prediction theory model presented here. Results from both of these models can apply here after appropriate massaging of results.

The online learning model [10] makes *no* assumptions. Typical theorems have the form "This learning algorithm's performance will be nearly as good as anyone of a set of classifiers." The online learning model has very general results and no ability to answer questions about future performance as we address here.

The prediction theory model can most simply be understood as a slight refinement of the information theory model.

## References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, M. Warmuth. "Occam's Razor." Information Processing Letters 24: 377-380, 1987.

[2] Avrim Blum, Adam Kalai, and John Langford 1999. Beating the Hold-out: Bounds for K-Fold and Progressive Cross-Validation. COLT99. http://www.cs.cmu.edu/~jcl/papers/progressive_validation/coltfinal.ps

[3] H. Chernoff, "A Measure of the Asymptotic Efficiency of Tests of a Hypothesis Based Upon the Sum of the Observations", Annals of Mathematical Statistics 23:493-507, 1952.

[4] Luc Devroye, Laszlo Gyorfi, Gabor Lugosi, "A Probabilistic Theory of Pattern Recognition" Springer-Verlag New York, 1996.

[5] Sally Floyd and Manfred Warmuth, "Sample Compression, Learnability, and the Vapnik-Chervonenkis Dimension", Machine Learning, Vol.21 (3), pp. 269–304, December 1995 http://www.cse.ucsc.edu/~manfred/pubs/sallycompr.forgalleys.ps

[6] R. Herbrich and T. Graepel. "Large scale Bayes point machines" In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, Advances in Neural Information System Processing 13, pages 528-534, Cambridge, MA, 2001.

[7] W. Hoeffding, "Probability inequalities for sums of bounded random variables", Journal of the American Statistical Association, 1963, 58:13-30.

[8] Thorsten Joachims, program SVMlight, available at http://svmlight.joachims.org/

[9] Michael Kearns and Dana Ron, "Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation." Neural Computation 11(6), pages 1427-1453, 1999. Also in Proceedings of the Tenth Annual Conference on Computational Learning Theory, ACM Press, 1997, pages 152–162. http://www.cis.upenn.edu/~mkearns/papers/multi.ps

[10] J. Kivinen and M. Warmuth, "Additive Versus Exponentiated Gradient Updates for Linear Prediction," in Journal of Information and Computation, vol. 132, no. 1, pp. 1-64, January 1997. http://www.cse.ucsc.edu/~manfred/pubs/lin.ps

[11] John Langford, Program "bound", http://www-2.cs.cmu.edu/~jcl/programs/bound/bound.html

[12] John Langford and Avrim Blum 1999. Microchoice Bounds and Self Bounding learning algorithms. Machine Learning Journal. http://www.cs.cmu.edu/~jcl/papers/microchoice/journal/journal_final.ps

[13] John Langford and David McAllester, "Computable Shell Decomposition Bounds" COLT 2000. http://www-2.cs.cmu.edu/~jcl/papers/computable_shell/colt_final.ps

[14] John Langford and Matthias Seeger, "Bounds for Averaging Classifiers" Technical report, Carnegie Mellon 2001 http://www-2.cs.cmu.edu/~jcl/papers/averaging/averaging_tech.ps

[15] John Langford and John Shawe-Taylor, "PAC-Bayes & Margins", NIPS 2002. http://www-2.cs.cmu.edu/~jcl/papers/PB-margin/PB-margin.ps
[16] John Langford, "Combining Train Set and Test Set Bounds", ICML2002. http://www-2.cs.cmu.edu/~jcl/papers/tnt_icml/train_n_test_final.ps
[17] Nick Littlestone and Manfred Warmuth, "Relating Data Compression and Learnability" Unpublished Manuscript http://www.cse.ucsc.edu/~manfred/pubs/lrnk-olivier.ps
[18] David McAllester, "PAC-Bayesian Model Averaging" COLT 1999. http://www.autoreason.com/posterior01.ps
[19] Mario Marchand and John Shawe-Taylor, "The Set Covering Machine", ICML 2001. http://www.ai.mit.edu/projects/jmlr/papers/volume3/marchand02a/marchand02a.pdf
[20] Matthias Seeger, "PAC-Bayesian Generalization Error Bounds for Gaussian Process Classification", Journal of Machine Learning Research 3 (2002), 233–269. http://www.dai.ed.ac.uk/homes/seeger/papers/seeger02a.ps.gz
[21] Sebastian Seung, Unpublished notes
[22] L.G. Valiant. "A Theory of the Leuarnable." Communications of the ACM 27(11):1134-1142, November 1984
[23] V. N. Vapnik and A. Y. Chervonenkis. "On the uniform convergence of relative frequencies of events to their probabilities." Theory of Probab. and its Applications, 16(2):264-280, 1971.