Quantitatively Tight Sample Complexity Bounds

John Langford

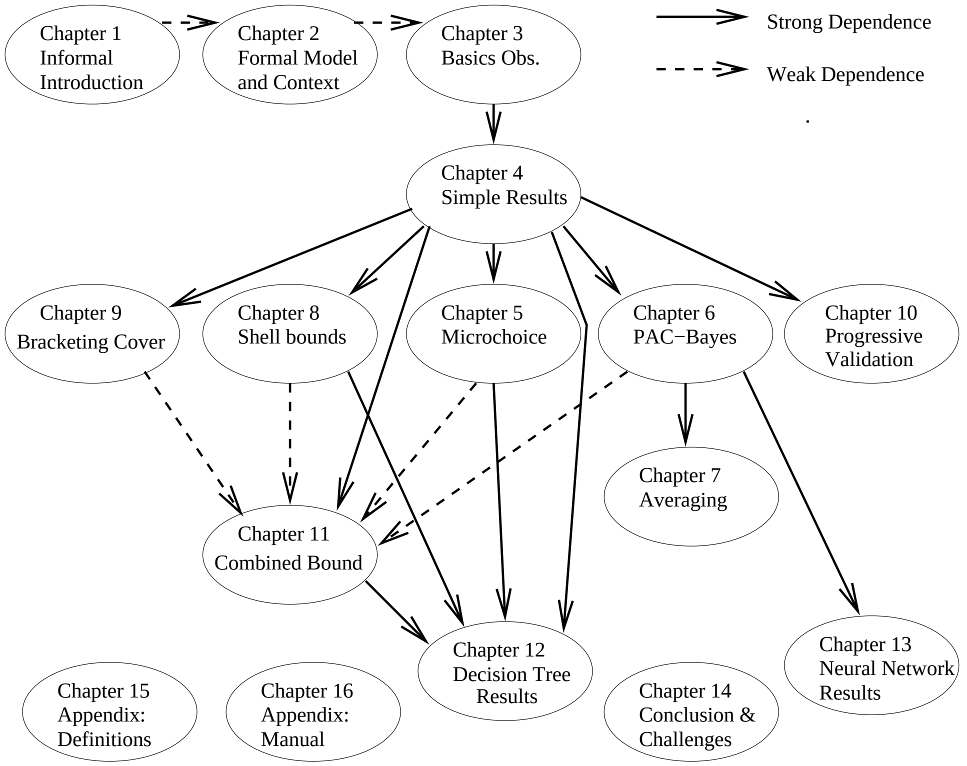
I present many new results on sample complexity bounds (bounds on the future error rate of arbitrary learning algorithms). Of theoretical interest are qualitative and quantitative improvements in sample complexity bounds as well as some techniques and criteria for judging the tightness of sample complexity bounds.

On the practical side, I show quantitative results (with true error rate bounds sometimes less than 0.01) for decision trees and neural networks with these sample complexity bounds applied to real world problems. I also present a technique for using both sample complexity bounds and (more traditional) holdout techniques.

Together, the theoretical and practical results of this thesis provide a well-founded practical method for evaluating learning algorithm performance based upon both training and testing set performance.

Code for calculating these bounds is provided.

Contents


Chapter 1
Informal
Introduction
Chapter 2
Formal Model
and Context
Chapter 3
Basics Obs.
Strong Dependence
Weak Dependence
Chapter 4
Simple Results
Chapter 9
Bracketing Cover
Chapter 8
Shell bounds
Chapter 5
Microchoice
Chapter 6
PAC−Bayes
Chapter 10
Progressive
Validation
Chapter 7
Averaging
Chapter 11
Combined Bound
Chapter 12
Decision Tree
Results
Chapter 13
Neural Network
Results
Chapter 15
Appendix:
Definitions
Chapter 16
Appendix:
Manual
Chapter 14
Conclusion &
Challenges

Part 1

Introductory Learning Theory

The introduction is broken into 4 parts. The first part is informal introduction to the learning model used here. The goal of the first part is to make explicit the design choices made in selecting our model. This is particularly important in machine learning because no model seems perfect.

- The second chapter formally introduces the model, the goals of the analysis, and provides context to related work.
- The third chapter states the fundamental statistical results upon which all of our techniques rest.
- The fourth chapter discusses basic learning theory results.

CHAPTER 1

Informal Introduction

What is a sample complexity bound? Informally, it is a bound on the number of examples required to learn a function. Therefore, in order to motivate the use of a sample complexity bound, we must first motivate the learning problem.

1.1. The learning problem

What is learning? Learning is the process of discovering relationships between events. Knowledge of the relationships between events is of great importance in coping with the environment. Naturally, humans tend to be quite good at the process of learning—so good that we sometimes do not even realize when it is hard.

The learning problem which we focus on is learning for computers. In particular, "How can a computer learn?" A few examples are illustrative:

(1) How can we make a computer with a microphone output a text version of what is being spoken?
(2) How can we make a computer with a camera recognize John?
(3) How can we make a computer controlling a robot arrive at some location?

We will work on learning a function from some input space to some output space - the supervised learning model.

1.2. The problem with the learning problem

The learning problem, as stated, is somewhat ill-posed. There are some very obvious ways for a computer to learn—for example by memorization. The difficulty arises when memorization is too expensive. Expense here typically has to do with acquiring enough experience so that future prediction problems have already been encountered. The real learning problem becomes, "Given incomplete information, how can a computer learn?" This formulation of the learning problem gives rise to a new problem - quantifying the amount of information required to learn. Sample complexity bounds address this second question: "When can a computer learn?"

In some cases learning is essentially hopeless. There are two notions of "hopeless": information theoretic and computational. Information theoretic difficulties arise when it is simply not possible to predict the output given the input. For example, predicting when a radioactive nucleus will decay is *always* difficult no matter what observations are made according to current physics[**9**]. Even when simple relations between inputs and outputs exist, the computation required to discover the simple relation can be formidable. A fine example of this is provided by cryptography [**19**] where a common task is to work out functions for which it is not feasible to predict the input given the output.

1.3. A plethora of learning models

There are several possible learning models which can be divided along several axes. Our first axis is the type of information given to the learning algorithm. There are several possibilities:

(1) Labeled Examples: Vectors of observations.
(2) Partial relations: partial relations between events such as might be provided by experts. This could include constraints or partial functions.
(3) Other forms of input

We will assume that just examples (vectors of observations) are available as a lowest common denominator amongst learning problems. It is worth noting that this does not preclude the use of other forms of information which could be much more powerful than mere examples.

Another important axis is the difficulty of the learning problem. Do we have an opponent trying to minimize learning? Is someone helping us learn? Or is the world oblivious?

(1) Teacher: The teacher model is a "best case" model. Here, we assume that someone is providing the best examples possible in order to learn a relationship.
(2) Oblivious: The oblivious model is an "in between" model where we assume that the world doesn't oppose or help us learn. Examples are picked in some neutral manner.
(3) Opponent: The opponent model is a "worst case" model. Here, we assume that world is choosing examples in way which minimize our chance of learning.

Clearly, the strongest form of learning is learning in the opponent model, because if something is learnable in the opponent model, then it is learnable in the oblivious model. The same relationship also holds for oblivious and teacher models. We will work in an oblivious model where examples are chosen in a neutral manner. Why the oblivious model? Aside from the intractability of analysis in an opponent model, we expect that most learning problems actually are oblivious: we have neither an active teacher nor an active opponent. Thus an analysis in the oblivious model will be directly applicable to many learning problems.

We have committed to an oblivious model with examples as our source of information. With these two questions decided all the remaining questions will essentially be decided in favor of simplicity. There are two more very important questions to decide. The first is: does our algorithm get to pick the examples or are the examples picked for us?

(1) Active learning: The learning algorithm chooses a partial example and the remainder is filled in by nature.
(2) Passive learning: The learning algorithm is simply given examples.

Active learning (aka experimental science) is inherently more powerful than passive learning. As an example, consider the problem of predicting whether or not it will rain or snow on any given day. By observation, we can eventually discover the "right" threshold temperature, but this might take many days. If we instead can control the temperature and make observations, it should be possible to narrow in on the threshold temperature very quickly - with exponentially fewer experiments than days of observation.

Despite the power of active learning, we will choose to work with an inactive learning model, because opportunities for passive learning are typically more common than opportunities for active learning since passive learning only requires observation while active learning requires experimentation. Analyzing the active learning setting in a generic manner also appears very difficult.

Our plan is to focus on an oblivious model with examples chosen by the world. The remaining question is: Do we know which relation we want to learn? The two possibilities are:

(1) Supervised learning: We want to learn to model an output in terms of an input.
(2) Unsupervised learning: We want to learn to model an arbitrary subset of observations in terms of other observations.

We will focus on supervised learning and our exact setting will be defined next.

The question we want to answer is, "When is supervised learning in an oblivious model with examples chosen by the world feasible?".

1.4. The oblivious passive supervised learning model

Oblivious will be modeled by an unknown distribution D over examples. Here, an "example" is just a vector of observations. Since this is a supervised learning model, all of our examples will split into two parts, (x, y) where x is the "input" and y is the "output" (the thing we wish to predict). A quick example is predicting whether precipitation will be in the form of rain or snow ("y" value) given the temperature ("x" value).

For simplicity, we will typically work with theorems for binary valued y. We can remove this choice by generalizing sample complexity bounds—but we do not do so for simplicity of presentation.

The fundamental assumption we will make in all of our sample complexity bounds is that all examples are drawn independently from the unknown distribution D. This assumption must be stated explicitly and always kept in mind when considering the relevance of sample complexity bounds.

AXIOM 1.4.1. *All examples are drawn independently from an unknown distribution D.*

With the exception of this assumption, all of the other parameters in our bounds will be verifiable at the time the bound is applied.

Note that we use a distribution over *labeled* examples and not a combination of a distribution over the input space along with a function from the input space to the output space as in many other formulations. This choice is made because it is both more general and mathematically simpler.

The number of samples, m, required for learning is the fundamental quantity we will be concerned with. In particular, we will not be concerned with the time complexity or the space complexity of learning algorithms. This choice is made for the purposes of simplicity and implies that the relationship between sample complexity bounds and learning algorithms will be similar to the difference between information theory and coding information for transmission across a noisy channel.

Any learning algorithm must output some hypothesis, h, for predicting the output given the input. This hypothesis is essentially a program which, given the input, predicts the output. The hypothesis may or may not be randomized—it might choose an output deterministically or according to some randomization.

The next item to quantify is learning. When has learning occurred? We will say that learning has occurred when the *true error* is significantly less than a uniform random prediction. The true error $e_D(h)$ is defined in the following way:

$$e_D(h) = \Pr_D(h(x) \neq y)$$

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. Given a sample set S of (x, y) pairs $\{(x_1, y_1), ..., (x_m, y_m)\}$, the *empirical error*, $\hat{e}_S(h)$ is defined similarly as:

$$\hat{e}_S(h) = \Pr_S(h(x) \neq y) = \frac{1}{m}\sum_{i=1}^{m} I(h(x_i) \neq y_i)$$

where $I()$ is a function which maps "true" to 1 and "false" to 0. Here $\Pr_S(...)$ is a probability taken with respect to the uniform distribution over the set of examples, S.

1.5. Questions we can answer

Our real goal in learning theory is to answer the question "When can we learn?" Unfortunately, there is no good answer to this question given only the assumption of independence. In particular, it may be impossible to learn. The simplest example of such a learning problem is the case of a distribution D which always flips a coin in deciding the value of the output y. The bias of the coin is the same irrespective of the input x. Since the value of the input is explicitly independent of the output we surely can not hope to learn a useful relation between the input and output.

Considerable work has been done elsewhere to answer "When can we learn?" Typically this is done in models with stronger assumptions—for example, under the additional assumption that the output is related to the input by an "OR" of a subset of the input bits.

We will instead focus on a different question: "Have we learned?" This question *is* answerable in a probabilistic manner. In particular we can make a statement such as "With high probability over samples drawn from D we have learned if the empirical error is less than some value." In practice, we will want to know how *much* we have learned which we can do by providing a high confidence bound on the true error rate of the learned hypothesis.

CHAPTER 2

Formal Model and Context

2.1. Formal Model

Let X be the space of the input to a predictor and Y be the space of the output. A labeled example (x, y) consists of an input, x and the desired output, y. Our formal model starts with the assumption that all labeled examples are drawn independently from a distribution D over the space $X \times Y$. This is strictly more general than the 'target concept' model which assumes that there exists some function $f : X \rightarrow Y$ used to generate the label [**50**]. In particular we can model probabilistic 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.

The set of m independently drawn samples presented to a learning algorithm will be denoted as S. The learning algorithm will output a hypothesis $h : X \rightarrow Y$ which has some unobservable true error rate $e_D(h)$ and an observable empirical error rate, $\hat{e}_S(h)$.

DEFINITION 2.1.1. (True error) The true error $e_D(h)$ of a hypothesis h is defined in the following way:

$$e_D(h) = \Pr_D(h(x) \neq y)$$

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 2.1.2. (Empirical Error) Given a sample set S, the *empirical error*, $\hat{e}_S(h)$ is defined as:

$$\hat{e}_S(h) = \Pr_S(h(x) \neq y) = \frac{1}{m} \sum_{i=1}^{m} I(h(x_i) \neq y_i)$$

where $I()$ is a function which maps "true" to 1 and "false" to 0. Here $\Pr_S(...)$ is a probability taken with respect to the uniform distribution over the set of examples, S.

2.2. Relationship to Prior Work

2.2.1. Distribution free learning. The question answered here differs significantly from much prior learning theory, including the results of Vapnik [**51**], Valiant [**50**], Devroye [**11**], and many others. See [**20**] for a good summary. The principle difference is the question we address: "Have we learned?"

Much of the prior work in learning theory addresses the question: "How many examples are needed in order to guarantee that I will choose (nearly) the best hypothesis from some fixed hypothesis set?"

In particular, suppose that we have a hypothesis set H. If we guarantee that:

$$\Pr_{S\sim D^m}(\exists h \in H: \ |e_D(h) - \hat{e}_S(h)| > \epsilon) < \delta$$

then if our learning algorithm choses the hypothesis with minimum empirical error (known as "empirical risk minimization" in the language of [**52**]), we will be guaranteed that the chosen hypothesis satisfies:

$$e_D(h) - e_D^* \leq 2\epsilon$$

where $e_D^* = \inf_{h\in H} e_D(h)$.

There are several difficulties inherent in this approach which we will avoid.

(1) Results in this model apply only for the empirical risk minimization (ERM) algorithm. The ERM algorithm is known to be NP-complete [**4**] for some hypothesis spaces and, in general, is essentially dependent upon the axiom of choice. These results will approximately apply to approximate ERM algorithms, but it is unclear how "approximate" typical learning algorithms are. By answering "Have we learned?" this complexity is avoided.
(2) There is no natural notion of preference (or "prior") amongst the hypotheses, h, in the hypothesis space, H. This is very important for practical application as is shown in Figure 12.3.3.
(3) Answers to this question are generally insensitive to the final result, $\hat{e}_S(h)$. This is again important in practice (shown here 3.4.1) because the variance of the distribution of the empirical error, $\hat{e}_S(h)$, changes significantly with different true error rates, $e_D(h)$.

These drawbacks can be alleviated (but not removed) to some extent. For example, many people apply results in this model to arbitrary learning algorithms by simply noticing that the deviation of the empirical and true error rates is small. This, in turn, implies that whatever hypothesis your algorithm learns (empirical minimum or not), it's true error is within ϵ of the empirical error. This is one approach to addressing the question "have we learned?" - others will be presented here.

The second drawback can be alleviated using "structural risk minimization" (as in [**52**]). Structural risk minimization removes most of drawback (2), although it is awkward for specifying very fine-grained preferences. We will use an arbitrary measure P over the hypothesis space h. This prior P need not (necessarily) be a Bayesian prior - all that is formally required is that this "prior" be specified without using information from the examples. The notion of measure P is more general than structural risk minimization because for *every* "structure" T on which structural risk minimization is done, we can produce a "prior" P_T dependent upon the structure T and derive the same (or tighter) results.

The third drawback can be alleviated using "relative risk" as in [**52**], but this still leaves some slack in the bounds.

The work in this thesis can be thought of as directly addressing the altered question which alleviates problem 1. Since our goal is addressing this altered question rather than deriving the answer from other results, we will be able to state and prove tighter results. Furthermore, because we address the question people encounter in practice, the bounds presented here will be more directly applicable. In particular, they will apply to arbitrary learning algorithms (although not necessarily *tightly* to arbitrary learning algorithms) rather than just the empirical risk minimization algorithm.

2.2.2. Bayesian Analysis. The basic result of Bayesian analysis is that Bayes rule:

$$\Pr(f|S) = \frac{\Pr(S|f)\Pr(f)}{\Pr(S)}$$

is the *optimal* learning algorithm when the learning problem is drawn from the distribution $\Pr(f)$. Note that the "hypothesis" learned by the Bayesian learning algorithm is the weighted average predictor,

$$h(x) = \text{sign}(\int \Pr(f|S)f(x)df)$$

This rather strong statement is difficult to utilize in practice because specifying and using arbitrary distributions $\Pr(f)$ is unwieldy. In practice, many people use approximations and there is considerable question about whether or not the specified $\Pr(f)$ is "right enough" after approximations. A chapter is later devoted to analyzing hypotheses of this weighted-average form and a theorem 7.2.1 about their accuracy is proved.

Some work has been done to analyze the robustness of Bayesian algorithms under approximation errors. There are two common traits of Bayesian-related analysis:

(1) All statements are parameterized by a prior, P.
(2) The analysis is typically an "average case" (w.r.t. the prior P and a fixed Bayesian or approximate Bayesian algorithm, A).

An example of this sort of analysis can be found in [**21**]. The work in this thesis adopts parameterization by a measure P, but is *not* an average case analysis. Our analysis is "worst-case" in the sense that it applies to *all* learning problems whether or not the measure P is a "correct" prior or not. This approach is strongly similar to the work of McAllester [**39**], and a later chapter is devoted to a refinement of this result. Despite this, it is interesting to note that our bounds are minimized (in some sense) when the measure P is in fact a "correct" Bayesian prior.

There have been other attempts to connect Bayesian average case settings with worst case settings. One interesting example is [**16**] which discusses the connection between Bayesian setting and the mistake bound model. This is especially interesting because the mistake bound model is, in some sense, more "worst-case" than the model we consider here as no assumption of example independence is made. Further work [**29**] has occurred in the "Minimum Description Length" (MDL) community.

2.3. Overview of the document

This document is primarily about the theory of sample complexity for answering the question "Have we learned?". However, we do not neglect the experimental side. In particular, following the theory we will present results for application of sample complexity bounds to machine learning problems. These results are the 'best known results' in terms of bound tightness and should be considered as a guide and challenge to others working on sample complexity bounds.

All of the sample complexity bounds presented here will fall within the paradigm of classical (non-Bayesian) statistics. Despite this, Bayesians may be interested in the results. In particular, it is worth noting that we will consider the use of a 'prior' and a 'posterior' (in a *classical* manner) within these bounds.

In order to make this thesis more coherent, previous work (of which there is quite a bit) will be integrated into the presentation rather than separated into a section of its own. Credit will be given at the time the work is introduced.

The document is organized into 3 parts:

(1) Introductory material.
(2) New results on Sample Complexity.
(3) Experimental results of applying Sample Complexity.

What follows is a brief chapter-by-chapter summary of the theoretical results in this thesis. Much of the work can be summarized as approaches which use extra information (in the algorithm, in error rates of hypotheses which are *not* chosen, in test sets, etc...) in order to construct tighter bounds on the future error rate of a hypothesis.

The principal *practical* result of this thesis is the construction of a program 'bound' which automatically uses any of several theorems in calculating a true error bound on the future error rate of a particular hypothesis. The use of this program will be demonstrated as the bounds are presented.

2.3.1. Microchoice Bounds.

The Microchoice technique allows for online construction of a "prior" which can be used in the Occam's Razor bound (4.6.1). Empirical testing (presented in chapter 11) shows that this approach is practical and yields useful results on decision trees for real-world problems drawn from the UCI database of problems. The Adaptive Microchoice bounds further extend this approach and can result in functional improvements over the Occam's Razor bound.

2.3.2. Pac-Bayes Bounds.

Pac-Bayes bounds are a new approach (first presented by David McAllester [**39**]) for for dealing with continuously parameterized classifiers such as (stochastic) neural networks. This chapter refines and improves the PAC-Bayes bound, giving it an information theoretic interpretation. Empirical results (presented in chapter 12) show that refined PAC-Bayes bound works to produce *nonvacuous* bounds on realistic learning problems. Nonvacuous bounds for continuous-valued classifiers are currently rare.

2.3.3. Averaging Bounds.

Averaging bounds deal with classifiers formed by picking according to the weighted majority on some other set of classifiers. Averaging is a very common technique in practice (see [**?**] for examples), so results specialized for averaging classifiers are useful. This chapters states and proves a bound on averaging classifiers which shows that "hypothesis space complexity" *decreases* as averaging becomes more uniform. Prior theoretical work [**?**] was of the form "averaging does not increase the hypothesis space complexity much".

2.3.4. Shell Bounds.

Shell bounds are a new approach which trades extra information for tighter bounds. Empirical results in (presented in chapter 11) show this can be a useful approach. In order to ameliorate the increased information requirements, a sampled version of the bound is stated which allows for smooth interpolation between simpler bounds and the full shell bound. In addition, the shell bound has been extended to continuous spaces with an approach similar to PAC-Bayes bounds.

2.3.5. Bracketing Covering Number Bounds. The results of this chapter are entirely theoretical. They point out an approach which may lead to useful technique for bounds on continuous valued hypothesis spaces. Using a stronger notion of cover (the bracketing cover), simplified bounds on the future error rate of continuous classifiers can be constructed. This approach can be significantly tighter than the standard covering number approach. More work in calculation of bracketing covers is needed before these results can be applied.

2.3.6. Progressive Validation. Progressive Validation is a variant of the holdout bound (4.1.1) which is roughly twice as efficient about how it uses examples. Progressive Validation is not used in the experimental results chapter(s) because more work is required in order to prove the bound without a Hoeffding-like approximation.

2.3.7. Combining training and test sets. The idea behind this chapter is that it should be possible to combine *any* test set bound with *any* training set based bound in order to derive a bound with more robust behavior. A general technique is stated and proved to work. Empirical results (in chapter 11) show that this approach can work well in practice.

CHAPTER 3

Basic Observations

The principle observable quantity is the empirical error rate ($\hat{e}_S(h)$) of a hypothesis. What is the distribution of the empirical error rate for a fixed hypothesis? For each example, we know that the probability that the hypothesis will err is given by true error rate, $e_D(h)$. This can be modeled by a biased coin flip: heads if you are wrong and tails if you are right.

Let us call the bias of the coin $p = e_D(h)$. What then is the probability of observing k heads out of m coin flips? This is a very familiar distribution in statistics called the Binomial distribution. Let $\hat{p}$ be the observed rate of heads.

DEFINITION 3.0.1. (Binomial Distribution) The Binomial distribution is given by:

$$\Pr_{\{0,1\}^m}\left(\hat{p} = \frac{k}{m}|p\right) = \binom{m}{k} p^k (1-p)^{m-k}$$

Here we use 'choose' notation defined by $\binom{m}{k} = \frac{m!}{(m-k)!k!}$.

3.1. The Basic Building Block

Our real interest will be captured by Binomial tails because we wish to bound the probability of observing a misleadingly small event. The probability of a Binomial tail is just the cumulative distribution function:

DEFINITION 3.1.1. (Binomial Tail)

$$\mathrm{Bin}(m,k,p) \equiv \Pr_{\{0,1\}^m}\left(\hat{p} \leq \frac{k}{m}|p\right) = \sum_{j=1}^{k} \binom{m}{j} p^j (1-p)^{m-j}$$

$=$ the probability that m coins with bias p produce k or fewer heads.

For the learning problem, we will always choose $p = e_D(h)$ and $\hat{p} = \hat{e}_S(h)$. With these definitions, we can interpret the Binomial tail as the probability of an empirical error less than or equal to $\frac{k}{m}$.

3.2. Approximation techniques

Exact calculation of $\mathrm{Bin}(m,k,p)$ (covered in the next subsection) can require computation at least proportional to m, which is often too expensive. For the bounds in this thesis, we will only need to calculate an upper bound on the quantity $\mathrm{Bin}(m,k,p)$. There are several inequalities which are often used. The first of these is the Hoeffding inequality[**23**]. Assume that $\frac{k}{m} < p$ then we have:

$$\mathrm{Bin}(m,k,p) \leq e^{-2m\left(p-\frac{k}{m}\right)^2}$$

Intuitively, this inequality can be seen as fitting a gaussian to the Binomial distribution with $p = \frac{1}{2}$. For any particular m, the variance of the Binomial distribution is maximized when $p = \frac{1}{2}$. Therefore, the Hoeffding inequality is relatively tight when $p = \frac{1}{2}$. Unfortunately, the Hoeffding approximation is not tight enough for our purposes. In machine learning, our goal is to find a hypothesis with a true error rate far away from $\frac{1}{2}$ where the Hoeffding inequality becomes loose.

There is another bound known as the "realizable bound" which applies only when $k = 0$. The realizable bound is:

$$\operatorname{Bin}(m, 0, p) = (1-p)^m \leq e^{-mp}$$

The realizable bound is noticeably tighter with an exponent proportional to $\left|p - \frac{k}{m}\right|$ rather than $\left(p - \frac{k}{m}\right)^2$. The disadvantage of the realizable bound is that it only applies to a very limited setting - when our empirical error rate happens to be 0.

Luckily, there exists a quickly calculable bound which achieves the generality of the Hoeffding bound along with the tightness of the realizable bound. We have the relative entropy Chernoff bound [**7**] for $\frac{k}{m} < p$:

$$\operatorname{Bin}(m, k, p) \leq e^{-m\mathrm{KL}\left(\frac{k}{m}||p\right)} \tag{3.2.1}$$

Here $\mathrm{KL}(q||p) = q \ln \frac{q}{p} + (1-q) \ln \frac{(1-q)}{(1-p)}$ is the KL-divergence between a coin of bias q and another coin of bias p. The relative Chernoff bound is as tight as the Hoeffding bound when p is near $\frac{1}{2}$ and as tight as the realizable bound when $k = 0$. In between the extremes, the relative Chernoff bound smoothly interpolates between these possibilities.

We are concerned with the different bounds here because much of the learning theory literature (see [**50**], [**20**], [**39**] for examples) works with either the realizable bound or the Hoeffding bound, or both. In contrast, we will work with either the relative Chernoff bound or the exact tail probability, $\operatorname{Bin}(m, k, p)$. There are several advantages to this approach:

(1) Sometimes, a different approach to producing a bound will appear better than previous approaches, but the apparent benefit can simply be traced to the use of a tighter bound on $\operatorname{Bin}(m, k, p)$.
(2) The bounds presented here will all be immediately applicable to direct calculation.
(3) We avoid the need to state two versions of the same theorem: once for the realizable (0 empirical error) case and once for the agnostic (arbitrary empirical error) case.

The principle *dis*advantage of this approach is that both the relative entropy Chernoff bound and $\operatorname{Bin}(m, k, p)$ are not analytically invertible. Lack of invertibility is a theoretical disadvantage because it means we can not easily parameterize our "precision" parameter, ϵ in terms of δ. Nonetheless, this is not a severe computational disadvantage because the quantity $\operatorname{Bin}(m, k, p)$ is convex in p implying that a binary search is capable of solving the inequality. The process of (and need for) inversion is discussed next.

3.3. Binomial Tail calculation techniques

How quickly can we calculate the binomial coefficient, $\binom{m}{k}$? This question is important to answer because we will apply the bounds derived later to real problems, and this application will require calculation of Binomial coefficients and Binomial tails.

The answer is: not very fast. It is an open problem to calculate $\binom{m}{k}$ in time not exponential in $\log m$ and $\log k$ (the representation length of m and k). In general, this problem is intractable. For example, we note that

$$\binom{m}{\frac{m}{2}} \simeq \frac{2^m}{\sqrt{m}}$$

which has an asymptotic representation length of $O(m)$. Since it takes $O(m)$ time to write the answer, we can not hope to compute the answer in less than $O(m)$ time. The problem is more difficult than this though - even calculating the most significant bits of the appears to take $O(m)$ time.

Our real goal is not merely calculating binomial coefficients but rather calculating the probability of a tail, $\text{Bin}(m, k, p)$. How can we calculate the tail probability quickly? For all approaches, it is necessary to calculate $\log \text{Bin}(m, k, p)$ rather than $\text{Bin}(m, k, p)$ to avoid underflow issues. This is generally possible because we can calculate $\log(a + b)$ given $\log(a)$ and $\log(b)$ without losing precision.

There are several possible approaches of increasing sophistication:

(1) Calculate $\binom{m}{i}$ independently from $i = 0$ to $i = k$ and use the results to calculate the sum, $\sum_{i=0}^{k} \binom{m}{i} p^i (1-p)^{m-i}$.
(2) Calculate Pascal's triangle and extract the Binomial coefficients.
(3) Use the fact that

$$\binom{m}{i+1} = \frac{m!}{(m-i-1)!(i+1)!}$$

$$= \frac{m-i}{i+1} \frac{m!}{(m-i)!i!} = \frac{m-i}{i+1} \binom{m}{i}$$

(4) Calculate $\binom{m}{k}$ directly and then $\binom{m}{i-1}$ given $\binom{m}{i}$ until the added quantity falls below the machine precision.

Approaches (1) and (2) both require $O(m^2)$ work while approaches (3) and (4) require merely $O(m)$ work. We will use approach (4) here. Yet, as noted in the beginning of this section, $O(m)$ is still sometimes too expensive for us. Luckily, there exist some quick approximations which can reduce the computation to constant time (or $O(\log m)$ time depending on your computational model).

3.4. Converting to a P-value approach

When making judgments about which hypothesis to choose, the relevant quantity is *not* the probability of error as we calculate above. Instead, it is a bound on the true error rate which holds with high probability over draws of the sample set. We might decide that $\delta = 0.05$ was an acceptable rate of bound failure and then ask ourselves, "What is a bound on the true error rate that holds with probability 0.9?"

Functionally, instead of calculating:

$$\text{Bin}(m, k, p) = \delta$$

we want to invert the output, δ, with respect to the input, p. Since p and δ are monotonically related to each other, this inversion can be defined as:

$$\bar{e}(m,k,\delta) \equiv \max_p \{p: \ \mathrm{Bin}(m,k,p) = \delta\}$$

What is the interpretation of $\bar{e}$? The inversion $\bar{e}(m,k,\delta)$ is a high confidence bound on the true error rate. With probability at least $1-\delta$ (over the draws of the examples), the true error rate will be less than $\bar{e}(m,k,\delta)$. This is exactly the kind of quantity that we desire in making decisions about which hypothesis is more desirable. This calculation has been done for several values of m and Binomial tail bounds in figure (3.4.1).

We will use the process of inversion in many places. The fundamental soundness of inversion rests upon the following lemma.

LEMMA 3.4.1. *(Inversion Lemma) For all predicates, $\phi(X)$*

$$\sup_P \Pr_{X\sim P}(\phi_P(X)) \le \delta \Rightarrow \forall P \ \ \Pr_{X\sim P}(\phi_P(X)) \le \delta$$

This lemma is the trivial statement that a set of objects is less than the sup over the set of objects. Nonetheless, it is a very important step which will appeal to implicitly and explicitly later on.

PROOF. By contradiction. Assume there exists P such that the right hand side is not satisfied. Then, the left hand side can not be correct. □

The Inversion lemma allows us to implicitly parameterize *all* precision parameters $\epsilon(\delta)$ in terms of a fixed probability of failure, δ. This is important for practical application because it means we can choose our probability of failure δ before looking at any examples.

3.5. Bounding the Union

One very common technique we will use is the union bound (known as the Bonferroni bound in statistics). Given two coins, each with a bias (probability of heads) of p, what is the probability that if we flip each coin m times, one of the coins will have k or fewer heads?

Let X_1 = the proportion of heads in the first coin flip and X_2 = the proportion of heads in the second coin flip. Then we get:

$$\Pr\left(X_1 \le \frac{k}{m} \text{ or } X_2 \le \frac{k}{m}\right)$$

$$\le \Pr\left(X_1 \le \frac{k}{m}\right) + \Pr\left(X_2 \le \frac{k}{m}\right)$$

where the inequality is known as the union bound. This step is very applicable because it works even when the values of X_1 and X_2 are correlated in arbitrary ways.

The union bound is the fundamental tool which allows us to reason about multiple hypotheses, each with possibly correlated empirical errors, $X_1 = \hat{e}_S(h_1)$ and $X_2 = \hat{e}_S(h_2)$. The use (and avoidance of the use) of the union bound is a constant issue.

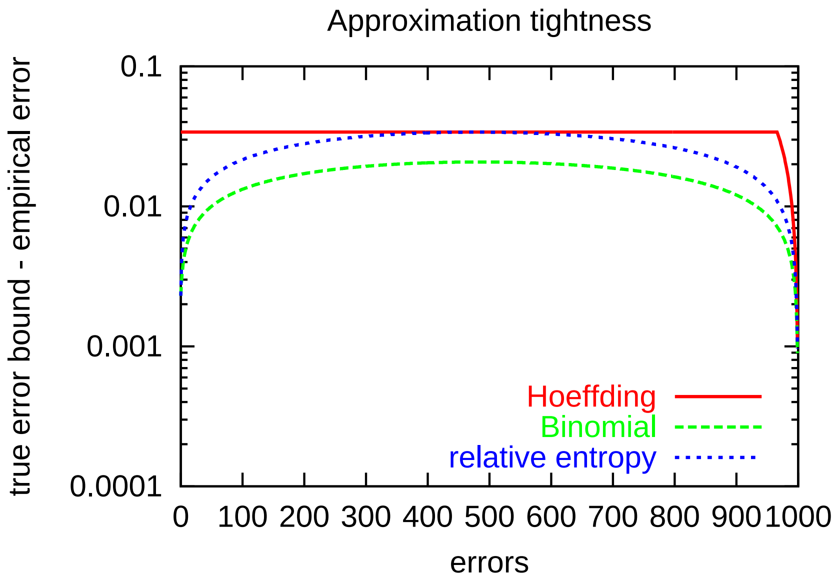


FIGURE 3.4.1. A plot of the difference between the true error upper bound and the empirical error for various Binomial tail bound approximations. Here $m = 1000$ coins of an unknown bias are used and a confidence of $\delta = 0.1$ and the horizontal axis is the number of errors between $k = 0$ and $k = 1000$. The relative entropy approximation to the Binomial tail is relatively well behaved while the Hoeffding approximation is not. In particular, the Hoeffding approximation does not take into account the decreased variance of low bias Binomials. Note that the dip at the end of the Hoeffding bound is due to the fact that the true error rate is *always* less than 1.

3.6. Arbitrary Loss functions

A loss function is *any* function which takes a hypothesis, h, and an example (x, y) as input then outputs a real number. In particular, we could choose $l(h,(x,y)) = I(h(x) \neq y)$ and regard most of the prior discussion as working with a specialized hamming loss function which is 1 when $h(x) \neq y$ and 0 otherwise. Many other possibilities exist. For any bounded loss function, $l(h,(x,y)) \in [0,1]$, we can define:

$$e_D(h) = E_D l(h,(x,y))$$

and

$$\hat{e}_S(h) = E_S l(h,(x,y))$$

All of the bounds reported here will apply for loss functions bounded on the interval $[0,1]$. The fundamental advantage that this gives us is the ability to treat the hypothesis *not* as a black box. Instead, we can derive a bound with a loss function dependent on the structure of the hypothesis. This will be important later when discussing averaging bounds (theorem 7.1.1) where the bound will partly depend upon the structure of the hypothesis.

For clarity of presentation, the bounds will all be presented using the hamming loss. However, they will all apply to the more general setting of arbitrary $0-1$ loss functions.

CHAPTER 4

Simple Sample Complexity bounds

The observations presented in this chapter are mostly "common knowledge" in the learning theory community. The goal of this chapter is to introduce basic common learning theory which the remainder of the thesis will depend upon.

4.1. Simple Holdout

The simplest bound arises for the classical technique of splitting the data set into two pieces: a training set of size m_{train} and a test set of size m_{test}. In this setting, the following simple bound applies:

THEOREM 4.1.1. *(Holdout Sample Complexity) Let $\hat{e}_{test}(h)$ be the empirical error on the test set and $e_D(h)$ be the true error rate of the hypothesis, then we have:*

$$\forall h \ \Pr_{D^m} \left(\ e_D(h) \geq \bar{e}\left(m_{test}, \hat{e}_{test}(h), \delta\right)\right) \leq \delta$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \max_p \{p : \ Bin(m,k,p) = \delta\}$

PROOF. The proof is just a simple identification with the Binomial. For any distribution over (x, y) pairs and any hypothesis, h, there exists some probability, $e_D(h)$, that the hypothesis predicts incorrectly. We can regard this event as a coin flip with bias $e_D(h)$. Since each example is picked independently, the distribution of the empirical error rate will then be a Binomial distribution. Given that the distribution is Binomial we calculate an upper bound which holds with high probability. □

There are two immediate corollaries of the holdout theorem (4.1.1) which are mathematically simpler although not as tight. The first corollary applies to the limited "realizable" setting where you happen to observe 0 test errors.

COROLLARY 4.1.2. *(Realizable Holdout Sample Complexity)*

$$\forall h \ \Pr_{D} \left(\hat{e}_{test}(h) = 0 | e_D(h) \geq \frac{\ln \frac{1}{\delta}}{m_{test}} \right) \leq \delta$$

PROOF. Specializing theorem 4.1.1 to the zero empirical error case, we get:

$$\text{Bin}(m_{\text{test}}, 0, \epsilon) = (1-\epsilon)^{m_{\text{test}}} \leq e^{-\epsilon m_{\text{test}}}$$

Setting this equal to δ and solving for ϵ gives us the result. □

A second corollary applies to all results, not just those where we observe 0 errors.

COROLLARY 4.1.3. *(Agnostic Holdout Sample Complexity)*

$$\forall h \ \Pr_D(e_D(h) - \hat{e}_{test}(h) \geq \sqrt{\frac{\ln\frac{1}{\delta}}{2m_{test}}}) \leq \delta$$

PROOF. Loosening theorem 4.1.1 with the Hoeffding approximation for $\frac{k}{m_{\text{test}}} < \epsilon$, we get:

$$\mathrm{Bin}(m_{\text{test}}, k, \epsilon) \leq e^{-2m_{\text{test}}(\epsilon - \frac{k}{m_{\text{test}}})^2}$$

Using the inversion lemma 3.4.1 we can set this equal to δ, and solve for ϵ to get the result. □

REMARK 4.1.4. Similar theorems apply to bound $\hat{e}_{\text{test}}(h) - e_D(h)$.

How tight is the test sample complexity theorem 4.1.1? The answer is very tight. Let us define

$$\bar{e}(m, \hat{e}(h), \delta) \equiv \max_p p : \ \mathrm{Bin}(m_{\text{test}}, m_{\text{test}}\hat{e}_S(h), p) \geq \delta$$

as our true error bound. We wish to know how much $e_D(h)$ and $\bar{e}(m_{\text{test}}, \hat{e}(h), \delta)$ differ. Applying the Hoeffding approximation, we know that with high probability, $e_D(h) \geq \bar{e}(m_{\text{test}}, \hat{e}(h), \delta) - 2\sqrt{\frac{\ln\frac{1}{\delta}}{2m_{\text{test}}}}$. Thus the region in which $e_D(h)$ is confined with high confidence is of size $2\sqrt{\frac{\ln\frac{1}{\delta}}{2m_{\text{test}}}}$ or smaller.

It is common practice in the field of machine learning to use the gaussian approximation in reporting error bars. The practice is reasonably safe because it is usually pessimistic. However, this can occasionally lead to embarrassing results where error rates such as 0.01 ± 0.02 are reported. The test sample complexity theorem *never* produces an upper bound greater than 1 or lower bound less than 0 because it uses the fundamental Binomial distribution. This approach is the "right" way to report test-set based errors, given the assumption of independence. Appendix Section 16.1 documents how to apply this bound. Pictorially we can represent this as in figure 4.1.1.

Some results for application of the simple test set bound are presented on page 106 in figure 12.3.3. In summary, the test set bound tends to work quite well (in practice) when sufficient examples are available.

Given that the bounds for the simple holdout technique are so tight, why do we need to engage in further work? There is one serious drawback to the holdout technique—application of the holdout technique requires m_{test} otherwise unused examples. This can strongly degrade the value of the learned hypothesis because an extra m_{test} examples for the training set could reduce the true error of the learned hypothesis from 0.5 to 0.0 on some learning problems.

There is another reason why training set based bounds are important. Many learning algorithms implicitly assume that the training error "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 training error and the true error. Training error based bounds can be used *in* the training algorithm.

There are two basic approaches to this difficulty:

(1) Try to reduce m_{test} using more sophisticated holdout techniques.
(2) Do not use a holdout set. Instead, train and test on the same set of examples using a more sophisticated bound.

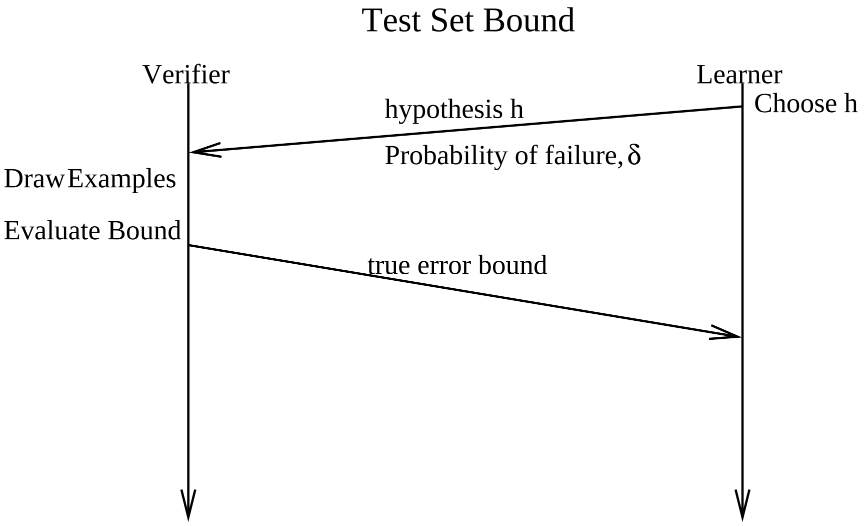


FIGURE 4.1.1. For this diagram "increasing time" is pointing downwards. The only requirement for applying this bound is that the learner must commit to a hypothesis without knowledge of the test examples. Similar diagrams for other bounds will be presented later (and they are somewhat more complicated). We can think of the bound as a technique by which the "Learner" can convince the "Verifier" that learning has occurred. Each of the proofs in this thesis can be thought of as a communication protocol for an interactive proof of learning by the Learner.

Before discussing approach (2) we will make a few comments about approach (1) to suggest the variety of theoretical difficulties which occur when using approach (1).

4.1.1. Cross Validation. One of the standard techniques for attempting to improve on the holdout bound is cross validation. K-fold cross validation divides the data into K folds of size $\frac{m}{K}$ (assume m is divisible by K for simplicity). Then, for every fold i, holdout fold i, train on the remainder of the data and test on fold i. Let the hypotheses we found by training be known as $h_1, ..., h_K$ and their respective holdout errors as $\hat{e}_1, ..., \hat{e}_K$. Also let $\hat{e}_{cv} = \frac{1}{K}\sum_{i=1}^{K} \hat{e}_i$.

There are several variations of cross validation. If $K = m$, the procedure is often called "leave one out cross validation". In one variant, you train on all of the data to learn a new hypothesis, h, and assume a true error rate near $\hat{e}_{CV}$. In another variant, you predict according to $h_{cv} = Uniform(h_1, ..., h_K)$. The latter variant is simpler to analyze because linearity of expectation implies that $\hat{e}_{cv}$ is an unbiased estimate of $\bar{e}_{cv}$.

There are strong results known for cross validation on nearest-neighbor, kernel, and histogram classifiers [**11**]. In general, only very weak results are known about bounds on the variance of cross validation for general classifiers. The "general" results include "Sanity check bounds" [**27**] which state that cross validation is not much worse than a holdout set and some slightly stronger results [**?**] and [**25**].

PROBLEM 4.1.5. (Open) Construct a bound on the deviation of cross validation for arbitrary classifiers which is a quantitative improvement on the results of [?].

4.2. The basic training set bound

The most basic of training set based sample complexity bounds is the simple combination of a Binomial tail bound and the union bound. In particular, we have:

THEOREM 4.2.1. *(Discrete Hypothesis Bound) For all hypothesis spaces, H, for all $\delta \in (0,1]$*

$$\Pr_{D^m}\left(\exists h \in H: \; e(h) \geq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \leq \delta$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \max_p\{p: \; Bin(m,k,p) = \delta\}$

Note that this theorem can only be nonvacuous when the hypothesis space, H, has some finite (discrete) size.

PROOF. For every individual hypothesis, we know that:

$$\forall h \;\; \Pr_{D^m}\left(e(h) \geq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \leq \frac{\delta}{|H|}$$

Applying the union bound (see section 3.5) for every hypothesis gives us:

$$\Pr_{D^m}\left(\exists h \in H: \; e(h) \geq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \leq \delta$$

which is the result. □

Intuitively, this theorem says that as the number of hypotheses grows, we can not guarantee that the empirical error will be near to the true error.

A better understanding can be gained by considering some of the Binomial tail bound approximations. This is also worth mentioning in order to compare this theorem with theorems in more common forms.

COROLLARY 4.2.2. *(Relative Entropy Discrete Hypothesis Bound) For all hypothesis spaces, H, for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists h \in H: \; KL(\hat{e}(h)||e(h)) \geq \frac{\ln|H| + \ln\frac{1}{\delta}}{m}\right) \leq \delta$$

PROOF. Loosen (theorem 4.2.1) with the relative entropy Chernoff bound (Eqn. 3.2.1), and use the inversion lemma 3.4.1. □

The form of this corollary allows us to make two more important observations:

(1) The "cost" of doubling the hypothesis space size is about one extra example. In other words, we can keep a constant bound on the probability of a large deviation with $c\log|H|$ hypotheses (for any c).
(2) The value of δ is not very important as m grows larger.

To understand this lemma, it is helpful to consider some approximations of $\mathrm{KL}(\hat{e}(h)||e(h))$. In particular, we have:

$$|\hat{e}(h) - e(h)| \geq \mathrm{KL}(\hat{e}(h)||e(h)) \geq 2(\hat{e}(h) - e(h))^2$$

which implies that the KL-divergence varies between an l_1 and an l_2 metric.

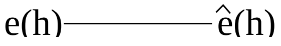
We can further loosen the last corollary with the Hoeffding approximation to get the following commonly stated bound:

COROLLARY 4.2.3. *(Agnostic Discrete Hypothesis Bound) For all hypothesis spaces, H, for all $\delta > 0$,*

$$\Pr_{D^m}\left(\exists h \in H : e(h) \geq \hat{e}(h) + \sqrt{\frac{\ln|H| + \ln\frac{1}{\delta}}{2m}}\right) \leq \delta$$

PROOF. Loosen corollary (4.2.2) with the bound $\mathrm{KL}(q||p) \geq 2(q-p)^2$. □

The agnostic form of this bound has the advantage that it explicitly shows that we are forcing the convergence (in high probability) of the empirical error rate to the true error rate. Graphically, we are forcing every empirical error to be near to its true error. This is a picture which represents the connection



Later bounds will have more complicated convergence conditions with more complicated graphs. These more complicated bounds are necessary in order to avoid the limitations of the holdout bound. See figure 12.3.3 for a comparison with the holdout set.

4.3. Lower Bounds

It is important to study lower bounds in order to discover how much room for improvement exists in our upper bounds.

There are two forms of lower bounds: a lower bounds on the true error rate and lower *upper* bounds which lower bound the value our upper bound could return (given available information). For lower bounds essentially the upper bound theorem applies with the bound reversed. This form of lower bound allow us to construct a two-sided confidence interval on the location of the true error rate. Typically, such lower bounds can be formed by simply changing the sign in upper bound arguments.

THEOREM 4.3.1. *(Discrete Hypothesis Lower Bound) For all hypothesis spaces, H, for all $\delta \in (0,1]$*

$$\Pr_{D^m}\left(\exists h \in H : \; e(h) \leq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \leq \delta$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \min_p\{p : \; Bin(m,k,p) = \delta\}$

PROOF. For every individual hypothesis, we know that:

$$\Pr_{D^m}(e(h) \leq \bar{e}(m, \hat{e}(h), \delta)) \leq \delta$$

Applying the union bound for every hypothesis gives us the result. □

4.4. Lower Upper Bounds

The second form of lower bound is a lower bound on the upper bound (similar to the results of [**14**]). If we can lower bound the upper bound (as a function of its observables), then we can be confident that the upper bound is no looser than the gap between the lower upper bound and the upper bound.

How tight is the discrete hypothesis bound (4.2.1)? The answer is *sometimes* tight. In particular, we can exhibit a set of learning problem where the discrete hypotheses bound can not be made significantly lower as a function of the observables, m, δ, $|H|$, and $\hat{e}(h)$. Fix the value of these quantities and then we will construct a learning problem for which a lower upper bound can not be stated.

Our learning problem will be defined over the input space $X = \{0,1\}^{|H|}$. The hypotheses will be $h_i(x) = x_i$ where x_i is the ith component of the vector x. This construction allows us to vary the true error rate of each hypothesis independent of the other hypotheses. In fact, we can pick any true error rate for any hypothesis by simply adjusting the probability that $x_i = y$. Our learning problem can therefore generate problems according to the following algorithm:

ALGORITHM 4.4.1. *Draw_ Sample(float e)*

```
(1) Pick $y \in \{0,1\}$ uniformly
(2) For $i$, pick $x_i \neq y$ with probability $e$.
```

By construction, the true error rate of each hypothesis will be $e(h_i)$. Now, we can prove the following theorem:

THEOREM 4.4.2. *(Discrete Hypothesis lower upper bound) For all true error rates* $e(h) > \frac{\ln|H| + \ln\frac{1}{\delta}}{m}$ *there exists a learning problem and algorithm such that:*

$$\Pr_{D^m}\left(\exists h \in H :\ e(h) \geq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \geq 1 - e^{-\delta}$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \max_p\{p :\ Bin(m,k,p) = \delta\}$

Intuitively, this theorem implies that we can not improve significantly on the results of theorem 4.2.1 without using extra information about our learning problem. Some of our later results do exactly this - they use extra information.

PROOF. Using the family of learning problems implicitly defined by algorithm 4.4.1, we know that

$$\forall h, \forall \delta \in [0,1] :\ \Pr_{D^m}\left(e(h) \geq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \geq \frac{\delta}{|H|}$$

(negation)

$$\Rightarrow \forall h, \forall \delta \in [0,1] :\ \Pr_{D^m}\left(e(h) < \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) < 1 - \frac{\delta}{|H|}$$

(independence)

$$\Rightarrow \forall \delta \in [0,1] :\ \Pr_{D^m}\left(\forall h\ e(h) < \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) < \left(1 - \frac{\delta}{|H|}\right)^{|H|}$$

(negation)

$$\Rightarrow \forall \delta \in [0,1] :\ \Pr_{D^m}\left(\exists h\ e(h) \geq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \geq 1 - \left(1 - \frac{\delta}{|H|}\right)^{|H|}$$

(approximation)

$$\Rightarrow \forall \delta \in [0,1] :\ \Pr_{D^m}\left(\exists h\ e(h) \geq \bar{e}\left(m, \hat{e}(h), \frac{\delta}{|H|}\right)\right) \geq 1 - e^{-\delta}$$

□

For small δ, we get that $1 - e^{-\delta} \simeq \delta$ which implies that, no significantly better true error bound can be stated for all learning algorithms. In particular, the empirical risk minimization algorithm (which chooses the hypothesis with minimal empirical error) will have a significant probability of a large deviation between the empirical and true error.

4.5. Structural Risk Minimization

Structural Risk Minimization [**51**] (SRM) is a technique used in the learning theory community to avoid the difficulties associated with convergence on hypothesis sets that are too "large". SRM works with a sequence of nested hypothesis sets, $H_1 \subset H_2 \subset .... \subset H_l$. For each hypothesis set, a discrete hypothesis bound (4.2.1) on the difference between empirical and true error exists. For "small" hypothesis sets, this bound may be tight while for large hypothesis sets it may be inherently loose. However, we also expect that the best hypothesis in the hypothesis set improves as the hypothesis set becomes larger. This naturally induces a trade-off: there will be some hypothesis set H_i for which the true error bound is minimized.

We can't simply apply the discrete hypothesis bound to the meta-algorithm which picks the algorithm (and associated hypothesis space) with the smallest true error bound since this meta-algorithm could, potentially, output any hypothesis in H_l. The simplest way to retrofit the bound to include all hypothesis sets is with a simple theorem which essentially states that we can guarantee *nearly* the same bound as would apply on the smallest hypothesis space H_i containing the output hypothesis, h.

THEOREM 4.5.1. *(Structural Risk Minimization) Let $p(i)$ be some measure across the l hypothesis sets with $\sum_{i=1}^{l} p(i) = 1$. Then:*

$$\forall p(i): \Pr_{D^m}\left(\exists h \in H_i \in \{H_1, ..., H_l\}: \ e(h) \le \bar{e}\left(m, \hat{e}(h), \frac{p(i)\delta}{|H_i|}\right)\right) \le \delta$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \min_p\{p: \ Bin(m,k,p) = \delta\}$.

PROOF. Apply the union bound to the discrete hypothesis bound (4.2.1). □

The SRM bound is slightly inefficient in the sense that the bound for all hypotheses in H_2 includes a bound for every hypothesis in H_1. This effect is typically small because the size of the hypothesis sets usually grows exponentially, implying that the extra confidence given to a hypothesis h in H_1 by the bounds used on hypothesis set $H_2, H_3, ...$ is small relative to the confidence given by the bound for H_1. One can remove this slack in Structural Risk Minimization bound by "cutting out" the nested portion of each hypothesis set in the formulation of $H_1, ..., H_l$. We will call this Disjoint Structural Risk Minimization (also mentioned in [**?**]).

4.6. Incorporating a "Prior"

In constructing the discrete hypothesis bound (4.2.1), it is notable that an arbitrary choice was made. We decided to give the same error allowance to every hypothesis. This is an arbitrary choice which, in practice, we will wish to make differently. The next theorem is essentially a restatement of the discrete hypothesis bound with this arbitrary choice made explicit. The same bound using the Hoeffding approximation has appeared elsewhere [**5**][**39**].

THEOREM 4.6.1. *(Occam's Razor Bound) For all hypothesis spaces, H, for all "priors" $p(h)$ over the hypothesis space, H, for all $\delta \in (0,1]$:*

$$\forall p(h) \ \Pr_{D^m}\left(\exists h \in H :\ e(h) \geq \bar{e}\left(m, \hat{e}(h), \delta p(h)\right)\right) \leq \delta$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \max_p\{p :\ Bin(m,k,p) = \delta\}$

It is very important to notice that the "prior" $p(h)$ must be selected *before* seeing the examples.

PROOF. The proof again starts with the basic observation that:

$$\Pr_{D^m}\left(e(h) \geq \bar{e}(m, \hat{e}(h), \delta)\right) \leq \delta$$

then, we apply the union bound in a *nonuniform* manner. In particular, we allocate confidence $\delta p(h)$ to hypothesis h. Since p is normalized, we know that

$$\sum_h \delta p(h) = \delta$$

which implies that the union bound completes the proof. □

Once again, we can relax the Occam's Razor bound (theorem 4.6.1) with the relative entropy Chernoff bound (3.2.1) to get a somewhat more tractable expression.

COROLLARY 4.6.2. *(Relative Entropy Occam's Razor Bound) For all hypothesis spaces, H, for all "priors" $p(h)$ over the hypothesis space, H, for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists h \in H :\ KL(\hat{e}(h)||e(h)) \geq \frac{\ln \frac{1}{p(h)} + \ln \frac{1}{\delta}}{m}\right) \leq \delta$$

PROOF. approximate the Binomial tail with (3.2.1) and solve for the minimum. □

The Occam's razor bound is often nonvacuous for discrete learning algorithms such as decision lists and decision trees. The next chapter will discuss a particular motivated choice of the measure $p(h)$ which can lead to much tighter bounds in practice.

The application of the Occam's Razor bound is somewhat more complicated then the application of the test set bound. Pictorially, the protocol for bound application is given in figure 4.6.1.

Examples of calculation of these bounds is detailed in appendix section 16.2.

The "Occam's Razor bound" is strongly related to compression. In particular, for any self-terminating description language, $d(h)$, we can associate a "prior" $p(h) = 2^{-d(h)}$ with the property that $\sum_h p(h) \leq 1$. Consequently, short description length hypotheses will tend to have a tighter convergence and the penalty term, $\ln \frac{1}{p(h)}$ is the number of "nats" (bits base e).

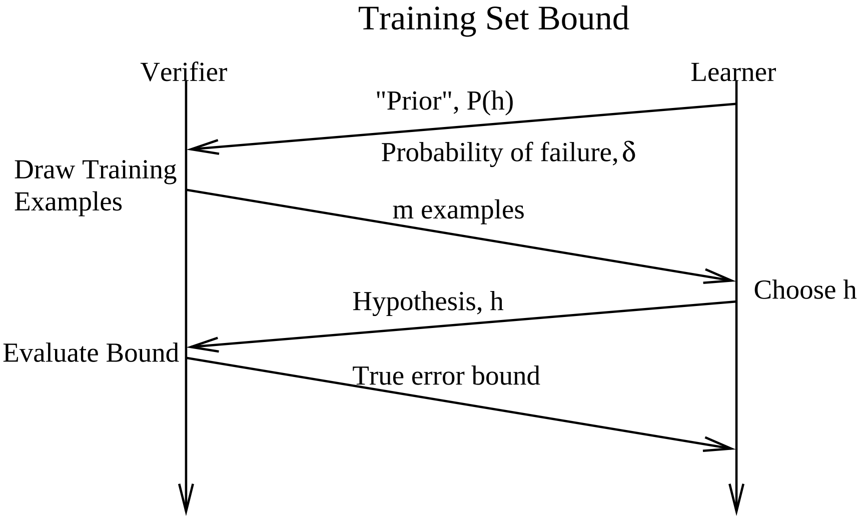


FIGURE 4.6.1. In order to apply this bound it is necessary that the choice of "Prior" be made before seeing any training examples. Then, the bound is calculated based upon the chosen hypothesis. Note that it *is* "legal" to chose the hypothesis based upon the prior $p(h)$ as well as the empirical error $\hat{e}_S(h)$.

Part 2

New Techniques

CHAPTER 5

Microchoice Bounds (the algebra of choices)

The work in this chapter is joint with Avrim Blum and was first presented at ICML [**30**] and then in the Machine Learning Journal [**31**]. The presentation here generalizes, unifies, and improves the earlier work. Microchoice bounds can be thought of as unifying "Self-bounding Learning Algorithms" [**17**] and the Occam's razor bound, [**5**].

The bounds of the previous chapter do not use much structure on the hypothesis space. Yet learning algorithms induce a natural structure. In particular, many learning algorithms work by an iterative process in which they take a sequence of steps each from a small set of choices (small in comparison to the overall hypothesis set size). Local optimization algorithms such as hill-climbing or simulated annealing, for example, work in this manner. Each step in a local optimization algorithm can be viewed as making a choice from a small set of possible steps to take. If we take into account the number of choices made and the size (and other properties) of each choice set, can we produce a tighter bound? This chapter introduce microchoice bounds which use this type of information to construct high confidence bounds on the future error rate.

The microchoice bounds can be thought of in several ways. The simple microchoice bound (given in section 5.2) can be thought of as a well motivated application of the Occam's Razor bound (4.6.1). The idea is to use the learning algorithm itself to define a description language for hypotheses, so that the description length of the hypothesis actually produced gives a bound on the estimation error. The adaptive microchoice theorem (in section 5.3) can be thought of as a computationally tractable adaptation of Self-bounding Learning algorithms [**17**]. Microchoice bounds tie together and show the relationship between these different sample complexity bounds.

Microchoice bounds also provide insight into the nature of choices. In general, we know that choice is "bad" for the purposes of creating a uniform bound on the true error rate. The microchoice bounds give a quantitative understanding of how much choice is "bad". In particular, the log of the choice space size is the natural measure of "badness". This is directly related to the log of the hypothesis space size in the discrete hypothesis bound (4.2.1). There is also an indirect relationship with information theory where the log of the alphabet size is an important parameter for specifying the number of bits required to send a message.

Viewed as an interactive proof of learning, microchoice bounds can be described pictorially as in figure 5.0.1.

Important early work developing approximately self-bounding learning algorithms was also done by Domingos [**12**].

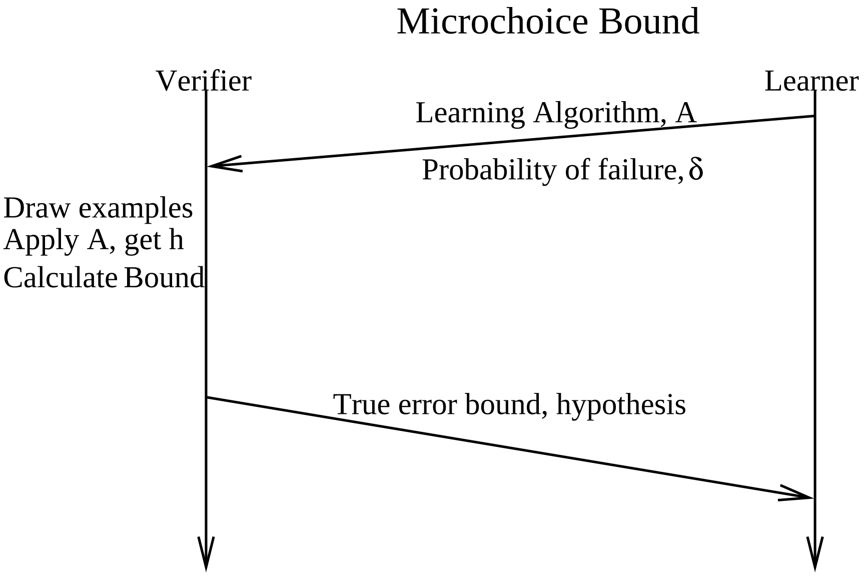


FIGURE 5.0.1. Microchoice bounds collapse the two-round Occam's Razor style protocol into a one round protocol. This is (essentially) done by providing a compiler for the verifier which takes a learning algorithm as input, and extracts a choice of "prior".

5.1. A Motivating Observation

Imagine, for the moment, that we know the (unknown) problem distribution, D. For a given learning algorithm A, a distribution D on labeled examples induces a distribution $q(h)$ over the possible hypotheses $h \in H$ produced by algorithm A after m examples[1]. A natural choice for the Occam's Razor bound (4.6.1) is the measure $p(h) = q(h)$. Is this choice optimal? The answer is "yes", given the right notion of optimal. In particular, if we start with the relative entropy Occam's razor bound (4.6.2), we can show that $p(h) = q(h)$ minimizes the expected value of the bound on the Kullback-Leibler divergence between the empirical error and true error.

THEOREM 5.1.1. *(KL divergence minimization) $p(h) = q(h)$ minimizes the expected value of the KL divergence in the relative entropy Occam's Razor bound (4.6.2).*

PROOF. We need to show that

$$q(h) = \text{argmin}_{p(h)} \sum_h q(h) \frac{\ln \frac{1}{p(h)} + \ln \frac{1}{\delta}}{m}$$

removing terms which the minimum does not depend on, we get:

$$\text{argmin}_{p(h)} \sum_h q(h) \ln \frac{1}{p(h)}$$

[1] $q(h)$ is the probability over draws of examples and any internal randomization of the algorithm.

adding a constant, we get:

$$\operatorname{argmin}_{p(h)} \sum_h q(h) \ln \frac{q(h)}{p(h)}$$

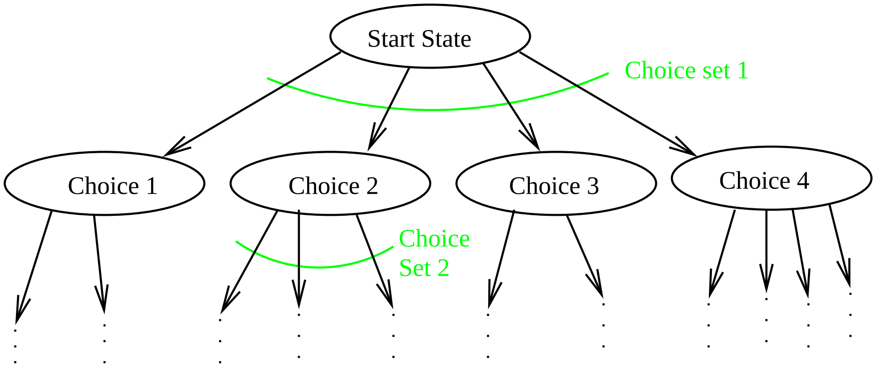
This is equivalent to minimizing the Kullback-Leibler divergence between the distribution of $q(h)$ and $p(h)$ which is minimized for $q(h) = p(h)$. □

Using the KL divergence as our notion of loss is somewhat non-intuitive. However, it is mathematically simple and not irrational. For all true errors, the KL divergence will be upper and lower bounded between an l_1 and an l_2 metric. Since these are two of the most common metrics, the choice of KL divergence based metric should behave similarly well.

The point of these observations is to notice that if the structure of the learning algorithm produces a choice $p(h)$ that approximates $q(h)$, the result should be better estimation bounds.

5.2. The Simple Microchoice Bound

The simple microchoice bound is essentially a compelling and easy way to select a measure $p(h)$ for learning algorithms that operate by making a series of small choices. In particular, consider a learning algorithm that works by making a sequence of choices, $c_1, ..., c_d$, from a sequence of choice sets, $C_1, ..., C_d$, finally producing a hypothesis, $h \in H$. Specifically, the algorithm first looks at the choice set C_1 and the data z^N to produce choice $c_1 \in C_1$. The choice c_1 then determines the next choice set C_2 (different initial choices produce different choice sets for the second level). The algorithm again looks at the data to make some choice $c_2 \in C_2$. This choice then determines the next choice set C_3, and so on. These choice sets can be thought of as nodes in a *choice tree*, where each node in the tree corresponds to some internal state of the learning algorithm, and a node containing some choice set C has branching factor $|C|$. Pictorially, we can draw the tree as follows:



Depending on the learning algorithm, sub-trees of the overall tree may be identical. We address optimization of the bound for this case later. Eventually there is a final choice leading to a leaf, and a single hypothesis is output.

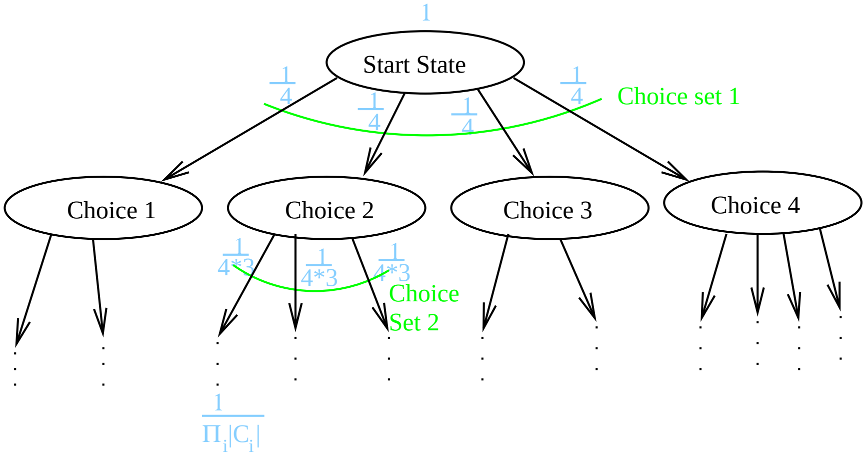
For example, the decision list algorithm of Rivest [**45**], applied to a set of n features, uses the data to choose one of $4n + 2$ rules (e.g., "if $\bar{x}_3$ then $-$") to put at the top. Based on the choice made, it moves to a choice set of $4n - 2$ possible rules to put at the next level, then a choice set of size $4n - 6$, and so on, until eventually it chooses a rule such as "else $+$" leading to a leaf.

The microchoice bound calculation program is as follows:

ALGORITHM 5.2.1. *Calculate_ Microchoice*

```
(1) set $p \leftarrow 1$
(2) while learning algorithm has not halted.
    (a) $|C| \leftarrow$ number of possible data-dependent choices
    (b) $p \leftarrow \frac{p}{|C|}$
(3) return $p$
```

Pictorially, this algorithm can be thought of as taking a "supply" of probability at the root of the choice tree.



The root takes its supply and splits it equally among all its children. Recursively, each child then does the same: it takes the supply it is given and splits it evenly among its children, until all of the supplied probability is allocated among the leaves. If we examine some leaf containing a hypothesis h, we see that this method gives at least probability $p(h) = \prod_{i=1}^{d(h)} \frac{1}{|C_i(h)|}$ to each h for any path of depth $d(h)$ reaching the hypothesis h.

Note it is possible that several leaves will contain the same hypothesis h, and in that case one should really add the allocated measures together. However, the microchoice bound neglects this issue, implying that it will be unnecessarily loose for learning algorithms which can arrive at the same hypothesis in multiple ways. The reason for neglecting this is that now, $p(h)$ is something the learning algorithm itself can calculate by simply keeping track of the sizes of the choice sets it has encountered so far. It is important to notice that this construction is defined before observing any data. Consequently, every hypothesis has some bound associated with it before the data is used to pick a particular hypothesis and its corresponding bound.

Another way to view this process is that we cannot know in advance which choice sequence the algorithm will make. However, a distribution D on labeled examples induces a probability distribution over choice sequences, inducing a probability distribution $q(h)$ over hypotheses. Ideally we would like to use $p(h) = q(h)$ in our bounds as noted above. However, we cannot calculate $q(h)$ (since the distribution D is unknown), so instead, our choice of $p(h)$ will be just an estimate. We hope that the algorithm designer has chosen a "good" leaning algorithm which induces a distribution $p(h)$ over the final hypotheses which is near to $q(h)$. Our estimate $p(h)$ is the probability distribution resulting from picking each choice uniformly at random from the current choice set at each level (note: this is different from picking a final hypothesis uniformly at random). I.e., it can be viewed as the measure associated with the assumption that at each step, all choices are equally likely.

We immediately find the following theorem:

THEOREM 5.2.2. *(Microchoice Bound) For all hypothesis spaces, H, for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists h \in H: \; e(h) > \bar{e}(m, \hat{e}(h), \frac{\delta}{\prod_{i=1}^{d(h)} |C_i(h)|})\right) \leq \delta$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \max_p\{p: \; Bin(m,k,p) = \delta\}$

Proof. Specialization of the Occam's Razor bound (4.6.1).

Once again, it will be worthwhile to slightly loosen this bound with the following corollary:

COROLLARY 5.2.3. *(Relative Entropy Microchoice Bound) For all hypothesis spaces, H, for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists h \in H: \; KL(\hat{e}(h)||e(h)) > \frac{\sum_{i=1}^{d(h)} \ln |C_i(h)| + \ln \frac{1}{\delta}}{m}\right) \leq \delta$$

The point of the microchoice bound is that the quantity $\bar{e}(...)$ is something the algorithm can calculate as it goes along, based on the sizes of the choice sets encountered. To see this, note that the hypothesis dependent term is $\sum_{i=1}^{d(h)} \ln |C_i(h)|$. The quantity $d(h)$ can be calculated by just noting the number of choices made before the learning algorithm terminates. The choice sets, $C_i(h)$, can often be easily deduced by reasoning about the possible microchoices the algorithm could have made given different datasets.

In many natural cases, a "fortuitous distribution and target concept" corresponds to a shallow leaf or a part of the tree with low branching, resulting in a better bound. For instance, in the decision list case, $\sum_{i=1}^{d(h)} \ln |C_i(h)|$ is roughly $d \ln n$ where d is the length of the list produced and n is the number of features. Notice that $d \ln n$ is also the description length of the final hypothesis produced in the natural encoding, thus in this case these theorems yield similar bounds to a simple application of Occam's razor or SRM.

More generally, the microchoice bound is similar to Occam's razor or SRM bounds when each k-ary choice in the tree corresponds to $\log k$ bits in the natural encoding of the final hypothesis h. However, sometimes this may not be the case. Consider, for instance, a local optimization algorithm in which there are n parameters and each step adds or subtracts 1 from one of the parameters. Suppose in addition the algorithm knows certain constraints that these parameters must satisfy (perhaps a set of linear inequalities) and the algorithm restricts itself to choices in the legal region. In this case, the branching factor, at most $2n$, might become much smaller if we are "lucky" and head toward a highly constrained portion of the solution space. One could always reverse-engineer an encoding of hypotheses based on the choice tree, but the microchoice approach is much more natural.

There is also an opportunity to use *a priori* knowledge in the choice of $p(h)$. In particular, instead of splitting our confidence equally at each node of the tree, we could split it unevenly, according to some heuristic function g. If g is "good" it may produce error bounds similar to the bounds when $p(h) = q(h)$. In fact, the method of section (5.3) where we combine these results with Freund's query-tree [**17**] approach can be thought of as an attempt to do exactly this.

5.2.1. Examples. It is difficult to create a bound which is universally better than previous bounds. The microchoice bound can be much better than the discrete hypothesis bound (4.2.1) and can be slightly worse. To develop some understanding of how they compare we consider several cases.

5.2.1.1. *Greedy Set Cover.* Consider a greedy set cover algorithm for learning an OR function over F Boolean features. The algorithm begins with a choice space of size $F+1$ (one per feature or halt) and chooses the feature that covers the most positive examples while covering no negative ones. It then moves to a choice space of size F (one per feature remaining or halt) and chooses the best remaining feature and so on until it halts. If the number of features chosen is k then the microchoice bound is:

$$\epsilon(h) = \frac{1}{m}\left(\ln\frac{1}{\delta} + \sum_{i=1}^{k}\ln(F-i+2)\right) \leq \frac{1}{m}\left(\ln\frac{1}{\delta} + k\ln(F+1)\right)$$

The bound of (4.2.1) is:

$$\epsilon = \frac{1}{m}\left(\ln\frac{1}{\delta} + F\ln 2\right).$$

If k is small, then the microchoice bound is a lot better, but if $k = O(F)$ then the microchoice bound is slightly worse than the discrete hypothesis bound. Notice that in this case the microchoice bound is essentially the same as the standard Occam's razor analysis when one uses $O(\ln F)$ bits per feature to describe the hypothesis.

5.2.1.2. *Decision Trees.* Decision trees over discrete sets (say, $\{0,1\}^F$) are another natural setting for application of the microchoice bound.

A decision tree differs from a decision list in that the size of the available choice set is larger due to the fact that there are multiple nodes where a new test may be applied. In particular, for a decision tree with K leaves at an average depth of d, the choice set size is $K(F-d)$, giving a bound noticeably worse than the bound for the decision list. This motivates a slightly different decision algorithm which considers only one leaf node at a time. The algorithm adds a new test or decides to never add a new test at this node. In this case, there are $(F-d(v)+1)$ choices for a node v at depth $d(v)$, implying the bound:

$$\mathrm{KL}(\hat{e}(h)||e(h)) \leq \frac{1}{m}\left(\ln\frac{1}{\delta} + \sum_{v}\ln(F-d(v)+1)\right) \tag{5.2.1}$$

where v ranges over the nodes of the decision tree. Once again, this is very similar to what might be produced by an Occam's Razor Bound with an appropriate choice of prior. This result is again sometimes much better than the Discrete Hypothesis bound and sometimes slightly worse.

5.2.2. Pruning. Decision tree algorithms for real-world learning problems often have some form of "pruning" as in [**44**] and [**41**]. The tree is first grown to full size producing a hypothesis with minimum empirical error. Then the tree is "pruned" starting at the leaves and progressing up through the tree toward the root node using some test for the significance of an internal node. An internal node is not significant if the reduction in total error is small in comparison to the complexity of its children. Insignificant internal nodes are replaced with a leaf resulting in a smaller tree.

Microchoice bounds have the property that they incidentally prove a bound for every decision tree which can be found by pruning internal nodes. In particular, one of the choices available when constructing a node is to make the node a leaf. Therefore, if we begin with the tree T and then prune to the smaller tree T', we can apply the bound (5.2.1) to T' *as if* the algorithm had constructed T' directly rather than having gone first through the tree T. This suggests another possible pruning criterion: prune a node if the pruning would result in an improved microchoice bound. That is, prune if the increase in empirical error is less than the decrease in $\epsilon(h)$. This pruning criteria is a "pessimistic criteria" [**38**].

The similarities to SRM are discussed next.

5.2.3. Microchoice and Structural Risk Minimization. The microchoice bound is essentially a compelling application of the Disjoint SRM bound 4.5.1 where the description language for a hypothesis is the sequence of data-dependent choices which the algorithm makes in the process of deciding upon the hypothesis. The hypothesis set H_i is all hypotheses with the same description length in this language.

As an example, consider a binary decision tree with F Boolean features and a Boolean label. The first hypothesis set, H_1 will consist of 2 hypotheses; always false and always true. In general, we will have one hypothesis set for every legal configuration of internal nodes. The size of a hypothesis set where every tree contains k internal nodes will be 2^{k+1} because there are $k+1$ leaves each of which can take 2 values. The weighting $p(i)$ across the different hypothesis sets is defined by the microchoice allocation of confidence.

The principle disadvantage of the microchoice bound is that the sequence of data-dependent choices may contain redundancy. A different SRM bound with a different set of disjoint hypothesis sets might be able to better avoid redundancy. As an example, assume that we are working with a decision tree on F binary features. There are $F+2$ choices (any of F features or 2 labels) at the top node. At the next node down there will be $F+1$ choices in both the left and right children. Repeat until a maximal decision tree is constructed. There will be $\prod_{i=0}^{F}(F-i+2)^{2^i}$ possible trees. This number is somewhat larger than the number of Boolean functions on F features: 2^{2^F}.

5.3. Combining Microchoice with Freund's Query Tree approach

The next section is devoted to an improvement of the microchoice bound called adaptive microchoice, which arises from synthesizing Freund's query trees [**17**] with the microchoice bound. This improvement is not easily expressed as a simplification of Structural Risk Minimization. In essence, the adaptive microchoice bound can gain from dependence on the learning problem distribution D and can take advantage of an "easy" distribution.

First we require some background material in order to state and understand Freund's bound.

5.3.1. Preliminaries and Definitions.

The statistical query framework introduced by Kearns [**26**] restricts learning algorithm to only access the data using statistical queries. A statistical query takes as input a binary predicate, χ, mapping examples to a binary output: $(X, Y) \to \{0, 1\}$. The output of the statistical query is the average of χ over the examples seen. Let z_i be the ith labeled example, then:

$$\hat{D}_\chi = \frac{1}{m}\sum_{i=1}^{m}\chi(z_i)$$

The output is an empirical estimate of the true value $D_\chi = \mathbf{E}_D[\chi(z)] = \Pr_{z\sim D}(\chi(z) = 1)$ of the query under the distribution D[2] . One simple example of a predicate χ is "the first bit is 1". A more complicated predicate might be "the third bit xor the 4th bit is 0". Naturally, the distribution of $\hat{D}_\chi$ will be the familiar Binomial distribution.

It is convenient to define

$$\bar{I}_{D_\chi}(\delta) = \frac{1}{m}\max_k\left\{k:\ 1 - \mathrm{Bin}\left(m, k, D_\chi\right) \geq \frac{\delta}{2}\right\}$$

and

$$\underline{I}_{D_\chi}(\delta) = \frac{1}{m}\min_k\left\{k:\ \mathrm{Bin}\left(m, k, D_\chi\right) \geq \frac{\delta}{2}\right\}$$

and let

$$I_\chi(\delta) = \left[\underline{I}_{D_\chi}(\delta), \bar{I}_{D_\chi}(\delta)\right]$$

Intuitively, $I_\chi(\delta)$ is a (fixed) interval in which the random variable $\hat{D}_\chi$ will fall with high probability. In other words, we know that:

$$\Pr\left[\hat{D}_\chi \notin I_\chi(\delta)\right] \leq \delta$$

Now, we want to construct a confidence interval based upon the high confidence interval $I_\chi(\delta)$. We can do this using the inversion lemma (3.4.1) to get:

$$\bar{D}_\chi(\delta) = \max_p\left\{p : \underline{I}_p(\delta) = \hat{D}_\chi\right\}$$

and

$$\underline{D}_\chi(\delta) = \min_p\left\{p : \bar{I}_p(\delta) = \hat{D}_\chi\right\}$$

The random interval defined here contains the "real" answer D_χ with high probability. In other words, we have:

$$\Pr\left[D_\chi \notin [\underline{D}_\chi(\delta), \bar{D}_\chi(\delta)]\right] \leq \delta$$

[2] In the real SQ model there is no set of examples. The algorithm asks a query χ and is given a response $\hat{D}_\chi$ that is guaranteed to be near to the true value D_χ. That is, the true SQ model is an abstraction of the scenario described here where $\hat{D}_\chi$ is computed from an observed sample.

5.3.2. Background and Summary. Freund [**17**] considers choice algorithms that at each step perform a Statistical Query on the sample, using the result to determine which choice to take. For an algorithm A, tolerance α (defined next), and distribution D, Freund defines the query tree $T_A(D, \alpha)$ as the choice tree created by considering only those choices resulting from answers $\hat{D}_\chi$ to queries χ such that $\left|\hat{D}_\chi - D_\chi\right| \leq \alpha$. The idea is that if a particular predicate, χ, is true with probability .9 (for example) on a random sample it is very unlikely that the empirical result of the query will be .1. More generally, the chance the answer to a given query is off by more than α is at most $2e^{-2m\alpha^2}$ by Hoeffding's inequality. So, if the entire tree contains a total of $|Q(T_A(D, \alpha))|$ queries in it, the probability *any* of these queries is off by more than α is at most $2 \cdot |Q(T_A(D, \alpha))| \cdot e^{-2m\alpha^2}$. In other words, this is an upper bound on the probability the algorithm ever "falls off the tree" and makes a low probability choice. The point of this is that we can allocate half (say) of the confidence parameter δ to the event that the algorithm ever falls off the tree, and then spread the remaining half evenly on the hypotheses in the tree (which hopefully is a much smaller set than the entire hypothesis set).

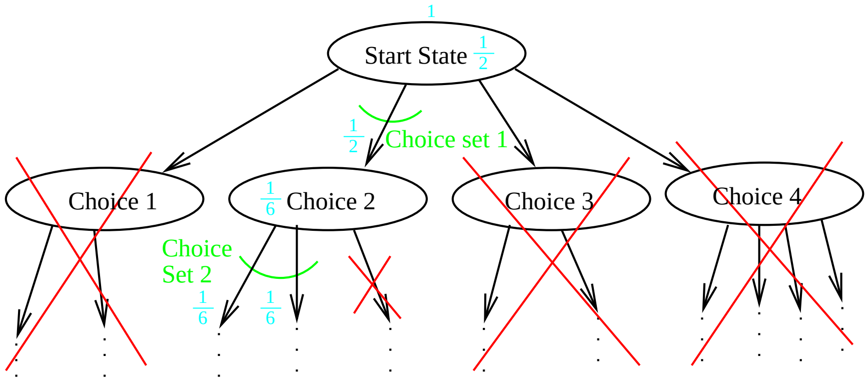
Unfortunately, the query tree suffers from the same problem as the $q(h)$ distribution considered in section (5.1), namely that to compute it, one needs to know D. So, Freund proposes an algorithmic method to find a super-set approximation of the tree. The idea is that by analyzing the results of queries, it is possible to determine which outcomes were unlikely given that the query is close to the desired outcome. In particular, each time a query χ is asked and a response $\hat{D}_\chi$ is received, if it is true that $|\hat{D}_\chi - D_\chi| \leq \alpha$, then the range $\left[\hat{D}_\chi - 2\alpha, \hat{D}_\chi + 2\alpha\right]$ contains the range $[D_\chi - \alpha, D_\chi + \alpha]$. Thus, under the assumption that no query in the *correct* tree is answered badly, a super-set of the correct tree can be produced by exploring all choices resulting from responses within 2α of the response actually received. By applying this method to every node in the query tree we can generate an empirically observable super-set of the query tree: that is, the original query tree is a pruning of the empirically constructed tree.

A drawback of this method is that it can easily take exponential time to produce the approximate tree, because even the smaller correct tree can have a size exponential in the running time of the learning algorithm. Instead, we would much rather simply keep track of the choices actually made and the sizes of the nodes actually followed, which is what the microchoice approach allows us to do. As a secondary point, given δ, computing a good value of α for Freund's approach is not trivial, see [**17**]; we will be able to finesse that issue and use the tighter bound $\hat{D}_\chi \in I_\chi(\delta)$.

In order to apply the microchoice approach, we modify Freund's query tree so that different nodes in the tree receive different confidence, δ, much in the same way that different hypotheses h in our choice tree receive different values of $\delta(h)$.

5.3.3. Microchoice Bounds for Query Trees. The manipulations of the choice tree are now reasonably straightforward. We begin by describing the *true* microchoice query tree and then give the algorithmic approximation. As with the choice tree in section (5.2), one should think of each node in the tree as representing the current internal state of the algorithm.

We incorporate Freund's approach into the choice tree construction by having each internal node allocate a portion, p' of its "supply" of failure probability to the event that $\hat{D}_\chi \notin I_\chi(\delta * p')$. The node then splits the remainder of its supply evenly among the children corresponding to choices that result from answers $\hat{D}_\chi$ with $\hat{D}_\chi \in I_\chi(\delta * p')$. Choices that would result from "bad" answers to the query are *pruned away* from the tree and get nothing. This continues down the tree to the leaves. Pictorially, this looks like:



How should p' be chosen? Smaller values of p' result in larger intervals $I_\chi(\delta * p')$ leading to more children in the pruned tree and less confidence given to each. Larger values of p' result in less left over to divide among the children. Unfortunately, our algorithmic approximation (which only sees the empirical answers $\hat{D}_\chi$ and needs to be efficient) will not be able to make this optimization. Therefore, we define p' in the *true* microchoice query tree to be $\frac{p}{d+1}$ where d is the depth of the current node. This choice will imply that the adaptive microchoice bound is never much worse than the Microchoice bound, and sometimes much better.

Since a particular query value $\hat{D}_\chi$ implies a particular choice c, we can think of the interval $I_\chi(\delta)$ as containing *choices* rather than query results. After all, we only care about the choices the algorithm makes. We can calculate the probability assigned to a hypothesis in the *true* adaptive microchoice query tree according to the following algorithm:

Algorithm 5.3.1. *True_ Adaptive_ Microchoice(δ)*

```
(1) set $p \leftarrow 1$
(2) set $d = 1$
(3) while learning algorithm has not halted.
    (a) $d \leftarrow d + 1$
    (b) $p' \leftarrow \frac{p}{d}$
    (c) Let $C$ = the current set of possible data-dependent choices.
    (d) $|\hat{C}| \leftarrow |\{c \in C | c \in I_\chi(\delta * p')\}|$
    (e) $p \leftarrow \frac{p-p'}{|\hat{C}|}$
(4) return $p$
```

There are two important things to note about this algorithm. First of all, we could plug the value $p(h)$ it returns into the Occam's Razor Bound 4.6.1 and receive a bound on the true error rate of our chosen classifier.

Second, this algorithm can not be executed. The essential problem is determining whether or not $c \in I_\chi(\delta * p')$ which cannot be done without knowledge of the underlying distribution D. However, we *can* calculate an approximate version of this algorithm which, with high probability, returns a value p which is smaller. Since a smaller value is pessimistic, we can use it in our bounds.

The algorithmic approximation uses the idea in [**17**] of including all choices within the *double* confidence interval of the observed value $\hat{D}_\chi$. Unlike [**17**], however, we do not actually create the tree; instead we just follow the path taken by the learning algorithm, and argue that the "supply" probability remaining at the leaf is no greater than the amount that would have been there in the original tree. Finally, the algorithm outputs a bound calculated with $p(h)$.

Specifically, the algorithm is as follows. Suppose we are at a node of the tree containing statistical query χ at depth $d(\chi)$ and we have a p supply of parameter. (If the current node is the root, then $p = 1$ and $d(\chi) = 1$). We choose $p' = p/(d+1)$, ask the query χ, and receive $\hat{D}_\chi$. Let

$$\bar{\bar{D}}_\chi(\delta) = \min_k \left\{ \frac{k}{m} :\ 1 - \mathrm{Bin}\left(m, k, \bar{D}_\chi(\delta)\right) \le \frac{\delta}{2} \right\}$$

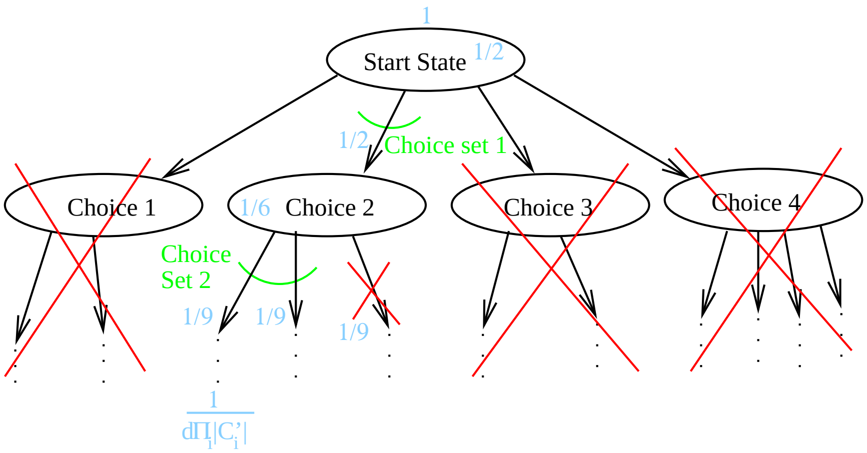
and

$$\underline{\underline{D}}_\chi(\delta) = \max_k \left\{ \frac{k}{m} :\ \mathrm{Bin}\left(m, k, \bar{D}_\chi(\delta)\right) \le \frac{\delta}{2} \right\}$$

with

$$\hat{I}_\chi(\delta) = \left[\underline{\underline{D}}_\chi(\delta), \bar{\bar{D}}_\chi(\delta)\right]$$

We now let k be the number of children of our node corresponding to answers in the range $\hat{I}_\chi(p')$. We then go to the child corresponding to the answer $\hat{D}_\chi$ that we received, giving this child a confidence parameter supply of $(p - p')/k$. This is the same as we would have given it had we allocated $p - p'$ to the children equally. We then continue from that child. Finally, when we reach a leaf, we output the probability left for the hypothesis. Pictorially, this looks like:



Notice that the second choice set is larger than in the true adaptive microchoice set tree. This can easily happen and it makes our results somewhat more pessimistic. The approximate adaptive microchoice algorithm is specified as follows:

ALGORITHM 5.3.2. *Approximate_ Adaptive_ Microchoice(δ)*

```
(1) set $p \leftarrow 1$
(2) set $d = 1$
(3) while learning algorithm has not halted.
    (a) $d \leftarrow d + 1$
    (b) $p' \leftarrow \frac{p}{d}$
    (c) Let $C$ = the current set of possible data-dependent choices.
    (d) $|\hat{C}| \leftarrow |\{c \in C | c \in \hat{I}_\chi(\delta * p')\}|$
    (e) $p \leftarrow \frac{p-p'}{|\hat{C}|}$
(4) return $p$
```

Let $d(h)$ be the depth of some hypothesis h in the empirical path and $\hat{C}_1(h)$, $\hat{C}_2(h)$, ..., $\hat{C}_d(h)$ be the sequence of choice sets resulting in h in the algorithmic construction; i.e., $\hat{C}_i(h)$ is the number of unpruned children of the i-th node. Then, the confidence placed on h will be:

$$p(h) = \prod_{i=1}^{d(h)} \left(\frac{i}{i+1} \frac{1}{|\hat{C}_i(h)|} \right) = \frac{1}{d(h)+1} \prod_{i=1}^{d(h)} \frac{1}{|\hat{C}_i(h)|} \tag{5.3.1}$$

THEOREM 5.3.3. *(Adaptive Microchoice Bound) For all hypothesis spaces, H, for all $\delta \in (0,1]$:*

$$\Pr_{D^m} \left(\exists h \in H : \; e(h) > \bar{e}(m, \hat{e}(h), p(h) * \delta) \right) \leq \delta$$

where $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \max_p\{p : \; Bin(m,k,p) = \delta\}$, and $p(h)$ is as defined in equation 5.3.1.

PROOF. By design, with probability $1 - \delta$ all queries in the true microchoice query tree receive good answers, *and* all hypotheses in that tree have their true errors within their estimates.
We will prove that in the high probability case, the output of the Approximate_Adaptive_Microchoice algorithm is less than the output of the True_Adaptive_Microchoice algorithm. Since a smaller $p(h)$ makes the bound more pessimistic, we will prove the bound. Assume inductively that at the current node of our empirical path the supply p_{emp} is no greater than the supply p_{true} given to that node in the true tree. This is clearly satisfied in the base case when $p_{emp} = p_{true} = 1$.
Under the assumption that the response $\hat{D}_\chi$ falls in the interval $I_\chi(p_{\text{true}}/(d+1))$, it must be the case that the interval $\hat{I}_\chi(p_{\text{emp}}/(d+1))$ contains the interval $I_\chi(p_{\text{true}}/(d+1))$. Therefore, the supply given to any child in the empirical path is no greater than the supply given in the true tree. □

The corresponding relative entropy corollary is:

COROLLARY 5.3.4. *(Relative Entropy Microchoice Bound) For all hypothesis spaces, H, for all $\delta \in (0,1]$,*

$$\Pr_{D^m} \left(\exists h \in H : \; KL(\hat{e}(h) || e(h)) > \frac{\sum_{i=1}^{d(h)} \ln |\hat{C}_i(h)| + \ln(d(h)+1) + \ln \frac{1}{\delta}}{m} \right) \leq \delta$$

The bound in theorem (5.3.3) is very similar to (5.2.2) except that the choice complexity is slightly worsened with the $\ln(d(h)+1)$ term but improved by replacing $C_i(h)$ with the smaller $\hat{C}_i(h)$.

5.3.4. Allowing batch queries. Most natural Statistical Query algorithms make each choice based on responses to a *set* of queries, not just one. For instance, to decide what variable to put at the top of a decision tree, we ask F queries, one for each feature; we then choose the feature whose answer was most "interesting". This suggests generalizing the query tree model to allow each tree node to contain a set of queries, executed in batch. Requiring each node in the query tree to contain just a single query as in the above construction would result in an unfortunately high branching factor just for the purpose of "remembering" the answers received so far. [3]

Extending the algorithmic construction to allow for batch queries is easily done. If a node has q queries $\chi_1, \ldots, \chi_q$, we choose the query confidence $\delta * p'$ as before, but we now split the mass evenly among all q queries. We then let k be the number of children corresponding to answers to the queries $\chi_1, \ldots, \chi_q$ in the ranges $\hat{I}_{\chi_1}(\delta p'/q), \ldots, \hat{I}_{\chi_q}(\delta p'/q)$ respectively. We then go to the child corresponding to the answers we actually received, and as before give the child a probability supply of $(p - p')/k$. Theorem (5.3.3) holds exactly as before; the only change is that $|\hat{C}_i(h)|$ means the size of the i-th choice set in the batch tree rather than the size in the single-query-per-node tree.

5.3.5. Example: Batch Queries for Decision trees. When growing a decision tree, it is natural to make a batch of queries and then make a decision about which feature to place in a node. The process is then repeated to grow the full tree structure. As in the decision tree example described in the simple microchoice section, if we have F features and are considering adding a node at depth $d(v)$, there are $F - d(v) + 1$ possible features that could be chosen for placement in a particular node. The decision of which feature to use is made by comparing the results of $F - d(v) + 1$ queries to pick the best feature according to some criteria, such as information gain. We can choose $p' = p/(d + 1)$, then further divide p' into confidences of size $p'/(F - d(v) + 1)$, placing each divided confidence on one of the $F - d(v) + 1$ statistical queries. We now may be able to eliminate some of the $F - d(v) + 1$ choices from consideration, allowing the remaining confidence, $p - p'$ to be apportioned evenly amongst the remaining choices. Depending on the underlying distribution this could substantially reduce the size of the choice set. The best case occurs when one feature partitions all examples reaching the node perfectly and all other features are independent of the target. In this case the choice set will have size 1 if there are enough examples.

[3] Consider a decision tree algorithm attempting to find the right feature for a node. If the first query returns a value of $\hat{D}_\chi$ with a confidence of δ then the branching factor would be approximately $m \cdot \hat{I}_\chi(\delta)$. This branching factor would be approximately the same for further queries required by the algorithm to make a decision about what feature to use. This results in a total multiplied choice space size of approximately $\left[4m \cdot \hat{I}_\chi(\delta)\right]^F$ which can be reduced to F or less using a batch query.

5.3.6. Adaptive Microchoice vs. Basic Microchoice. The adaptive microchoice bound is a significant improvement over the simple microchoice bound when the distribution is such that each choice is clear. For example, consider F Boolean features and $N = O(F)$ examples. Suppose that one feature is identical to the label and all the rest of the features are determined with a coin flip independent of the label.

When we apply a decision tree to a data set generated with this distribution, what will be the resulting bound? Given enough examples, with high probability there will only be one significant choice for the first batch query: the feature identical to the label. The second and third batch queries, corresponding to the children of the root feature, will also have a choice space of size 1 with very high probability. The "right" choice will be the label value. Each choice set has size 1 resulting in a complexity of $\ln 4$ due to allocation of confidence to the statistical queries necessary for learning the decision tree. $\ln 4$ is considerably better than $\ln(F+2) + 2\ln(F+1)$ which the simple version of the microchoice bound provides. Note that the complexity reduction only occurs with a large enough number of examples m implying that the value of $\bar{e}$ calculated can improve faster than (inverse) linearly in the number of examples.

The adaptive microchoice bound is never much looser than the simple microchoice bound because under the assumption that choice sets are of size at least 2, the penalty for using the adaptive version, $\ln d$, is always small compared to the complexity term for the simple microchoice bound, $\sum_{i=1}^{d(h)} \ln |C_i(h)|$.

5.3.7. Other Adaptive Microchoice. The adaptive microchoice bound provides a simple scheme for dividing confidence between choices and queries. There are other choices which may be useful in some settings. Any scheme which *a priori* divides the confidence between queries and choices at every node will generally work. Here are two schemes which may be useful:

- Assign a constant proportion of confidence to the query. This scheme is more aggressive than the one used in the adaptive microchoice bounds and may result in a lower complexity when many choices are eliminatable. The drawback is we no longer get the telescoping in equation (5.3.1) and so the term logarithmic in $d(h)$ in theorem (5.3.4) becomes linear in $d(h)$.
- For a decision tree, assign a portion dependent on the depth of the node in the decision tree that the choice set is over. It is unlikely that choices are eliminatable from nodes not near the root because the number of examples available at a node typically decays exponentially with the (decision tree) depth. A progressive scheme which allocates less confidence to queries for deep nodes will probably behave better in practice.

5.3.8. Comparison with Freund's Self-Bounding algorithms. Freund's approach for self-bounding learning algorithms can require exponentially more computation then the microchoice approach. In its basic form, it requires explicit construction of every path in the state space of the algorithm not pruned in the tree. There exist some learning algorithms where this process can be done implicitly making the computation feasible. However, in general this does not appear to be possible.

The adaptive microchoice bound only requires explicit construction of the size of each subset from which a choice is made. Because many common learning algorithms work by a process of making choices from small subsets, this is often computationally easy. The adaptive microchoice bound does poorly, however, when Freund's query tree has a high degree of sharing; for example, when many nodes of the tree correspond to the same query, or many leaves of the tree have the same final hypothesis. Allowing batch queries alleviates the most egregious examples of this. It is also possible to interpolate between the adaptive microchoice bound and Freund's bound by a process of conglomerating the subsets of the microchoice bound.

5.3.9. Choice Set Conglomeration. The mechanism of choice set conglomeration is a similar to the batch query technique. It allows you to trade increased computation for a tighter bound. When starting with the simple microchoice bound, this technique can smoothly interpolate with the discrete hypothesis bound (4.2.1). When starting with the adaptive microchoice bound, we can interpolate with Freund's bound.

Consider a particular choice set, $\hat{C}_i$, with elements c_i. Each c_i indexes another choice set, $\hat{C}_{i+1}(c_i)$. If the computational resources exist to calculate the union $\hat{C}_{i,i+1} = \bigcup_{c_i \in \hat{C}_i} \hat{C}_{i+1}(c_i)$, then $|\hat{C}_{i,i+1}|$ can be used in place of $|\hat{C}_i| \cdot |\hat{C}_{i+1}|$ in the adaptive microchoice bound. The conglomeration can be done repeatedly to build large choice sets and also applies to the simple microchoice bound (5.2.2). Conglomeration can be useful for tightening the bound when there are multiple choice sequences leading to the same hypothesis. However, choice set conglomeration is not always helpful because it trades away the fine granularity of the microchoice bound. The extreme case where all choice sets are conglomerated into one choice set and every hypothesis and query have the same weight is equivalent to Freund's bound.

When the choices of the attached choice sets are all different, conglomeration will have little use because the size of the union of the choice sets is the sum of the sizes of each choice set $|\hat{C}_{i,i+1}| = \sum_{c_i \in \hat{C}_i} |\hat{C}_{i+1}(c_i)|$. If the child sets each have the same size $|\hat{C}_{i+1}|$ then this simplifies to $|\hat{C}_{i,i+1}| = |\hat{C}_i| \cdot |\hat{C}_{i+1}|$ which results in the same confidence applied to each choice whether conglomerating or not. The best case for conglomeration is equivalent to the batch query case: every sub-choice set contains the same elements. Then we have $|\hat{C}_{i,i+1}| = |\hat{C}_{i+1}|$ and can pay no cost for the choice set $|\hat{C}_i|$.

5.4. Microchoice discussion

Microchoice bounds can be used in practice and do yield results comparable with holdout set based techniques (see figure .12.3.5 and the surrounding section for details). There are two significant insights which the microchoice bounds provide.

1. The "cost" of choices is made very explicit. The cost of a choice (in terms of sample complexity) is the log of the number of choices.
2. It is possible to improve upon the Occam's Razor Bound (theorem 4.6.1) by using information from the sample set to infer properties (such as the choice tree) of the distribution D. This can be done *without* any explicit knowledge of the distribution D.

PROBLEM 5.4.1. (Open) Is there a satisfying, natural bound for the continuous case? Preliminary work and thoughts by several people has occurred, but nothing has yet come of it.

CHAPTER 6

PAC-Bayes bounds

The work presented here is also published in [**35**].

PAC-Bayes bounds are a generalization of the Occam's razor bound for algorithms which output a *distribution* over classifiers rather than just a single classifier. This includes the possibility of a distribution over a single classifier, so it is a generalization. Most classifiers do not output a distribution over base classifiers. Instead, they output either a classifier, or an average over base classifiers. Nonetheless, PAC-Bayes bounds are interesting for several reasons:

(1) PAC-Bayes bounds are much tighter (in practice) than most common VC-related [**51**] approaches on continuous classifier spaces. This can be shown by application to stochastic neural networks (see section 13) as well as other classifiers. It also can be seen by observation: when specializing the PAC-Bayes bounds on discrete hypothesis spaces, only $O(\ln m)$ sample complexity is lost.
(2) Due to the achievable tightness, the result motivates new learning algorithms which strongly limit the amount of overfitting that a learning algorithm will incur.
(3) The result found here will turn out to be useful for averaging hypotheses.

PAC-Bayes bounds were first introduced by McAllester [**39**].

There are three relatively independent observations in this chapter:

(1) A quantitative improvement of the PAC-Bayes by retrofit with relative entropy Chernoff bound 3.2.1. This retrofit is not as trivial as might be expected, but it can be done. The result is the tightest known PAC-Bayes bound. In addition to the quantitative improvements, this tightening simplifies the proof and adds to our qualitative understanding of the bound.
(2) A method for (partially) derandomizing the PAC-Bayes stochastic hypothesis
(3) A method for stochastic evaluation of the empirical error.

The first observation is the most important. Observation (3) is important for many practical applications because it is safely avoids a (sometimes) very complicated evaluation problem. Observation (2) is of little theoretical interest, but it might interest some people who feel reassured when every classifier randomized over has a low empirical error rate.

Figure 6.0.1 shows what the PAC-Bayes bound looks like as an interactive proof of learning.

6.1. PAC-Bayes Basics

In the PAC-Bayes setting, a classifier is defined by a distribution $q(h)$ over the hypothesis space. Each classification is carried out according to a hypothesis

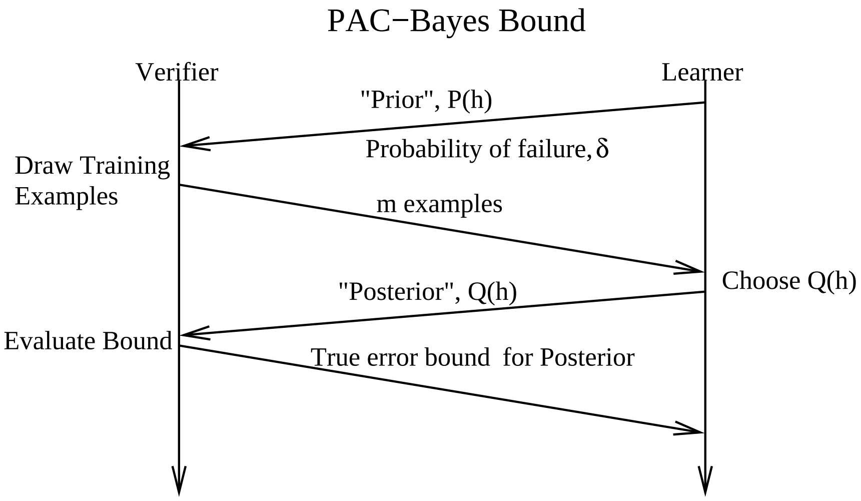


FIGURE 6.0.1. The PAC-Bayes bound can be viewed as a new style for a proof of learning. The learner must commit to a "Prior" as in the Occam's Razor Bound 4.6.1 before seeing examples, but it does not commit to a single hypothesis. Instead, it commits to a distribution over hypotheses, $q(h)$ and the bound applies to a randomization with respect to the distribution $q(h)$.

sampled from $q(h)$. We are interested in the gap between the *expected* generalization error $e_q(h) \equiv E_q[e(h)]$ and the *expected* empirical error $\hat{e}_q(h) = E_q[\hat{e}(h)]$, where both expectations are taken with respect to $q(h)$. The gap will be parameterized by the Kullback-Leibler divergence (see [**10**]). Recall that:

$$\text{KL}(q||p) = E_{h\sim q(h)} \ln \frac{q(h)}{p(h)} \tag{6.1.1}$$

If the support is finite, we have

$$\text{KL}(q||p) = \sum_h q(h) \ln \frac{q(h)}{p(h)} \tag{6.1.2}$$

The relative entropy is an asymmetric distance measure between probability distributions, with $\text{KL}(q||p) = 0 \Leftrightarrow q = p$ almost everywhere.

THEOREM 6.1.1. *(PAC-Bayes* [**39**]*) For all "priors" $p(h)$ and for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists q(h) : \ e_q(h) \geq \hat{e}_q(h) + \sqrt{\frac{KL(q||p) + \ln\frac{m}{\delta} + 2}{2m-1}}\right) \leq \delta$$

PROOF. Given in [**39**]. □

This PAC-Bayes bound is almost the same as the Occam's Razor bound (theorem 4.6.1) when the distribution is peaked on a single hypothesis and the Occam's razor bound is proved using the looser Hoeffding inequality. This can be seen by noting that the KL-divergence when q is all on one hypothesis, h satisfies

$\mathrm{KL}(q||p) = \log \frac{1}{p(h)}$. Comparing with the Occam's Razor bound, we see that a (small) extra term of size $\frac{\ln m}{m}$ has been introduced in return for the capability to average with respect to *any* posterior $q(h)$. It is unclear yet that this $\frac{\ln m}{m}$ term needs to be there.

PROBLEM 6.1.2. (Open) Remove the $\frac{\ln m}{m}$ term from the sample complexity.

The real power of the PAC-Bayes bound occurs when the average is over many hypotheses. This might occur if the distribution $q(h)$ is picked using Bayes law or a Gibbs distribution. One of the most interesting aspects of the PAC-Bayes bound is that it holds for finite *and* infinite hypothesis spaces.

6.2. A Tighter PAC-Bayes Bound

We can tighten this bound by employing a more accurate tail bound on the Binomial distribution. The proof of this improved lemma is not as straightforward as a simple substitution of the Hoeffding bound with the relative entropy Chernoff bound 3.2.1 but it can be worked out nonetheless.

THEOREM 6.2.1. *(Relative Entropy PAC-Bayes bound) For all binary loss functions, $l(h,(x,y))$, for all "priors" $p(h)$ and for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists q(h) : \ KL(\hat{e}_q(h)||e_q(h)) \geq \frac{KL(q||p) + \ln\frac{m}{\delta}}{m-1}\right) \leq \delta$$

This bound is always at least as tight as the original PAC-Bayes bound [**39**] and sometimes much tighter, such as when the average empirical error is near 0. In particular, when the average empirical error is zero ($\hat{e}_q(h) = 0$) the bound can be significantly tighter as shown in figure 3.4.1 on page 23.

One interesting new feature of this PAC-Bayes bound is "dimensionally consistency"[1]. In particular, each side of the equation is an expectation of log probabilities—"nats". Rewriting, we get that with high probability, approximately the following holds:

$$m\mathrm{KL}(\hat{e}_q(h)||e_q(h)) \leq \mathrm{KL}(q||p)$$

There is a coding theory interpretation of KL divergence: $\mathrm{KL}(q||p) =$ the expected number of *extra* bits required to encode symbols drawn from q given a code designed for symbols drawn from p rather than from q.

Using the coding theory interpretation of KL divergence, this says approximately: "With high probability the number of *extra* bits required to encode the empirical errors is less than the number of *extra* bits required to encode hypotheses drawn from the posterior."

The retrofit of the PAC-Bayes bound is accomplished by reproving a technical lemma about distributions. The proof relies upon two lemmas. The first is Lemma 22 from [**39**] which is given by:

LEMMA 6.2.2. *For all $\beta > 0, K > 0$ and $Q, P, y \in R^n$ satisfying $P_i > 0, Q_i > 0$ and $\sum_i Q_i = 1$, if*

$$\sum_{i=1}^{n} P_i e^{\beta y_i} \leq K$$

[1] My thanks to Patrick Haffner for pointing this out.

then

$$\sum_{i=1}^{n} Q_i y_i \leq \frac{KL(Q||P) + \ln K}{\beta}$$

PROOF. given in [**39**]. □

We will need to prove the following lemma in order to tighten the PAC-Bayes bound. It is analogous to Lemma 17 from [**39**].

LEMMA 6.2.3. *For all "priors" $p(h)$, for all $\delta, \alpha \in (0,1)$:*

$$\Pr_{D^m}\left(E_p e^{(1-\alpha)mKL(\hat{e}(h)||e(h))} > \frac{1}{\alpha\delta}\right) \leq \delta$$

PROOF. For any given hypothesis h we will prove the following.

$$\forall h \; E_{D^m} e^{(1-\alpha)m\mathrm{KL}(\hat{e}(h)||e(h))} \leq \frac{1}{\alpha} \tag{6.2.1}$$

The Lemma then follows from the sequence:

$$\Rightarrow \forall p \; E_p E_{D^m} e^{(1-\alpha)m\mathrm{KL}(\hat{e}(h)||e(h))} \leq \frac{1}{\alpha}$$

$$\Rightarrow \forall p \; E_{D^m} E_p e^{(1-\alpha)m\mathrm{KL}(\hat{e}(h)||e(h))} \leq \frac{1}{\alpha}$$

$$\Rightarrow \forall p \; \Pr_{D^m}\left(E_p e^{(1-\alpha)m\mathrm{KL}(\hat{e}(h)||e(h))} \geq \frac{1}{\alpha\delta}\right) \leq \delta$$

Consequently, we must only prove equation 6.2.1. Given the hypothesis, we have a fixed true error rate, $e(h)$, and the empirical error rate $\hat{e}(h)$ will be distributed like a Binomial. Let $R(e(h))$ be the random variable with a cumulative distribution given by the relative entropy Chernoff bound for a hypothesis with true error $e(h)$. In other words, define a cumulative distribution function on $[0,1]$ according to:

$$e^{-m\mathrm{KL}(R||p)}$$

(note that we defined KL() so that it is always 0 when $\frac{k}{m} > p$). Note that the relative entropy Chernoff bound implies R satisfies:

$$\mathrm{Bin}(m, mR, p) \leq e^{-m\mathrm{KL}(R||p)}$$

whenever mR is integer.

Since $e^{(1-\alpha)m\mathrm{KL}(\hat{e}(h)||e(h))}$ increases monotonically with decreasing $\hat{e}(h)$, the probability distribution function of $R(e(h))$ will have a larger expected value. In other words:

$$E_{D^m} e^{(1-\alpha)m\mathrm{KL}(\hat{e}(h)||e(h))} \leq E_R e^{(1-\alpha)m\mathrm{KL}(R||e(h))}$$

The probability distribution function of R is given by:

$$f(x) = \begin{cases} 0 & \text{for } x \geq p \\ -m\frac{\partial \mathrm{KL}(x||p)}{\partial x} e^{-m\mathrm{KL}(x||p)} & \text{for } x \leq p \end{cases}$$

Taking the expectation with respect to this distribution gives us:

$$\begin{aligned} E_R\left[e^{(1-\alpha)m\mathrm{KL}(R||e(h))}\right] &= \int_0^1 e^{(1-\alpha)m\mathrm{KL}(x||e(h))}f(x)dx \\ &= \int_0^{e(h)} -m\frac{\partial\mathrm{KL}(x||e(h))}{\partial x}e^{-\alpha m\mathrm{KL}(x||e(h))}dx \\ &= \frac{1}{\alpha}e^{-\alpha m\mathrm{KL}(x||e(h))}\Big|_0^{e(h)} \\ &\leq \frac{1}{\alpha} \end{aligned}$$

This finishes the proof of the technical lemma. □

Now, we can prove the relative entropy PAC-Bayes bound 6.2.1.

Proof. First, we can specialize lemma 6.2.3 with $\alpha = \frac{1}{m}$ to get that with probability $1-\delta$

$$E_p e^{(m-1)\mathrm{KL}(\hat{e}(h)||e(h))} \leq \frac{m}{\delta}$$

Apply lemma 6.2.2 with $K = \frac{m}{\delta}$, $\beta = m-1$, $Q_i = q(h_i)$ and $y_i = \mathrm{KL}(\hat{e}(h)||e(h))$ to get:

$$E_q\mathrm{KL}(\hat{e}(h)||e(h)) = \sum_{i=1}^{n} Q_i\mathrm{KL}(\hat{e}(h_i)||e(h_i)) \leq \frac{\mathrm{KL}(q||p)+\ln\frac{m}{\delta}}{m-1}$$

Jensen's inequality, gives us:

$$\mathrm{KL}(\hat{e}_q(h)||e_q(h)) \leq E_q\mathrm{KL}(\hat{e}(h)||e(h)) \leq \frac{\mathrm{KL}(q||p)+\ln\frac{m}{\delta}}{m-1}$$

which proves the theorem for the finite case. For the infinite case, a sequence of limits can be defined just as in [**39**]. □

6.3. PAC-Bayes Approximations

6.3.1. Approximating the empirical error. In practice, it is not always easy to calculate some of the observable variables in the PAC-Bayes bound. In particular, $\hat{e}_q(h)$ is not necessarily easy to calculate when q is some continuous distribution. We can avoid the need for a direct evaluation by a Monte Carlo evaluation and a bound on the tail of the Monte Carlo evaluation. Let $\hat{e}_{\hat{q}}(h) \equiv \Pr_{\hat{q},S}(h(x) \neq y)$ be the observed rate of failure of n random hypotheses drawn according to $q(h)$ and applied to a random training example. Once again, we have a familiar Binomial distribution. Direct calculation will give us:

Theorem 6.3.1. *(Monte Carlo Sampling Bound) For all $\delta \in (0,1]$*

$$\Pr_{q^n}\left(\ \hat{e}(h) > \bar{e}\left(n, \hat{e}_{\hat{q}}(h), \delta\right)\right) \leq \delta$$

where $\bar{e}\left(n, \frac{k}{n}, \delta\right) \equiv \max_p\{p:\ Bin(n,k,p)\} \geq \delta$

Proof. Observer that the Monte Carlo estimate is distributed like a Binomial distribution and apply the Binomial Tail bound. □

In order to calculate a bound on the expected true error rate, we can first bound the expected empirical error rate $\hat{e}_q(h)$ with confidence $\frac{\delta}{2}$ then bound the expected true error rate $e_q(h)$ with confidence $\frac{\delta}{2}$, using our bound on $\hat{e}_q(h)$. Since the total probability of failure is only $\frac{\delta}{2} + \frac{\delta}{2} = \delta$ our bound will hold with probability $1 - \delta$.

6.3.2. Derandomizing the PAC-Bayes bound. It is sometimes desirable to derandomize the PAC-Bayes bound. There are several ways to do this. The next chapter will talk about replacing the randomization over $q(h)$ with a thresholded average. Another technique is to simply pick a hypothesis according to $q(h)$. While this would probably be effective in practice, the theoretical guarantees that can be made for this technique are weak. Strong theoretical guarantees *can* be made for a similar technique.

Suppose we make n draws form $q(h)$. Let the drawn hypotheses be $\{h_1, ..., h_n\}$. We can form a new distribution $\hat{q}(h)$ which is uniform over the n draws. The true error rate of this distribution can be bound with high probability according to the following theorem.

THEOREM 6.3.2. *(PAC-Bayes Derandomization) For all* $\delta \in (0, 1]$

$$\Pr_{q^n}(\ e_{\hat{q}}(h) > \bar{e}\,(n, e_q(h), \delta)) \leq \delta$$

where $\bar{e}\left(n, \frac{k}{n}, \delta\right) \equiv \max_p\{p :\ Bin(n, k, p)\} \geq \delta$

PROOF. Observer that the distribution of $e_{\hat{q}}(h)$ is distributed like a Binomial around $e_q(h)$ and apply the Binomial Tail bound. □

Note the this theorem and the last theorem are essentially the same theorem.

This theorem allows us to do an (incomplete) derandomization. Instead of drawing from $q(h)$ in order to evaluate an input, we can draw from $\hat{q}(h)$ which requires a fixed finite number of bits. This may allow for more efficient algorithms, and some people may find it reassuring that every hypothesis in $\{h_1, ..., h_n\}$ has a low empirical error. The same confidence splitting trick of the last section can be used in order to guarantee $e_q(h)$ is bounded and $e_{\hat{q}}(h)$ is bounded given that $e_q(h)$ is bounded.

It is worth mentioning that *no* assumption of independence applies to either this theorem or the last theorem since we explicitly control (and create) the independence ourselves. These theorems hold for totally verifiable preconditions.

6.4. Application of the PAC-Bayes bound

The goal of this chapter is making PAC-Bayes bounds more applicable. This is done by tightening the analysis from a Hoeffding-like to a Chernoff-like statement, and by noting that we can use monte-carlo evaluation to safely bound the stochastic empirical error rate quickly.

In work detailed in chapter 13, results for the application of PAC-Bayes bounds to stochastic neural networks is presented. PAC-Bayes bounds are one of very few approaches capable of producing nonvacuous learning theory bounds on continuous valued classifiers for real-world problems.

CHAPTER 7

Averaging Bounds (Improved margin)

The work in this chapter is joint with Matthias Seeger and Nimrod Megiddo. It was first presented at ICML [**34**].

Averaging bounds are specialized for averaging classifiers. An average has the form

$$f(x) = \int q(h)h(x)dx \text{ or } f(x) = \int_{i=1}^{N} q(h_i)h_i(x)dx$$

where $q(h_i) \geq 0$ and $\int q(h)dh = 1$. Averaging classifiers have the form:

$$c(x) = \text{sign}\,(f(x))$$

Averaging bounds are especially interesting because there are many learning algorithms which use averaging. These techniques include:

(1) Boosting [**18**]
(2) Bayes-Optimal learning (see section 6.7 of [**37**])
(3) Support Vector Machines [**8**]
(4) Bagging [**6**]
(5) Maximum Entropy classification [**24**]

Viewed as an interactive proof of learning (see figure 7.0.1) the bound presented here is almost the same as the PAC-Bayes bound except that it applies to the *average* over the posterior rather than to stochastic choices over the posterior.

The bound in this section is a qualitative improvement on prior results for averaging bounds. For the average learning algorithms listed above, the form of the improvements is most interesting for Maximum Entropy Classification and Bayes-Optimal classification. All (currently known) specialized averaging bounds use as a parameter the "margin". For this section only, suppose the label and the hypotheses have value -1 or 1. ($y \in \{-1, 1\}$ and $h(x) \in \{-1, 1\}$). Then, the margin will be defined as

$$t(x, y) = yf(x)$$

Some simple observations are immediate:

(1) The margin is bounded. $t(x, y) \in [-1, 1]$
(2) If the classifier is correct, the margin is positive. $c(x) = y \Rightarrow t(x, y) \in (0, 1]$

The *error* at some margin is the quantity actually used in the averaging bounds. The empirical error at margin θ is defined by:

$$\hat{e}_\theta(c) = \Pr_S(t(x, y) \leq \theta)$$

and the true error at margin θ is defined by:

$$e_\theta(c) = \Pr_{D^m}(t(x, y) \leq \theta)$$

Note that $e_0(c) = e_D(c)$ (the true error rate of c).

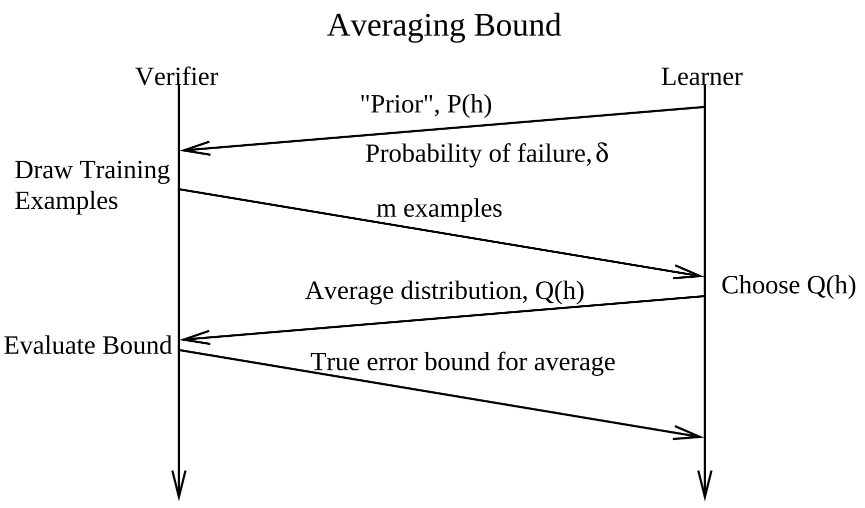


FIGURE 7.0.1. For the averaging bound, the learning must commit to some measure $p(h)$, receive examples, and then commit to another measure $q(h)$. The true error bound applies to the *average* over $q(h)$ rather than stochastic choices as in the PAC-Bayes bound 6.

The "margin" is a useful way to parameterize our learning algorithms. It will turn out that the sample complexity will be low (and the guarantees we can make better) when the margin of most of the training examples is large. There is, however, a price associated with using the margin: some hypotheses have no notion of a margin. Thus the theory in this chapter is less general than appears elsewhere.

7.1. Earlier Results

The improved averaging bound arises from improving one critical step in the proof of the original margin bound [**46**] which is stated next.

THEOREM 7.1.1. *(Margin Bound* [**46**]*) For all $\delta \in (0,1]$, for all base hypothesis spaces, H,*

$$\Pr_{D^m}\left(\exists c, \theta \in (0,1]: \; e(c) > \hat{e}_\theta(c) + O\left(\sqrt{\frac{\frac{\ln |H|}{\theta^2}\ln m + \ln\frac{1}{\delta}}{m}}\right)\right) \leq \delta$$

PROOF. Given in [**46**]. A simplification of the improved averaging bound proof. □

Here, the notation $b(m) = O(a(m))$ means there exists a constant C such that $b(m) \leq C \cdot a(m)$ for all m. This margin bound implies that if most training examples have a large margin θ (i.e. $t(x,y) > \theta$ for most $(x,y) \in S$) and the hypothesis space is not too large, then the generalization error cannot be large. This theorem can only be non-vacuous when the base hypothesis space is finite. There are various extensions (see [**46**]) of this bound for continuous hypothesis spaces based upon VC dimension and covering number techniques. However, the extensions tend to result in extremely loose guarantees and are not relevant to the discussion here.

One of the advantages of the improved averaging bound is that it *can* apply in a non-vacuous way to infinite hypothesis spaces. This generalization comes about with essentially zero loosening of the underlying bound.

7.2. A generalized averaging bound

Before discussing the main theorem, it is important to notice that the averaging classifier, $c(x)$ *implies* a distribution over the base hypothesis space H. This implied distribution is $q_c(h)$ where

$$c(x) = \text{sign}(\int h(x) q_c(h) dh)$$

The distribution q_c is used in the following theorem.

THEOREM 7.2.1. *(Relative Entropy Averaging Theorem) For all distribution $p(h)$, for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists c, \theta \in (0,1]: \; KL\left(\hat{e}_\theta(c)||e(c)\right) \geq O\left(\frac{\frac{KL(q_c||p)}{\theta^2}\ln m + \ln m + \ln\frac{1}{\delta}}{m}\right)\right) \leq \delta$$

PROOF. Given in the next section. □

The main theorem uses a KL-divergence based pseudodistance which is a bit hard to understand intuitively. In order to gain intuition, we can relax the tightness of the proof with an inequality.

$$\text{KL}(\hat{e}_\theta(c)||e(c)) \geq 2(\hat{e}_\theta(c) - e(c))^2$$

This relaxation gives us an immediate corollary.

COROLLARY 7.2.2. *(Relative Entropy Averaging Theorem) For all distribution $p(h)$, for all $\delta \in (0,1]$:*

THEOREM 7.2.3.

$$\Pr_{D^m}\left(\exists c, \theta \in (0,1]: \; e(c) \geq \hat{e}_\theta(c) + O\left(\sqrt{\frac{\frac{KL(q_c||p)}{\theta^2}\ln m + \ln m + \ln\frac{1}{\delta}}{m}}\right)\right) \leq \delta$$

This theorem improves upon theorem 7.1.1 because $\text{KL}(q_c||p)$ is used instead of $\ln|H|$. For the case of a uniform distribution on $|H|$ different base classifiers, these results will agree when the average is over just one classifier. As the average becomes "broader" the results will improve. In the limit when the average is over nearly all classifiers, the term $\text{KL}(q_c||p)$ will be nearly 0.

The theorems are stated in an asymptotic fashion which is not be very useful in practical applications. Section 7.4 gives some ideas of how to tighten the result, and the non-asymptotic form (7.3.15) given at the end of the proof can be used directly in practice.

The improved averaging bound applies to averages over continuous hypothesis spaces. In this setting, the average needs to be an integral over an uncountably-infinite set of hypotheses or the KL-divergence will not converge to a finite value. It is exactly because of this limitation that the improvements of this bound are most applicable to Bayes Optimal and Maximum Entropy classifiers.

In practice, the limitation may not be a significant problem because machine learning algorithms over large hypothesis spaces typically have some parameter stability. In other words, a small shift in the parameters of the learned model produces a small change in the prediction of the hypothesis. With hypothesis stability, we can convert any average over a finite set of hypotheses into an average over an infinite set of hypotheses without significantly altering the predictions of the average. This technique is explored in chapter 13 with positive results.

7.3. Proof of main theorem

7.3.1. Definitions and observations. The proof has the same structure as the original margin bound (7.1.1) proof with one step replaced by the application of the relative entropy PAC-Bayes theorem (6.2.1).

Let N be any natural number; later, the choice of N will be optimized. In the first part of the proof, we regard θ and N as fixed. Later we generalize this so that they may depend on the sample S.

We construct the distribution q_N as follows. Draw N hypotheses $h_i \sim q(h)$ independently. Q_N might therefore be viewed as the product distribution

$$q_c(h)^N. \tag{7.3.1}$$

Given the h_i we define

$$g(x) = \frac{1}{N}\sum_{i=1}^{N} h_i(x) \tag{7.3.2}$$

For fixed x, y, the value of $yh(x)$ are i.i.d. Bernoulli variables with the mean equal to the expected margin:

$$E_{q\sim h} yg(x) = yf(x) \tag{7.3.3}$$

Therefore, $\forall x, y$ $E_{g\sim Q_N}[yg(x)] = yf(x)$ and $yg(x)$ is distributed according to the familiar Binomial distribution. Later, we will apply a Binomial tail bound on this quantity.

Before writing out the proof mathematically, it is helpful to consider a graphical view of what we will prove. We will force convergence between three quantities.

$$\begin{tikzcd} e(h) \arrow[r, dash] & e_{\frac{\theta}{2}}(g) \arrow[r, dash] & \hat{e}_{\frac{\theta}{2}}(g) \arrow[r, dash] & \hat{e}_{\theta}(h) \end{tikzcd}$$

The convergences are then tied together with triangle inequalities resulting in the overall proof. The critical improvement of this paper is a refined version of the second convergence.

7.3.2. The Proof. For every $\theta \in (0, 1]$ and for every (fixed) g, the following simple inequality holds:

$$e(f) = \Pr_D[yf(x) \leq 0] \tag{7.3.4}$$

$$= \Pr_D[yg(x) \leq \frac{\theta}{2}, yf(x) \leq 0] + \Pr_D[yg(x) > \frac{\theta}{2}, yf(x) \leq 0] | yf(x) \leq 0]$$

$$\leq \Pr_D[yg(x) \leq \frac{\theta}{2}] + \Pr_D[yg(x) > \frac{\theta}{2} \mid yf(x) \leq 0]$$

Note that the left-hand side does not depend on g. By taking the expectation over $g \sim Q_N$ (and exchanging the order of expectations in the second term on the right-hand side), we arrive at

$$e(f) \le E_{g \sim Q_N} \left[\Pr_D[yg(x) \le \frac{\theta}{2}] \right] + E_D \left[P_{g \sim Q_N}[yg(x) > \frac{\theta}{2} \,|\, yf(x) \le 0] \right]. \tag{7.3.5}$$

For the rightmost term, we can apply a Binomial tail bound to get:

$$e(f) \le E_{g \sim Q_N} \left[\Pr_D[yg(x) \le \frac{\theta}{2}] \right] + E_D \left[1 - \mathrm{Bin} \left(N, \left\lceil N \frac{1 + \theta/2}{2} \right\rceil, 0 \right) \right] \tag{7.3.6}$$

$$= E_{g \sim Q_N} \left[\Pr_D[yg(x) \le \frac{\theta}{2}] \right] + 1 - \mathrm{Bin} \left(N, \left\lceil N \frac{1 + \theta/2}{2} \right\rceil, 0 \right) \tag{7.3.7}$$

We would like to apply the PAC-Bayes theorem6.2.1 to the right-hand term. Here we use the loss function $I_{\{yg(x) \le \theta/2\}}$. The PAC-Bayes theorem applies for any "prior" distribution. We use as the "prior" the distribution $P_N = p(h)^N$ where $p(h)$ is the "prior" over the base hypothesis space. The choice of this prior allows us to use the identity:

$$\mathrm{KL}(Q_N || P_N) = N \mathrm{KL}\{q(h) || p(h))$$

It follows from Theorem 6.2.1 that: with probability at least $1 - \delta$ over random choices of S, for every Q,

$$\Pr_D \left(\exists q(h) : \mathrm{KL} \left(\hat{e}_{\frac{\theta}{2}}(g) || e_{\frac{\theta}{2}}(g) \right) \le \frac{\mathrm{KL}(Q_N || P_N) + \ln \frac{m}{\delta}}{m - 1} \right) \tag{7.3.8}$$

where $e_{\frac{\theta}{2}}(g) = 1$ with probability $E_{g \sim Q_N} \left[\Pr_D[yg(x) \le \frac{\theta}{2}] \right]$ and 0 otherwise, and $\hat{e}_{\frac{\theta}{2}}(g) = 1$ with probability $E_{g \sim Q_N} \left[\Pr_S[yg(x) \le \frac{\theta}{2}] \right]$ and 0 otherwise.

By the same argument as in (7.3.4), we get:

$$\hat{e}_{\frac{\theta}{2}}(g) = \Pr_S[yg(x) \le \frac{\theta}{2}] \le \Pr_S[yg(x) \le \frac{\theta}{2},\ yf(x) > \theta] + \Pr_S[yf(x) \le \theta] \tag{7.3.9}$$

$$\le \Pr_S[yg(x) \le \frac{\theta}{2} \,|\, yf(x) > \theta] + \Pr_S[yf(x) \le \theta] \tag{7.3.10}$$

$$= \Pr_S[yg(x) \le \frac{\theta}{2} \,|\, yf(x) > \theta] + \hat{e}_\theta(f) \tag{7.3.11}$$

Again, we take expectations over $g \sim Q_N$ on both sides, interchange the order of the expectations and apply the Binomial tail bound to get:

$$E_S \left[P_{g \sim Q_N}[yg(x) \le \frac{\theta}{2}] \right] \le \mathrm{Bin} \left(N, \left\lceil N \frac{1 + \theta/2}{2} \right\rceil, \frac{1 + \theta}{2} \right) + \Pr_S \left[yf(x) \le \theta \right]. \tag{7.3.12}$$

Combining our inequalities, we conclude that with probability at least $1 - \delta$, for every $q(h)$

$$\mathrm{KL}(q_S || p_D) \le \frac{N \mathrm{KL}(q || p) + \ln \frac{m}{\delta}}{m - 1} \tag{7.3.13}$$

where $q_S = 1$ with probability $\mathrm{Bin}\left(N, \left\lceil N\frac{1+\theta/2}{2}\right\rceil, \frac{1+\theta}{2}\right) + \Pr_S\left[yf(x) \leq \theta\right]$ and 0 otherwise, and $p_D = 1$ with probability $\Pr_D[yf(x) \leq 0] - 1 + \mathrm{Bin}\left(N, \left\lceil N\frac{1+\theta/2}{2}\right\rceil, 0\right)$ and 0 otherwise.

This bound holds for any fixed N and θ, which is not yet what we need here, since we want to allow these to depend on the data S. In essence, the bound we proved so far is a statement about certain events, parameterized by N and θ, namely the probability of each event is smaller than δ. However, we need to prove that the probability of the *union* of all these events is smaller than δ. To this end, we first observe that this union is contained in the union over only a *countable* number of events. Note that $g(x) \in \{(2k-N)/N | k = 0, 1, \ldots, N\}$. Thus, even with all the possible (positive) values of θ, there are no more than $N+1$ events of the form $\{yg(x) \leq \theta/2\}$. Denote by $k(\theta, N)$ the largest integer k such that $k/N \leq \theta/2$. We observe that for every $\theta > 0$, every g and every distribution over (x, y):

$$\Pr\left[yg(x) \leq \theta/2\right] = \Pr\left[yg(x) \leq \frac{k(\theta, N)}{N}\right]. \tag{7.3.14}$$

This means that the middle step in the proof above, i.e. the application of theorem 6.2.1, depends on (N, θ) only through (N, k). Since the other steps are true with probability one, we see that we can restrict ourselves to the union of countably many events, indexed by (N, k). Now, we "allocate" parts of the confidence quantity δ to each of these events, namely (N, k) receives $\delta_{N,k} = \delta/(N(N+1)^2)$, $N = 1, 2, \ldots$; $k = 0, \ldots, N$. It follows easily that the union of all these events has probability at most $\sum_{N,k} \delta_{N,k} = \delta$. Therefore we have proved that with probability at least $1-\delta$ over random choices of S it holds true that for *all* N and *all* $\theta \in (0, 1]$,

$$\mathrm{KL}(q_S || p_D) \leq \frac{N\mathrm{KL}(Q||P) + \ln\frac{2m}{\delta_{N,k}}}{m-1} \leq \frac{N\mathrm{KL}(Q||P) + \ln\frac{2m}{\delta} + 3\ln N + 1}{m-1} \tag{7.3.15}$$

We can now choose N to minimize this bound. N may depend on θ, Q and the sample S.

The asymptotic bound stated in the theorem can be derived by choosing N (with respect to θ and Q) so as to *approximately* minimize the bound we have derived above. The first step in this minimization is the replacement of the Binomial tail with a looser approximation such as the Hoeffding bound which gives us:

$$1 - \mathrm{Bin}\left(N, \left\lceil N\frac{1+\theta/2}{2}\right\rceil, 0\right) \leq e^{-2N\frac{\theta^2}{4^2}} = e^{-N\frac{\theta^2}{8}}$$

$$\mathrm{Bin}\left(N, \left\lceil N\frac{1+\theta/2}{2}\right\rceil, \frac{1+\theta}{2}\right) \leq e^{-2N\frac{\theta^2}{4^2}} = e^{-N\frac{\theta^2}{8}}$$

We can then choose

$$N = \left\lceil 8\frac{\ln\frac{m}{\mathrm{KL}(q||p)}}{\theta^2}\right\rceil.$$

This choice gives us:

$$e^{-\frac{N\theta^2}{8}} = \frac{\mathrm{KL}(q||p)}{m}$$

Which implies we have an equation of the form:

$$\mathrm{KL}\left(q+\frac{\mathrm{KL}(q||p)}{m}||p-\frac{\mathrm{KL}(q||p)}{m}\right)\leq\frac{\theta^{-2}\mathrm{KL}(q||p)\ln m+\ln m+\ln\frac{1}{\delta}}{m} \tag{7.3.16}$$

In order to prove the result, we must show that:

$$\mathrm{KL}\left(q+\frac{\mathrm{KL}(q||p)}{m}||p-\frac{\mathrm{KL}(q||p)}{m}\right)=O\left(\mathrm{KL}\left(q||p\right)\right) \tag{7.3.17}$$

We can do this by taking the difference to get an equation of the form:
$(q+k)\ln\frac{q+k}{p-k}+(1-q-k)\ln\frac{1-q-k}{1-p+k}-q\ln\frac{q}{p}-(1-q)\ln\frac{1-q}{1-p}$
If $p-q>2k$ and $k<\frac{1}{2}$ then we have:
$\simeq(q+k)\ln\frac{q+k+\frac{k}{p}}{p}+(1-q-k)\ln\frac{1-q-k-\frac{k}{1-p}}{1-p}-q\ln\frac{q}{p}-(1-q)\ln\frac{1-q}{1-p}$
$\simeq(q+k)[\ln\frac{q}{p}+\frac{k+\frac{k}{p}}{\frac{q}{p}}+(1-q-k)\ln\frac{1-q}{1-p}-\left[\frac{k+\frac{k}{1-p}}{\frac{1-q}{1-p}}\right]-q\ln\frac{q}{p}-(1-q)\ln\frac{1-q}{1-p}$
$\simeq(q+k)[\ln\frac{q}{p}+k(\frac{p+1}{q})]+(1-q-k)[\ln\frac{1-q}{1-p}-k(\frac{2-p}{1-q})]-q\ln\frac{q}{p}-(1-q)\ln\frac{1-q}{1-p}$
$\simeq k(1+p)-k(2-p)$
$\simeq k$
Since we have that the difference

$$\mathrm{KL}\left(q+\frac{\mathrm{KL}(q||p)}{m}||p-\frac{\mathrm{KL}(q||p)}{m}\right)-\mathrm{KL}\left(q||p\right)\simeq\left[\mathrm{KL}\left(q||p\right)\right] \tag{7.3.18}$$

the main theorem follows immediately.

7.4. Methods for tightening

The previous section showed a bound in asymptotic form which is good for understanding the trade-offs between the number of examples (m), the size of the hypothesis space ($|H|$), the margin (θ) and the entropy of the average ($H(Q)$). However, it is not a good form for those interested in quantitative application of the bound to specific problems. We state improvements which aid in the development of a quantitatively applicable bound. We can tighten the bound above through several techniques:

1. Parameterizing and then optimizing the parameterization of arbitrary choices within the proof. In the (improved) margin bound proof, we arbitrarily decided to work with the margin of the randomly produced function $g(x)$ at $\frac{\theta}{2}$. This is a good heuristic, but not the optimal choice when we use the improved tail bounds. Since the decision of the margin for the random function $g(x)$ is a parameter of the proof, we are free to optimize it.
2. Tighter argument within the proof. The optimal value of N is a function of $\theta, m, \mathrm{KL}(q||p)$ and δ. All of these are known in advance except for $\mathrm{KL}(q||p)$. If we can estimate in advance the value of $\mathrm{KL}(q||p)$, then it becomes possible to optimize the value of N in a data-independent manner. Consequently, it becomes unnecessary to split our confidence over the possible values of N and we need only split the confidence over the values of θ in proving the bound. The effect of this improvement is reducing $1/\delta_{N,k}=1/(N(N+1)^2)$ to $1/\delta_{N,k}=1/(N(N+1))$ giving us a small improvement in the low order terms of the improved averaging bound.

7.5. Final thoughts for Averaging Bounds

The practical implication of this more-refined analysis of averaging bounds is more support for the practice of averaging. Sufficient averaging over the hypothesis space can reduce the “complexity” allowing tight estimates on the true error rate—possibly even tighter than could be guaranteed with a single hypothesis.

The improved averaging bound is not yet the tightest possible and it appears there are several possible theoretical improvements.

(1) Remove the low order terms from the bound to make it more quantitatively applicable.

(2) Improve the argument to take into account the *distribution* of the margin rather than the margin at some point.

(3) Prove a lower bound which corresponds to the upper bound given here. Since no good lower bound yet exists, we do not know that large improvements in the upper bound are not possible.

Open Problem: In practice, is the averaging bound (7.3.15) ever better than the PAC-Bayes bound 6.2.1? The extra triangle inequality applications in the averaging bound may make it worse than the PAC-Bayes bound in practice.

CHAPTER 8

Computable Shell bounds

The first shell bound paper was joint work with David McAllester and was presented at Colt [**33**]. The work presented here incorporates significant refinement, generalization, and simplification of the first Colt paper.

Roughly speaking, the shell bound (usually) provides much tighter true error rate upper bounds on learned hypotheses than conventional Occam's Razor bound (theorem 4.6.1) or PAC-Bayes bounds (theorem s6.2.1). It does this without violating lower upper bounds 4.4 by incorporating *much* more information into the bound calculation.

The inspiration behind the work on Shell bounds rests on two pieces of work. In [**22**] by Haussler, Kearns, Seung, and Tishby, learning theory curves are investigated from an omniscient point of view where the true error rates of various hypotheses are known. The principle improvement in this paper is that our bounds are reduced to *observable* quantities. Put another way, we do not need to know the underlying learning distribution, D. In [**47**], an analysis was made assuming some distribution over true error rates. Our analysis does not rely on any assumption about the distribution of true error rates—only the independence assumption is made. Despite using only observable information and making no extra assumptions, the results here are quite tight and yield practical results.

We start with the distribution of empirical errors over hypotheses and subtract a small amount from the empirical error rates to create a pessimistic distribution. With high probability, the cumulative of the pessimistic distribution will lower bound the cumulative distribution of hypothesis true error rates. Given this, we can directly calculate a bound on the probability that a "large" hypothesis will produce a misleadingly small error. This bound can be *much* tighter than standard union bound techniques although the quantity of improvement is highly problem dependent.

After presenting the first bound, we will transform it into a bound on continuous hypothesis spaces using a PAC-Bayes like approach [**39**].

Viewed as an interactive proof of learning (figure 8.0.1), the stochastic shell bound is much like the PAC-Bayes bound.

The strongest criticism of shell bounds is, in fact, that too much information is required. While this information is always theoretically observable, it may not be tractable to collect. There are two answers to this criticism given here. The first is an empirical employment on decision tree learning algorithms which shows that in practice, there are often shortcuts which make the information gathering feasible. The second answer is to construct a sampled version of the shell bound which approximate versions of the information required by the shell bound. We will:

(1) Present the discrete shell bound

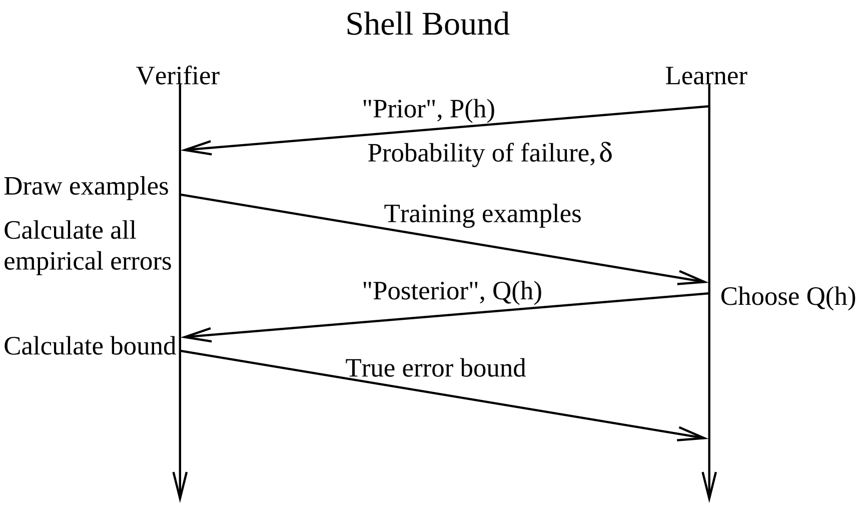


FIGURE 8.0.1. The stochastic shell bound, as an interactive proof of learning, has the same general outline as the PAC-Bayes bound except that *much* more information is required in order to calculate the bound. The shell bound (proved first below) is a simplification which is somewhat tighter when the "Posterior" places all mass on one hypothesis.

(2) Present the sampled version of the shell bound.
(3) Extend the discrete shell bound to continuous spaces

8.1. The Discrete Shell Bound

8.1.1. Knowledge of learning distribution. Let $B(m,K,p) = \binom{m}{K} e(h)^K (1-e(h))^{m-K}$ be the probability that hypothesis with true error p produces K errors on m independent examples.

The discrete shell bound works directly with the probability that there will be a confusingly small empirical error. Let

$$P(\epsilon, K) \equiv \sum_{h: e(h) > K + \epsilon} B(m, K, e(h))$$

Intuitively, $P(\epsilon, K)$ is a bound on the probability that a hypotheses with a true error rate larger than $\frac{K}{m} + \epsilon$ will have an empirical error rate of $\frac{K}{m}$. The contribution to the sum will fall off exponentially as the true error, $e(h)$, increases.

Our first step is stating a shell bound which requires unknown information. The purpose of this bound is motivational - it provides incite into why we can expect a large improvement. Later, we will remove the unknown information requirements and recover a useful bound.

THEOREM 8.1.1. *(Full knowledge theorem) For all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists h \in H \ e(h) \geq \hat{e}(h) + \epsilon(\delta, \hat{e}(h))\right) \leq \delta$$

where $\epsilon(\delta, \frac{K}{m}) = \min\left\{\epsilon : \ P(\epsilon, K) = \frac{\delta}{m}\right\}$

The full knowledge theorem relies on unobservable information—the true error rates of all hypotheses. This theorem is not (quite) trivial because it does not rely upon information about *which* hypothesis has a particular true error.

PROOF. For every hypothesis with a true error rate of

$$e(h) > \frac{K}{m} + \epsilon(\delta, K)$$

the probability of producing an empirical error of

$$\hat{e}(h) = \frac{K}{m}$$

is $B(K, e(h), m)$. Applying the union bound over all hypotheses and all m possible nontrivial values of K completes the proof. □

There are a couple things to note about this theorem. First, for most balanced machine learning problems most hypotheses typically have a true error rate near to 0.5. Given this, and noticing that Binomial tails fall of exponentially, dramatic improvements in the bound are feasible.

Second, we must use $\frac{\delta}{m}$ rather than δ in order to make the proof work. It is possible that theorem holds without splitting δ "m-ways". Removing the factor of m is an open problem. For the special case of empirical risk minimization algorithms, McDiarmid's inequality [**40**] implies that the range of hypotheses with minimum empirical error is of size $O(\sqrt{m})$ with high probability. Therefore, we need only apply the union bound to $O(\sqrt{m})$ possible minimum empirical error rates.

8.1.2. No knowledge of learning distribution. Applying the full knowledge theorem (8.1.1) is not practical in almost all learning settings because we do not know the distribution of true error rates. Therefore, it is necessary to construct a slightly looser theorem which relies upon only observable quantities. Surprisingly, this is possible while introducing only slightly more slack.

First, we need a couple of definitions.

$$\bar{e}(\epsilon, K, \delta, \frac{i}{m}) = \max\left\{\frac{K}{m} + \epsilon,\ \min\left\{p:\ 1 - \mathrm{Bin}(m, i, p) \geq \delta\right\}\right\}$$

Intuitively, $\bar{e}(\epsilon, K, \delta, \frac{i}{m})$ is a *lower* bound on the true error rate of the hypothesis with an empirical error of $\frac{i}{m}$.

$$\hat{P}(\epsilon, K, \delta) \equiv 2\sum_{h} B(m, K, \bar{e}(\epsilon, K, \delta, \hat{e}(h)))$$

The quantity $\hat{P}(\epsilon, K, \delta)$ is an upper bound on the probability that one of the hypotheses with a true error rate of $\bar{e}$ or more could produce K empirical errors.

Noting that there are only $m+1$ possible empirical errors, we can first let

$$c\left(\frac{i}{m}\right) = \left|\left\{h:\ \hat{e}(h) = \frac{i}{m}\right\}\right|$$

be the count of empirical errors at $\frac{i}{m}$. Then we can redefine:

$$\hat{P}(\epsilon, K, \delta) \equiv 2\sum_{i=0}^{m} c\left(\frac{i}{m}\right) B(m, K, \bar{e}(\epsilon, K, \delta, \frac{i}{m}))$$

Later, we will prove that with high probability, $\hat{P}(\epsilon, K, \delta) \geq P(\epsilon, K)$. Given that this is so, we can prove a theorem which *only* relies on observable quantities.

THEOREM 8.1.2. *(Observable Shell Bound) For all $\delta > 0$:*

$$\Pr_{D^m}(\exists h \in H \; e(h) \geq \hat{e}(h) + \epsilon(\delta, \hat{e}(h))) \leq \delta$$

where $\epsilon(\delta, \frac{K}{m}) = \min\left\{\epsilon : \; \hat{P}\left(\epsilon, K, \frac{\delta}{2}\right) \leq \frac{\delta}{2m}\right\}$

The observable shell bound preserves the important locality property of the full knowledge shell bound. In particular, when most of the true error rates are "far" from the empirical error rate (and m is large enough), we expect to make large (functional) improvements on the discrete hypothesis bound 4.2.1.

The proof rests upon a technical lemma which allows us to bound the unobservable "probability of a misleading event" with an observable event.

LEMMA 8.1.3. *(Unobservable bound) For all empirical errors, K, for all distributions $Q(h)$, for all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(2\sum_{h} Q(h)B(m, K, \bar{e}(\epsilon, K, \delta, \hat{e}(h))) \geq \sum_{h:e(h)>K+\epsilon} Q(h)B(m, K, e(h))\right) \leq \delta$$

This lemma is powerful because it bounds the unobservable right hand side in terms of the observable left hand side.

PROOF. Let $\bar{e}\left(m, \frac{k}{m}, \delta\right) \equiv \min_p\{p : \; 1 - \mathrm{Bin}(m, k, p) \geq \delta\}$

$$\forall \alpha \in (0,1] \; \forall h : \; E_{D^m} I(e(h) < \bar{e}(m, \hat{e}(h), \alpha)) \leq \alpha$$

$$\Rightarrow \forall P(h) \; E_P E_{D^m} I(e(h) < \bar{e}(m, \hat{e}(h), \alpha)) \leq \alpha$$

$$\Rightarrow \forall P(h) \; E_{D^m} E_P I(e(h) < \bar{e}(m, \hat{e}(h), \alpha)) \leq \alpha$$

$$\Rightarrow \forall P(h) \; \Pr_{D^m}\left(E_P I(e(h) < \bar{e}(m, \hat{e}(h), \alpha)) \geq \frac{\alpha}{\delta}\right) \leq \delta$$

$$\Rightarrow \forall P(h) \; \Pr_{D^m}\left(\Pr_{P}(e(h) < \bar{e}(m, \hat{e}(h), \alpha)) \geq \frac{\alpha}{\delta}\right) \leq \delta$$

Let $P(h) \propto Q(h)B(m, K, e(h))$. Then:

$$\Rightarrow \forall Q(h) \; \Pr_{D^m}\left(\frac{\sum_{h:e(h)>\frac{K}{m}+\epsilon \wedge e(h)>\bar{e}(m,\hat{e}(h),\alpha)} Q(h)B(m, K, e(h))}{\sum_{h:e(h)>\frac{K}{m}+\epsilon} Q(h)B(m, K, e(h))} \geq \frac{\alpha}{\delta}\right) \leq \delta$$

set $\alpha = \frac{\delta}{2}$and replace $e(h)$ with $\bar{e}(m, \hat{e}(h), \alpha)$ to achieve the result. □

We now have all the tools required to prove the theorem.

PROOF. (of theorem 8.1.2) Choose $Q(h)$ = the uniform distribution on a our hypothesis space. Then, we know that with probability $1 - \delta$, $\hat{P}(\epsilon, K, \delta) \geq P(\epsilon, K)$. Therefore, we can (arbitrarily) allocate a $\frac{\delta}{2}$ probability of failure to the unobservable bound 8.1.3 and a $\frac{\delta}{2}$ probability of failure to the full knowledge bound 8.1.1. Assuming $\hat{P}(\epsilon, K, \delta) \geq P(\epsilon, K)$, the observable bound will be more pessimistic than the full knowledge bound. □

The Observable Shell bound behaves in a strange manner which is unlike other true error bounds. In particular, the true error bound can be discontinuous in the value of δ. This discontinuity implies that relatively small improvements in the shell bounds can result in dramatic improvements in the value of the true error bound. While dramatic improvements can happen with small improvements in the shell bound, we expect that in practice, such large improvements will not be too common, simply because a small improvement is unlikely (amongst all learning problems) to shift the bound across one of these discontinuous transitions.

8.2. Sampling Shell Bound

The Shell bound relies upon the distribution of empirical errors across the entire hypothesis space. Calculating $c\left(\frac{i}{m}\right)$, while theoretically possible, is often practically intractable. For example, the space of all binary functions on n features has size 2^{2^n} and for a decision tree with a number of nodes $k = O(m)$ there are more than 2^k hypotheses with the same number of nodes. In order to avoid this difficulty, we will use sampling which is made possible by noticing that the Shell bound does not require exact knowledge of the empirical error distribution. Instead, we can safely count a hypothesis twice because over-counting monotonically worsens the bound. Assume that we have an oracle which can be used to sample uniformly from the set of all hypotheses. Then, we can bin the sampled hypotheses according to their error rate on the example set. After repeating k times we will arrive at an empirical distribution over error rates which can be altered into a bound on the true distribution of error rates by upper bounding the count $c\left(\frac{i}{m}\right)$. Let $\hat{c}\left(\frac{i}{m}\right)$ be the observed number of hypotheses out of k uniform choices with empirical error $\frac{i}{m}$ and define:

$$\bar{c}\left(k, \frac{\hat{c}}{k}, \delta, |H|\right) \equiv |H| * \max_{p}\left\{p : \ \mathrm{Bin}(k, \hat{c}, p) = \frac{\delta}{m}\right\}$$

Intuitively, $\bar{c}$ will be a high confidence upper bound on the number of hypotheses with empirical error rate $\hat{c}$. Given these quantities, we can calculate an approximate $\hat{P}$ according to the formula:

$$\hat{\hat{P}}(\epsilon, K, \delta) \equiv 2\sum_{i=0}^{m} \bar{c}\left(k, \frac{\hat{c}\left(\frac{i}{m}\right)}{k}, \frac{\delta}{2m}, |H|\right) B\left(m, K, \bar{e}\left(\epsilon, K, \frac{\delta}{2}, \frac{i}{m}\right)\right)$$

THEOREM 8.2.1. *(Sampling Shell Bound) For all $\delta > 0$:*

$$\Pr_{D^m}\left(\exists h \in H \ e(h) \leq \hat{e}(h) + \epsilon(\delta, \hat{e}(h))\right) \leq \delta$$

where $\epsilon(\delta, K) = \min\left\{\epsilon : \ \hat{\hat{P}}\left(\epsilon, K, \frac{\delta}{2}\right) \leq \frac{\delta}{2m}\right\}$

PROOF. For every i, we know that $\hat{\hat{P}}(\epsilon, K, \delta) \geq \hat{P}\left(\epsilon, K, \frac{\delta}{2}\right)$ with probability at least $1 - \frac{\delta}{2}$. This implies that $\hat{\hat{P}}(\epsilon, K, \delta) \geq P(\epsilon, K)$ with probability at least $1 - \delta$. Applying the union bound with the full knowledge theorem gives us the result. □

The Sampling Shell bound is still relatively fast to calculate given the sampling results, but it is worth noting that *many* samples are required—the bound will typically tighten linearly with $\ln k$. In other words, an exponentially large k is required to realize all of the gains of the shell bound. Thus, the sampling shell bound will interpolate between the discrete hypothesis bound 4.2.1 and the shell bound as $\ln k$ moves from 1 to $\ln |H|$.

8.3. Lower Bounds

The lower upper bound 4.4 does not apply to shell bounds because we are using more information than just the empirical error rate of a learned hypothesis. In particular, we are using the empirical error rates of *all* the hypotheses in calculating the bound. Is there a lower upper bound which applies for the information used by the shell bound? The same independent hypothesis technique will allow us to lower bound the full knowledge theorem 8.1.1. In particular, assume that we are given a set of independent hypotheses, each with some true error $e(h)$. What is a lower bound on the probability that one of these hypotheses will have an empirical error of $\frac{K}{m}$?

If A and B are independent events, then:

$$\begin{aligned}\Pr(A \text{ or } B) &= \Pr(A) + \Pr(B) - \Pr(A \text{ and } B)\\ &= \Pr(A) + \Pr(B) - \Pr(A)\Pr(B)\\ &= (1-\Pr(B))\Pr(A) + \Pr(B)\end{aligned}$$

This implies that we can "add" the independent probabilities together as long as we rescale. In particular, B might be the probability of a "bad" hypothesis in some set of hypotheses and A might be the probability that some new hypothesis with a large true error rate has a small empirical error.

Using this fact, we get the following theorem:

THEOREM 8.3.1. *(Lower Upper Shell Bound) For all true errors $e(h)$, K:*

$$\Pr_{D^m}\left(\exists h : e(h) > \frac{K}{m} + \epsilon \text{ and } \hat{e}(h) = \frac{K}{m}\right) \geq P(\epsilon, K)(1 - P(\epsilon, K))$$

PROOF. The proof is by finite induction on the set of hypotheses with a large true error rate. Let

$$P_H(\epsilon, K) = \sum_{h \in H} B(m, K, e(h))$$

be the sum of the probabilities that each hypothesis in H' produces an empirical error of $\frac{K}{m}$. Now, we want to prove that:

$$\forall H \ \Pr_{D^m}\left(\exists h \in H : \ \hat{e}(h) = \frac{K}{m}\right) \geq P_H(\epsilon, K)(1 - P_H(\epsilon, K)$$

This is true for the base case of $|H| = 1$. Assuming that it is true for the case of $|H| = n$, we need to prove it for the case of $H' = H \cup |h|$. In particular, we have assumed

$$\Pr_{D^m}\left(\exists h \in H : \ \hat{e}(h) = \frac{K}{m}\right) \geq P_H(\epsilon, K)(1 - P_H(\epsilon, K)$$

Using the earlier independent principle, we get that:

$$\begin{aligned}&\Pr_{D^m}\left(\exists h \in H' : \ \hat{e}(h) = \frac{K}{m}\right)\\ &\geq P_H(\epsilon, K)(1 - P_H(\epsilon, K)) + (1 - P_H(\epsilon, K)(1 - P_H(\epsilon, K)))B(K, e(h), m)\\ &\geq P_H(\epsilon, K)(1 - P_H(\epsilon, K)) + (1 - P_H(\epsilon, K))B(K, e(h), m)\\ &= P_{H'}(\epsilon, K)(1 - P_H(\epsilon, K))\end{aligned}$$

$$\geq P_{H'}(\epsilon, K)(1 - P_{H'}(\epsilon, K))$$

By induction, this property therefore holds for the set of all hypotheses with a large true error rate. □

Assuming that $\delta < \frac{1}{2}$, the lower upper shell bound is tight to within a factor (in δ) of $2m$ with the full knowledge bound 8.1.1. Given the exponential behavior of Binomial tails, this usually (but not always) implies a small impact on the true error bound. One important question remains: how does this bound compare to the observable shell bound 8.1.2? In the observable shell bound, the distribution of true errors is replaced with a pessimistic distribution based upon the observed empirical errors. The "size" of this pessimism in terms of the true error bound is, in general, of size $\frac{1}{\sqrt{m}}$. Thus, the gap between the lower upper shell bound and the upper shell bound is typically of size $\frac{1}{\sqrt{m}}$.

8.4. Shell Bounds for Continuous Spaces

Applying Shell bounds to continuous hypothesis spaces is not easy. In fact, upon first inspection, this appears to be impossible since shell bounds require knowledge of the number of hypotheses with a particular empirical error. We can avoid these difficulties, while introducing a small amount of slack, by always being concerned with the *measure* rather than the count of hypotheses. In particular, we will keep track of the *measure* of hypotheses with a confusingly small error and the *measure* of the hypotheses that we pick. The approach here is similar to the approach in section 6 although more simplistic.

First assume that there is some measure Q over the hypothesis space H. Suppose that we choose some subset, U, of the hypotheses with measure $Q(U)$. We will be concerned with the *largest* empirical error rate $\hat{e}_U(h) = \max_{h \in U} \hat{e}(h)$ and the *average* true error rate, $e_U(h) = E_{Q_U} e(h)$. A bound on the gap between the largest empirical error rate and the average true error rate will imply a true error bound for the stochastic classifier which chooses randomly and evaluates.

We will need a different definition of P. Let:

$$P_s(\epsilon,\ K) \equiv \int_{h:e(h)>K+\epsilon} Q(h)\mathrm{Bin}(K, e(h), m)dh$$

We will also need the concept of rounding. Choose i such that $\frac{1}{m^i} \geq Q(U) \geq \frac{1}{m^{i+1}}$ then, define:

$$\lfloor Q(U) \rfloor = \frac{1}{m^{i+1}}$$

Now, the following theorem holds:

THEOREM 8.4.1. *(Stochastic Full knowledge) For all distributions Q, For all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists U\ e_U(h) \leq \hat{e}_U(h) + \frac{1}{m} + \epsilon(\delta, \hat{e}, U)\right)$$

where $\epsilon\left(\delta, \frac{K}{m}, U\right) = \min\left\{\epsilon :\ P_s(\epsilon, K) \leq \frac{\delta \lfloor Q(U) \rfloor}{m^3}\right\}$

PROOF. Call a hypothesis with a large true error $(e(h) > \frac{K}{m} + \epsilon)$ and small empirical error $(\hat{e}(h) \leq \frac{K}{m})$ a "bad" hypothesis. $P_s(\epsilon, K)$ is the expected measure of the bad hypotheses. We will use Markov's inequality to bound the actual measure of bad hypotheses. Then, given that the quantity is bounded, we can bound the expected true error by assuming that we included every bad hypothesis in the set U.

Let

$$\epsilon_{Ki} = \min\left\{\epsilon : \ P_s(\epsilon, K) \leq \frac{\delta}{m^{3+i}}\right\}$$

Intuitively, ϵ_{Ki} is the value we will use when $\hat{e} = \frac{K}{m}$ and $\lfloor Q(\hat{e})\rfloor = \frac{1}{m^i}$. Also let

$$\hat{P}_s(\epsilon, K) = \int_{h: e(h) > K+\epsilon \wedge \hat{e}(h) \leq K} p(h)dh$$

be the actual measure of bad hypotheses. Then, Markov's inequality tells us:

$$\forall \delta > 0 \ \Pr_D(\hat{P}_s(\epsilon_{Ki}, K) \geq \frac{m^2}{\delta} P_s(\epsilon_{Ki}, K)) \leq \frac{\delta}{m^2}$$

Taking the union bound over all values of ϵ_{Ki}, we get:

$$\forall \delta > 0 \ \Pr_D(\forall K, i \ \hat{P}_s(\epsilon_{Ki}, K) \geq \frac{m^2}{\delta} P_s(\epsilon_{Ki}, K)) \leq \delta$$

So, with probability $1-\delta$ for all values of K and i, we have: $\hat{P}_s(\epsilon_{Ki}, K) \leq \frac{1}{m^{1+i}}$. Therefore, if $Q(U) \geq \frac{1}{m^i}$, we know that $\frac{Q(U)}{\hat{P}_s(\epsilon_{Ki}, K)} \geq m$. Assuming that all of the bad hypotheses have true error 1 and all the rest have true error at most $\frac{K}{m} + \epsilon_{Ki}$, we get the following true error bound:

$$e_U(h) \leq \hat{e}_U(h) + \frac{1}{m} + \epsilon_{\hat{e}i}$$

□

The stochastic full knowledge bound is loose when applied to the full knowledge setting by $\delta \to \frac{\delta}{m^2}$. Typically, this results in only a $\frac{2 \ln m}{m}$ increase in the size of ϵ, although the increase can sometimes be much larger near phase transitions. These factors of $\frac{\ln m}{m}$ may be removable with improved argumentation. Naturally, the stochastic full knowledge bound can be used to prove a stochastic observable bound.

The next theorem is the observable analog of the stochastic full knowledge bound. Here, we eliminate the unobservable quantities to produce a stochastic observable shell bound. The observable quantity we will use is:

$$\hat{P}_s(\epsilon, K, \delta) \equiv 2 \sum_h Q(h) \mathrm{Bin}(K, \bar{e}(\epsilon, K, \delta, \hat{e}(h)), m)$$

where

$$\bar{e}(\epsilon, K, \delta, \frac{i}{m}) = \max \ \{K + \epsilon, \ \min \ \{p : \ 1 - \mathrm{Bin}(m, K, p) \geq \delta\}\}$$

is a slightly pessimal estimate of the true error rate given the empirical error rate as before.

THEOREM 8.4.2. *(Stochastic Observable Shell Bound) For all distributions Q, For all $\delta \in (0,1]$:*

$$\Pr_{D^m}\left(\exists U \ e_U(h) \leq \hat{e}_U(h) + \frac{1}{m} + \epsilon(\delta, \hat{e}, U)\right)$$

where $\epsilon\left(\delta, \frac{K}{m}, U\right) = \min\left\{\epsilon : \ \hat{P}_s(\epsilon, K, \frac{\delta}{2}) \leq \frac{\delta\lfloor Q(U)\rfloor}{2m^3}\right\}$

PROOF. First note that the unobservable bound lemma 8.1.3 implies that with probability $1 - \frac{\delta}{2}$, we have $\hat{P}_s(\epsilon, K, \frac{\delta}{2}) \geq P_s(\epsilon, K)$. Given that this is the case, our choice of ϵ will be at least as pessimistic as the choice defined by the Stochastic Full Knowledge bound 8.4.1. We thus have two sources of failure: the unobservable bound lemma will fail with probability at most $\frac{\delta}{2}$ and the stochastic full knowledge bound will fail with probability $\frac{\delta}{2}$. The union bound then implies that the Stochastic Observable Shell Bound holds with probability $1 - \frac{\delta}{2}$. □

The information requirements for the continuous space shell bound are even more severe than the requirements for the discrete space shell bound. In particular, we need to know the exact measure of the hypotheses with any particular empirical error. Clearly, this is unrealistic. We can relax this requirement to observations computable in a finite amount of time using the sampling techniques of theorem 8.2.1. In particular, given a well-behaved distribution Q, we can make a bounded estimate of the measure of hypotheses with some empirical error rate. Given this bounded estimate, we can then calculate a pessimistic shell bound for the continuous space.

8.5. Conclusion

Details for calculating the shell bound are given in appendix section 16.3.

Shell bounds are easily calculable and can provide large improvements when we can afford to enumerate the hypotheses. Their application is less clear when it is not possible to enumerate the hypotheses because the computational burden may become too large. Via sampling techniques, it is possible to smoothly improve from earlier techniques to the best achievable shell bound result in an anytime fashion.

There remain several important open questions:

(1) Can we remove the remaining division of δ by m? This would make shell bounds a bit more elegant and tight.
(2) Is there a natural lower bound on the true error rate which uses the shell approach? This improvement is of principally theoretical interest.
(3) For the continuous space shell bounds, two additional divisions by m were introduced. Is it possible to remove these factors with an improved argument? This improvement would clean up the continuous shell bound.
(4) Our extension to the continuous case was done in the style of PAC-Bayes bounds, but a more common technique for extending to the continuous case is via the use of covering numbers. Is there a natural way to extend Shell bounds to the continuous case using the concept of a covering number? This is another approach which might yield a result requiring less calculation.

CHAPTER 9

Tight covering number bounds

9.1. Introduction

Covering number bounds are used to bound the true error rate of classifiers chosen from an infinite hypothesis space using m examples [**20**]. A cover is a finite set of hypotheses which satisfies the following property: every hypothesis in the infinite space is "near" some element in the finite cover. When a Lipschitz condition holds on the hypothesis space, it is generally possible to construct these covers and the existence of a cover is required for learnability [**2**]. Alternatively, Sauer's lemma (see [**43**] or [**51**]) bounds the size of the cover in terms of the VC dimension which is defined combinatorially: the VC dimension is the largest number of examples which the hypothesis space can classify in an arbitrary manner.

The principal disadvantage of covering number results is that they are notoriously loose, to the point that they are often useless when applied in practice (see "criticisms" in [**20**]). Here, "useless" means that the bound on the true error rate is "always wrong". The amount of "looseness" can be quantified by comparison with other bounds in the regimes where other bounds hold. On a finite hypothesis space we have near-perfect agreement between the upper bound 4.2.1 and the lower upper bound 4.4.2 for independent hypotheses. In fact, as the number of examples goes to infinity, the agreement is perfect, regardless of the size of the hypothesis space. When we apply covering number bounds to this problem such properties do not arise. Since part of the argument involves splitting the examples into 2 sets, the difference between a covering number based upper bound and the lower upper bound can be large even when the number of examples goes to infinity. In practice, the covering number bound at least squares the discrete hypothesis size. Obviously, further loosening of covering number bounds with Sauer's lemma result in even worse bounds.

Can we construct a calculable true error upper bound for continuous hypothesis spaces which is at least asymptotically tight? In a sense, this has already been done with PAC-Bayes bounds in chapter 6, but there are drawbacks in applicability to that approach since PAC-Bayes bounds do not apply in a meaningful way to a single hypothesis drawn from an infinite hypothesis space. A covering number argument would hopefully apply in a meaningful way to a singly hypothesis. A covering number bound which is asymptotically tight on *some* learning problems does exist and is covered next.

9.2. The Setting and Prior Results

We will first discuss standard covering number bounds. Define a "distance" in terms of how often hypotheses disagree according to:

$$d_D(h,f) = \Pr_D(h(x) \neq f(x))$$

Now, start with an epsilon net defined by:

$$N(H,\epsilon,d_D) = \inf_F |F : \ \forall h \in H \exists f \in F : \ d_D(h,f) \leq \epsilon|$$

An epsilon net is the minimum size of a set which contains an element "near" to every element in H.

Then a covering number is defined as:

$$C(H,\epsilon) = \sup_D N(H,\epsilon,d_D)$$

The covering number is the worst epsilon net.

THEOREM 9.2.1. *(Covering number bound) For all $\delta \in (0,1]$:*

$$\forall H \ \Pr_{D^m}\left(e(h) \geq \hat{e}(h) + 4\sqrt{\frac{\ln 4C(H,\epsilon) + \ln\frac{1}{\delta}}{m}}\right) \leq \delta$$

PROOF. In [**20**]. □

How tight is this bound when applied to a finite independent hypothesis space? We can improve the constants by using an argument with fewer triangle inequalities in the discrete case and get the following results:

$$\Pr_D\left(e(h) - \hat{e}(h) \geq 2\sqrt{\frac{\ln 4|H| + \ln \delta}{m}}\right) \leq \delta$$

Comparing this with a very loose application of the discrete hypothesis bound 4.2.3 we see that the penalty term in the covering number bound is worse by factor of $2\sqrt{2}$. Put another way, dividing the number of samples by 8 or increasing the hypothesis space size to $|H|^8$ and then applying a sloppy discrete hypothesis bound is about equivalent to applying a very specialized covering number bound. We seek a covering number bound which does not divide the effective value of a hypothesis by 8.

9.3. Bracketing Covering Number Bound

There is an alternative version of the covering number bounds which has been little used in learning theory. This is mentioned as the "direct method" in [**20**] and Section II.2 of [**43**] discusses this approach but is only concerned with asymptotic consistency rather than rates of convergence.

We start with a more restricted notion of covering—a covering in which we bracket the hypotheses. Let Z be the space of (x,y) labeled examples and $h(Z) = \{(x,y) : h(x) \neq y\}$ in:

$$N_{[]}(H,\gamma,d_D) = \inf_F \left| F : \forall h \in H \ \exists (f,f') \in F : \ \begin{array}{c} d_D(f,f') \leq \gamma \\ \text{and } f(Z) > h(Z) > f'(Z) \end{array} \right|$$

In other words, f and f' are two sets which are "above" and "below" every hypothesis that they cover. Note that it is very important in this definition that the sets f

and f' are not required to correspond to functions in H. In fact, they can simply be understood as arbitrary sets of (x, y) pairs.

It is immediately obvious that:

$$N_{[]}(H, \gamma, d_D) \geq N(H, \gamma, d_D)$$

However, the relationship between $N_{[]}(H, \gamma, d_D)$ and $C(H, \gamma)$ is ambiguous: either could be larger.

Using this alternative notion of a covering, we have the following theorem:

THEOREM 9.3.1. *Let $f(h)$ be the upper bracket of any hypothesis, h. For all $\gamma \in (0, 1]$, for all $\delta \in (0, 1]$:*

$$\Pr_{D^m}\left(\exists h \in H: \begin{array}{c} e(h) \geq \bar{e}\left(m, \hat{e}(f(h)), \frac{\delta}{2N_{[]}(H,\gamma,d_D)}\right) \\ \text{or } \hat{e}(f(h)) - \hat{e}(h) \geq b\left(m, \gamma, \frac{\delta}{2N_{[]}(H,\gamma,d_D)}\right) \end{array}\right) \leq \delta$$

$$\text{where} \quad \begin{array}{c} \bar{e}\left(m, \frac{K}{m}, \delta\right) \equiv \max_p\{p: \ Bin(m, K, p) = \delta\} \\ \text{and } b\left(m, p, \delta\right) \equiv' \min_K \left\{\frac{K}{m}: \ 1 - Bin(m, K, p) \leq \delta\right\} \end{array}.$$

This theorem constrains the distance between $\hat{e}(f(h))$ and $e(h)$ and the distance between $\hat{e}(f(h))$ and $\hat{e}(h)$. Consequently, it constrains the distance between $e(h)$ and $\hat{e}(h)$. The proof of the theorem rests on the following two lemmas which each assert a convergence. Graphically, the proof has the following form:

$$\begin{tikzcd} e(f(h)) \arrow[r, dash] & \hat{e}(f(h)) \arrow[r, dash] & \hat{e}(h) \end{tikzcd}$$

LEMMA 9.3.2. *Let $f(h)$ be the upper bracket of any hypothesis, h. For all $\gamma \in (0, 1]$, for all $\delta \in (0, 1]$:*

$$\Pr_{D^m}\left(\exists h \in H: \ \hat{e}(f(h)) - \hat{e}(h) \geq b\left(m, \gamma, \frac{\delta}{N_{[]}(H, \gamma, d_D)}\right)\right) \leq \delta$$

where $b\left(m, p, \delta\right) \equiv' \min_K \left\{\frac{K}{m}: \ 1 - Bin(m, K, p) \leq \delta\right\}$

PROOF. If $f(h), f'(h)$ bracket h, we have:

$$e(f(h)) - e(f'(h)) \leq \gamma$$

Therefore a coin which is "heads" when $f(h)(z) \neq f'(h)(z)$ has a bias of γ or less. Since $f'(h)$ is correct everywhere that h is correct, we also know:

$$\forall \epsilon \ \Pr_{D^m}(\hat{e}(f(h)) - \hat{e}(h) \geq \epsilon) \leq \Pr_{D^m}(\hat{e}(f(h)) - \hat{e}(f'(h)) \geq \epsilon)$$

Therefore, we have at most $N_{[]}(H, \gamma, d_D)$ Binomial distributions, each with bias at most γ, and wish to bound the probability of a large deviation. Using an upper binomial tail calculation, we get the result. □

LEMMA 9.3.3. *For all $\delta \in (0, 1]$:*

$$\Pr_{D^m}\left(\exists f, f' \in N_{[]}(H, \gamma, d_D): \ e(f(h)) \geq \bar{e}\left(m, \hat{e}(f(h)), \frac{\delta}{N_{[]}(H, \gamma, d_D)}\right)\right)$$

where $\bar{e}\left(m, \frac{K}{m}, \delta\right) \equiv \max_p\{p: \ Bin(m, K, p) = \delta\}$

PROOF. Application of the discrete hypothesis space bound 4.2.1 for f in every (f, f') pair. □

We can now combine the lemmas to get the theorem.

Proof. (of theorem 9.3.1) Allocate $\frac{\delta}{2}$ confidence to each lemma and use the union bound with both lemmas to get:

$$\Pr_{D^m}\left(\exists h \in H : \begin{array}{c} e(f(h)) \geq \bar{e}\left(m, \hat{e}(f(h)), \frac{\delta}{2N_{[]}(H,\gamma,d_D)}\right) \\ \hat{e}(f(h)) - \hat{e}(h) \geq b\left(m, \gamma, \frac{\delta}{2N_{[]}(H,\gamma,d_D)}\right) \end{array}\right) \leq \delta$$

where $\begin{array}{c} \bar{e}\left(m, \frac{K}{m}, \delta\right) \equiv \max_p\{p : \ \mathrm{Bin}(m,K,p) = \delta\} \\ \text{and } b\left(m,p,\delta\right) \equiv' \min_K\left\{\frac{K}{m} : \ 1 - \mathrm{Bin}(m,K,p) \leq \delta\right\} \end{array}$. Since $e(h) \leq e(f(h))$ by construction, the proof is complete. □

The alternative covering number argument has a significant advantage over the standard argument: when restricted to a finite hypothesis space, the argument becomes tight. In particular, on a finite hypothesis space, we can set $\gamma = 0$ and get: $N_{[]}(H, 0, d_D) \leq |H|$. The bound reduces to the standard discrete hypothesis space bound 4.2.1. Consequently, there exist learning problems where this bound is quite tight.

The bracketing covering approach has the following advantages and disadvantages:

(1) Advantage: Covering defined in terms of the actual problem rather than a worst case over all problems.
(2) Advantage: Asymptotically tight for some learning problems.
(3) Disadvantage: Less general. There may exist spaces which are not coverable in the alternative approach.
(4) Disadvantage: $N_{[]}(H, \gamma, d_D)$ is more difficult to compute than $N(H, \gamma, d_D)$.

In a sense, this bound is useless because it requires knowledge of the unknown distribution D in order to calculate a covering number. In the next section, we will see that bounds on the bracketing covering number which hold for all D can often be found.

9.4. Covering number calculations

It is important to demonstrate that this covering number is feasible to calculate and gives a better answer than the traditional approach. We will do this by first calculating the bracketing covering number for a very simple continuous classifier and then comparing the results with the traditional covering number approach.

Bracketing covering numbers have already been proved for many function classes **[13]**[**49**]. Here, we will present a proof for the simplest of continuous hypothesis spaces: the step function on a line segment. Each hypothesis will be indexed by a number $a \in [0, 1]$ according to:

$$h_a(x) = \mathrm{sign}(x - a)$$

What is $N_{[]}(H, \gamma, d_D)$ for this hypothesis set?

Theorem 9.4.1. *Assume that D can be described by a probability density function, then:*

$$N_{[]}(H, \gamma, d_D) \leq \frac{1}{\gamma} + 1$$

PROOF. Consider a range of hypotheses from h_a to h_b. For this range of hypotheses, we can choose a bracketing pair, (f, f'). In particular, we can choose f and f' which agree on $[0, a)$ and $[b, 1]$ and always predict either incorrectly (f) or correctly (f') on $[a, b)$. The distance between these functions satisfies:

$$d_D(f, f') = \Pr_D(x \in [a, b))$$

and every hypothesis h_c in $[a, b)$ satisfies $\forall z\ f(z) \geq h_c(z) \geq f'(z)$. Consequently, if a and b are chosen appropriately, we will observe $d_D(f, f') \leq \gamma$.

If D can be described by a probability distribution, then we can simply calculate the marginal distribution, D_x, and the cumulative distribution of the margin, $F_D(x)$. Now, find a_i for which $F_D(a_i) = \frac{i}{\gamma}$ for $i < \frac{1}{\gamma}$. Choose $b_i = a_{i+1}$. There are at most $\frac{1}{\gamma} + 1$ intervals, each with a measure (according to D) of at most γ. Consequently, we can cover H with $\frac{1}{\gamma} + 1$ pairs of (f_i, f_i'). □

Given that the bracketing cover is $\frac{1}{\gamma} + 1$, we can use theorem 9.3.1 to define a constraint that the true error rate must satisfy with high probability. Setting $\gamma = \frac{1}{m-1}$, we get:

$$\hat{e}(f(h)) \leq \hat{e}(h) + b\left(m, \frac{1}{m-1}, \frac{\delta}{2m}\right)$$

and

$$e(h) \leq \bar{e}\left(m, \hat{e}(f(h)), \frac{\delta}{2m}\right)$$

To be fair in comparison to the standard covering number approaches, we should relax our theorem to use the Hoeffding approximation. Note that this is a bit unfair because the first inequality is (inherently) a highly biased Binomial with lower variance. Relaxing to the Hoeffding bound, we get:

$$\hat{e}(f(h)) \leq \hat{e}(h) + \frac{1}{m-1} + \sqrt{\frac{\ln \frac{2m}{\delta}}{2m}}$$

and

$$e(h) \leq \hat{e}(f(h)) + \sqrt{\frac{\ln \frac{2m}{\delta}}{2m}}$$

which implies:

$$e(h) \leq \hat{e}(h) + \frac{1}{m-1} + 2\sqrt{\frac{\ln 2m + \ln \frac{1}{\delta}}{2m}}$$

Note again that we are being "unfair" to the new approach by using the Hoeffding approximation rather than exact Binomial-tail bounds. The standard covering number approach has not yet been reduced to exact Binomial-tail bounds. Using the standard approach, the covering number, we get $C(H, \frac{1}{m-1}) = m$. This implies a bound of:

$$e(h) \leq \hat{e}(h) + 4\sqrt{\frac{\ln 8m + \ln \frac{1}{\delta}}{m}}$$

Comparing the bounds, we see that the new approach is about $\frac{16}{2} = 8$ times more efficient in the number of examples required to achieve a bound on a given deviation.

9.4.1. Note on the Bracketing Cover proof. There are several important things to note about this proof.

(1) We used the property that a small change in the hypothesis only affected the prediction on a small portion of the input space.
(2) The bound on $N_{[]}$ holds for all D with a density function, not just the D which we happen to observe.
(3) The bound on $N_{[]}(H, \gamma, D)$ is exactly the same as a bound on $N\left(H, \frac{\gamma}{2}, D\right)$.

In fact, the proof can be extended to *all* D (even ones with point masses) at the cost of a factor of 2 worsening and a messier argument. Property (2) is desirable because it is often not the case that we know the distribution D when we wish to apply the bound. Property (1) is an essential technique that can be used to prove other covering number bounds for this notion of covering number.

Can we show partial order covering number bounds for other classifiers? There is a straightforward extension of the previous proof for classifiers which consist of axis parallel intervals in R^n. More work is required to prove partial order covering numbers for the hypothesis spaces of standard learning algorithms.

9.5. Conclusion and Future Work

We presented an alternative covering number argument and showed that the true error rate bounds constructed using this argument are within $O\left(\frac{\ln m}{m}\right)$ of the lower bound on some learning problems. This is a significant improvement over prior results which just bound the ratio of the lower and upper bounds up to a constant. We also presented a simple improvement on PAC-Bayes bounds for stochastic classifiers which achieves a similar $O\left(\frac{\ln m}{m}\right)$ difference between the lower and upper bounds.

It is interesting to examine the relationship between the bracketing covering number and the PAC-Bayes bound. With this notion of covering number we can guarantee that all of the hypotheses covered by the same bracketing pair have similar empirical as well as true errors. Thus, we can relate the error rate of an individual hypothesis to a set of hypotheses with a significant measure — exactly the setting where the PAC-Bayes bound is tight.

Much work remains to be done in order to fulfill a quest for quantitatively tight learning bounds.

(1) Proofs on the size of the partial order covering number need to be made for common learning algorithms.
(2) Can this alternate form of covering number be related to the VC dimension or to the standard definition of covering number?
(3) Can we extend the class of problems for which the lower and upper bounds differ by only $O\left(\frac{\ln m}{m}\right)$ to a larger set?

CHAPTER 10

Holdout bounds: Progressive Validation

10.1. Progressive Validation Technique

Progressive validation is a technique which allows you to use almost $\frac{1}{2}$ of the data in a holdout set for training purposes while still providing the same guarantee as the holdout bound. It first appeared in [**3**] and is discussed in a more refined and detailed form here.

Suppose that you have a training set of size m_{train} and test set of size m_{pv}. Progressive validation starts by first learning a hypothesis on the training set and then testing on the first example of the test set. Then, we train on training set plus the first example of the test set and test on the second example of the test set. The process continues m_{test} iterations. Let m abbreviate m_{pv}. Then, we have m hypotheses, $h_1, ..., h_m$ and m error observations, $\hat{e}_1, ..., \hat{e}_m$. The hypothesis output by progressive validation is the randomized hypothesis which chooses uniformly from $h_1, ..., h_m$ and evaluates to get an estimated output. Note that this protocol is similar to those in [**36**] and the new thing here is an analysis of performance.

Since we are randomizing over hypotheses trained on m_{train} to $m_{\text{train}} + m_{\text{pv}} - 1$ examples, the expected number of examples used by any hypothesis is $m_{\text{train}} + \frac{m_{\text{pv}}-1}{2}$. Given that training can exhibit phase transitions, the extra few examples can greatly improve the accuracy of the trained example.

Viewed as an interactive proof of learning, the progressive validation technique follows the protocol of figure 10.1.1.

The true error rate of this randomized hypothesis will be:

$$e_{\text{pv}} = \frac{1}{m_{\text{pv}}} \sum_{i=1}^{m_{\text{pv}}} e(h_i)$$

where $e(h_i) = \Pr_D(h_i(x) \neq y)$ and the empirical error estimate of this randomized hypothesis will be:

$$\hat{e}_{\text{pv}} = \frac{1}{m_{\text{pv}}} \sum_{i=1}^{m_{\text{pv}}} \hat{e}_i$$

10.2. Variance Analysis

Bounding the deviation of $\hat{e}_{\text{pv}}$ is more difficult than bounding the deviation of a holdout error. To understand this, we can think of two games, the holdout game and the progressive validation game.

In the holdout game, your opponent chooses a bias and then nature flips m coins with that bias. If the deviation of the average number of heads is larger than ϵ, then you lose. Otherwise, you win.

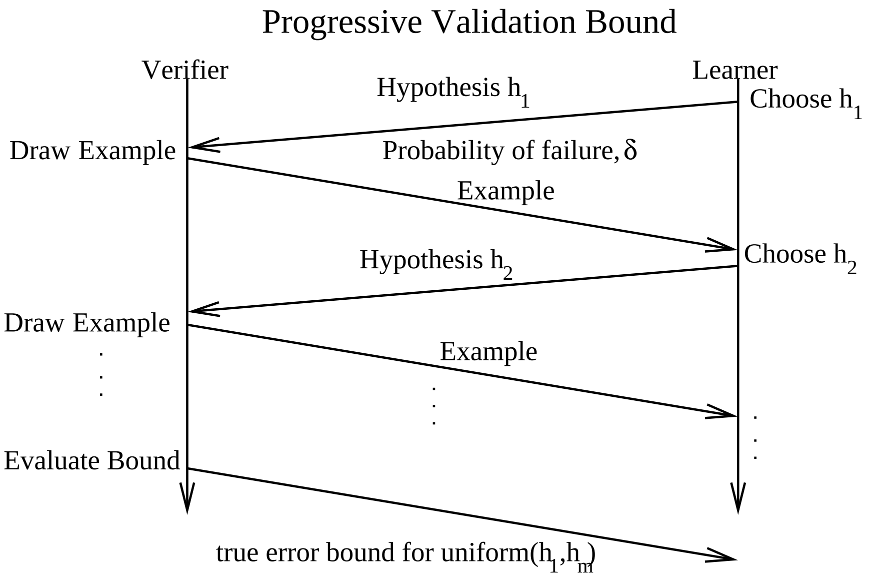


FIGURE 10.1.1. The progressive validation protocol has a learner repeatedly commit to a hypothesis before it is given a new example. Based upon the test errors, a bound on the true error rate of the metahypothesis which chooses randomly from each of $(h_1, ..., h_m)$ before each evaluation is provided.

In the progressive validation game, the opponent chooses the bias of *each* coin just before it is flipped. The goal of the opponent remains the same, and the opponent wins if a large deviation is observed.

The progressive validation opponent is at least as strong as the holdout opponent since the progressive validation opponent could choose the same bias for every coin. Nonetheless, we will see that the progressive validation opponent is *not* much stronger.

There are two ways in which we can show that the progressive validation opponent is not much stronger. The first technique will show that the variance of the progressive validation estimate is smaller than might be expected. Then we will show that the *deviations* of the progressive validation opponent behave much like the deviations of an independent opponent.

THEOREM 10.2.1. *Suppose we test the progressive validation hypothesis on* $m_{test} = m_{pv}$ *additional examples. Let* $\hat{e}_{test}$ *be the empirical error on these examples. Then, we have:*

$$E_{(x,y)^{m_{pv}+m_{test}} \sim D^{m_{pv}+m_{test}}} (\hat{e}_{test} - e_{pv})^2$$

$$\geq E_{(x,y)^{m_{pv}} \sim D^{m_{pv}}} (\hat{e}_{pv} - e_{pv})^2$$

PROOF. Every example on the left hand side can be though of as a coin with bias $e_{\rm pv}$. The variance of the LHS is then $m * e_{\rm pv}(1 - e_{\rm pv})$. The right hand side is:

$$E_{(x,y)^m \sim D^m} (\hat{e}_{\rm pv} - e_{\rm pv})^2$$

$$= E_{(x,y)^m \sim D^m}\left[\sum_{i=1}^{m} \hat{e}_i - e_i\right]^2$$

where $e_i = e(h_i)$

$$= E_{(x,y)^m \sim D^m}\left[\sum_{i \neq j}(\hat{e}_i - e_i)(\hat{e}_j - e_j) + \sum_{i=1}^{m}(\hat{e}_i - e_i)^2\right]$$

$$= \sum_{i \neq j} E_{(x,y)^m \sim D^m}(\hat{e}_i - e_i)(\hat{e}_j - e_j) + \sum_{i=1}^{m} E_{(x,y)^m \sim D^m}(\hat{e}_i - e_i)^2$$

The cross product term is:

$$E_{(x,y)^m \sim D^m}(\hat{e}_i - e_i)(\hat{e}_j - e_j)$$

Without loss of generality, assume that $i < j$. What we wish to prove is that the expected value of this quantity is 0 conditional on the values of all random variables other than ith or jth example. Let S_{ij} be the set of examples minus the ith and jth example. Also, let z_i and z_j be the ith and jth labeled examples. If we can show that:

$$\forall S_{ij} : \; E_{z_i, z_j | S_{ij}}(\hat{e}_i - e_i)(\hat{e}_j - e_j) = 0$$

then linearity of expectation will imply that:

$$E_{(x,y)^m \sim D^m}(\hat{e}_i - e_i)(\hat{e}_j - e_j) = 0$$

The value of e_i is fixed after conditioning on S_{ij} (and assuming a deterministic learning algorithm) while the value of e_j is not fixed: it is dependent on the random variable z_i. Let $e_j(z_i)$ be the derived random variable. Then, the expectation is:

$$E_{z_i, z_j | S_{ij}}(\hat{e}_i - e_i)(\hat{e}_j - e_j)$$

(implicitly conditioning on S_{ij})

$$= \sum_{z_i, z_j} \Pr_D(z_i) \Pr_D(z_j)(\hat{e}_i(z_i) - e_i)(\hat{e}_j(z_j, z_i) - e_j(z_i))$$

$$= \sum_{z_i} \Pr_D(z_i)(\hat{e}_i(z_i) - e_i) \sum_{z_j} \Pr_D(z_j)(\hat{e}_j(z_j, z_i) - e_j(z_i))$$

For any fixed z_i, we want to show that $\sum_{z_j} \Pr_D(z_j)(\hat{e}_j(z_j, z_i) - e_j(z_i)) = 0$.With z_i fixed, the true error rate of the jth hypothesis is $e_j(z_i)$. Therefore, the probability of observing an error ($\hat{e}_j(z_j, z_i) = 1$) is $e_j(z_i)$, and the probability of observing no error ($\hat{e}_j(z_j, z_i) = 0$) is $1 - e_j(z_i)$. This implies:

$$\sum_{z_j} \Pr_D(z_j)(\hat{e}_j(z_j, z_i) - e_j(z_i))$$

$$= \sum_{z_j : \hat{e}_j(z_j, z_i) = 0} \Pr_D(z_j)(-e_j(z_i)) + \sum_{z_j : \hat{e}_j(z_j, z_i) = 1} \Pr_D(z_j)(1 - e_j(z_i))$$

$$= (1 - e_j(z_i))(-e_j(z_i)) + e_j(z_i)(1 - e_j(z_i))$$

$$= 0$$

Putting this together, we get:

$$\text{cross term} = \sum_{z_i} \Pr_D(z_i)(\hat{e}_i(z_i) - e_i) * 0$$

$$= 0$$

Since this sum is zero regardless of the values of all other random variables, the cross product terms all have expectation zero. Note that we can extend this proof to randomized algorithms by conditioning on the random bits of the algorithm in the above argument. Consequently:

$$= \sum_{i=1}^{m} E_{(x,y)^m \sim D^m} \left[(\hat{e}_i - e_i)^2 \right]$$

$$= \sum_{i=1}^{m} e_i(1 - e_i)$$

So, all that we must show is:

$$m e_{\text{pv}}(1 - e_{\text{pv}}) \geq \sum_{i=1}^{m} e_i(1 - e_i)$$

which follows from Jensen's inequality and the convexity of $x(1-x)$ on the interval $[0, 1]$. □

10.3. Deviation Analysis

The second way in which we can show that the progressive validation opponent is not so strong is by bounding the deviation directly. Surprisingly, we can prove exactly the same bound as for Hoeffding inequality.

THEOREM 10.3.1.

$$\Pr_{(x,y)^m \sim D^m} (e_{pv} \geq \hat{e}_{pv} + \epsilon) \leq e^{-2m\epsilon^2}$$

PROOF. (A variant of Chernoff)

$$\Pr_{(x,y)^m \sim D^m} (e_{\text{pv}} \geq \hat{e}_{\text{pv}} + \epsilon)$$

$$\Leftrightarrow \forall \lambda \quad \Pr_{(x,y)^m \sim D^m} (e^{\lambda m e_{\text{pv}}} \geq e^{\lambda m (\hat{e}_{\text{pv}} + \epsilon)})$$

$$\Leftrightarrow \Pr_{(x,y)^m \sim D^m} (e^{\lambda m (e_{\text{pv}} - \hat{e}_{\text{pv}} - \epsilon)} \geq 1)$$

$$\leq E_{(x,y)^m \sim D^m} e^{\lambda m (e_{\text{pv}} - \hat{e}_{\text{pv}} - \epsilon)}$$

$$= E_{(x,y)^m \sim D^m} e^{\lambda (\sum_{i=1}^{m} e_i - \hat{e}_i - \epsilon)}$$

$$= \prod_{i=1}^{m} E_{(x,y) \sim D} \left[e^{\lambda (e_i - \hat{e}_i - \epsilon)} | \hat{e}_1, ..., \hat{e}_{i-1} \right]$$

The value of $e^{\lambda(e_i - \hat{e}_i - \epsilon)}$ is only dependent on $\hat{e}_1, ..., \hat{e}_{i-1}$ through the value of e_i. For all possible hypotheses, h_i, we know that $\hat{e}_i$ is a Bernoulli random variable with bias $e_i = \Pr_D(h_i(x) \neq y)$. This is true regardless of how we choose h_i. Consequently, we have:

$$= \prod_{i=1}^{m} E_{(x,y) \sim D} \left[e^{\lambda (e_i - \hat{e}_i - \epsilon)} \right] \tag{10.3.1}$$

Since this is true for all λ, we can choose the optimal λ. In general, this is an optimization problem which can be worst-case simplified with the inequality:

$$E_{(x,y)\sim D} e^{\lambda(e_i-\hat{e}_i)} \leq e^{\frac{\lambda^2}{8}}$$

which implies:

$$\forall\lambda \ \leq \prod_{i=1}^{m} e^{\frac{\lambda^2}{8}-\lambda\epsilon)}$$

This is optimized for $\frac{2\lambda}{8} = \epsilon$ which implies $\lambda = 4\epsilon$ giving:

$$\leq \prod_{i=1}^{m} e^{-2\epsilon^2}$$

$$= e^{-2m\epsilon^2}$$

□

PROBLEM 10.3.2. (Open) Starting with equation 10.3.1, derive a bound similar to the relative entropy Chernoff bound which holds for progressive validation.

10.4. A Quick Experiment

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

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

In the first test, we trained on $m - 10$ examples and tested on 10 examples. In the second test, we trained on $m - 10$ examples and applied progressive validation to the next 10 examples. We repeated this experiment 1000 times for $10 \leq m \leq 30$ and averaged the results in order to get an empirical estimate of the true error of all hypotheses produced, shown in Figure 10.4.1.

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

In order to compare the quality of error estimation, we did another set of runs calculating the error discrepancy |true error−estimated error|. Five training

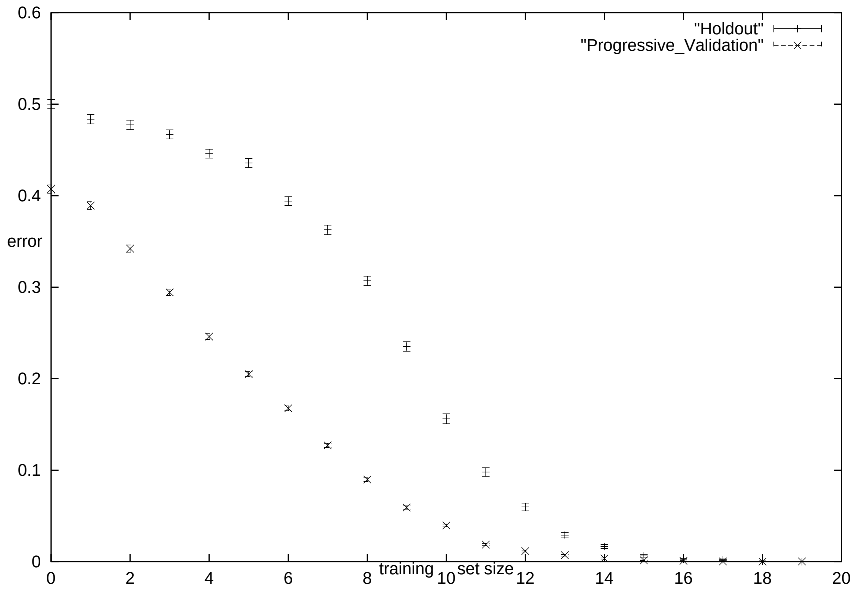


FIGURE 10.4.1. True error vs. training size for hold-out and progressive validation. Error bars in the figure are using computed by fitting a gaussian to the empirical mean and variance and are at one standard deviation.

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

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

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

10.5. Conclusion

Progressive Validation provides an improvement over the simple technique of using a holdout set by allowing almost half (on average) of the test set examples to be used in training without loosening the resulting bound *when* we compare to the Hoeffding bound for a holdout set. In practice, limiting ourselves to using the Hoeffding bound for holdout sets is not acceptable so progressive validation may not (in practice) have a better bound than the holdout bound. For this reason, progressive validation is not compared with other bound calculations in later experiments.

Tightening the analysis to statements about Binomial tail bounds is a significant open problem.

CHAPTER 11

Combining sample complexity and holdout bounds

11.1. Combination Possibilities

We have two forms of bound, one which uses training set errors and one which uses holdout set errors. The obvious question to ask is: can we combine the information from both bounds? Presumably, if we use both the training error and the test error, we should be able to construct a better confidence interval for the location of the true error rate.

Viewed as an interactive proof of learning (see figure 11.1.1), the train and test approach will just add an extra testing phase to every training set based bound.

Given a fixed hypothesis and learning problem, we know that the test error will be Binomially distributed. Given a fixed learning algorithm and learning problem, the training error will have a considerably more complicated distribution. We can nonetheless, regard the training error as a fixed random variable which has some cumulative distribution parameterized by many parameters, one of which is the true error rate of the output hypothesis (which is itself a random variable).

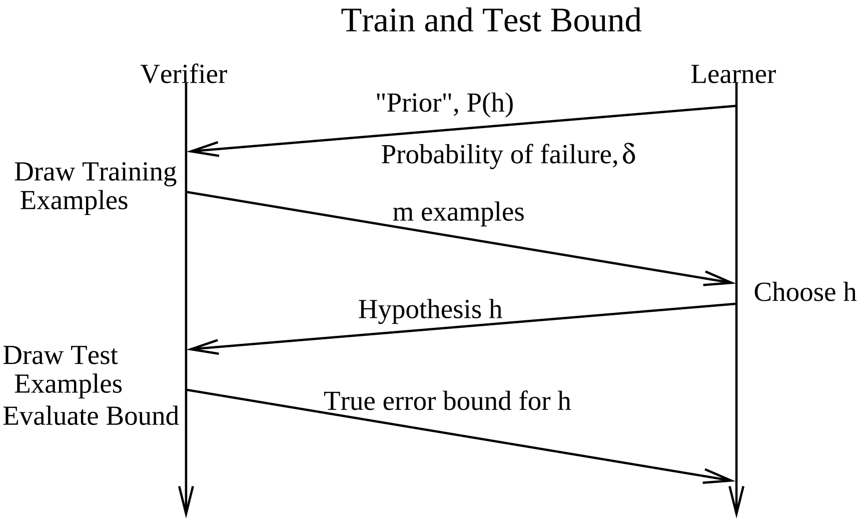
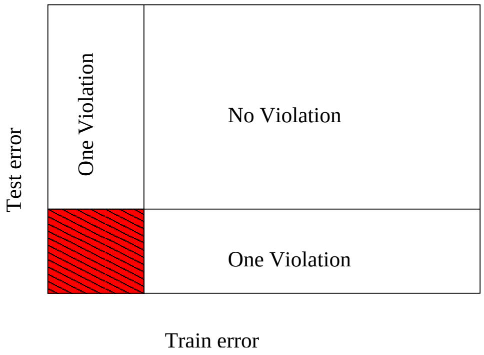


FIGURE 11.1.1. The train and test protocol starts with the learner committing to a "prior" $p(h)$, receiving training examples, choosing a hypothesis, and then evaluating on test examples. The train and test approach can be composed with PAC-Bayes bounds (theorem 6.2.1) as well which is not illustrated here.

How can we construct a confidence interval based upon information from both the training and testing sets? There are several possibilities.

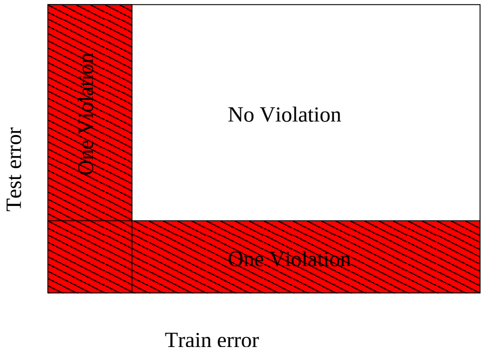
(1) Construct an interval based upon the probability that *both* lower bounds are violated.
(2) Construct an interval based upon the probability that at most one of the lower bounds is violated.
(3) Something else?

Technique (1) can be seen visually by graphing training error vs test error and marking the regions that are bounded away.



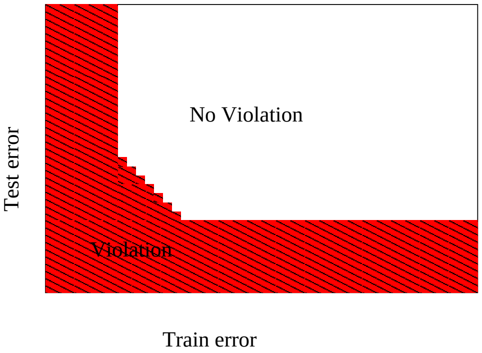
The essential problem with technique (1) is that the resulting true error bound takes the *maximum* (minus a small amount) of the bounds based upon both the test set and the training set. Given that we don't trust either bound to always return tight information, we expect the maximum will not behave well.

Technique (2) can be seen visually in a similar way:



Technique (2) works moderately well. Mathematically, we can calculate the minimum of the two error bounds and add a small amount. This approach is equivalent to taking a union bound. While this approach allows us to combine the bounds, it does not let us achieve an improvement over either which is intuitively possible. Certainly, if we use two test sets, we expect to construct improved confidence intervals.

A better approach may be possible. We would like to construct a rejection region of the following form:



Such a rejection region has two important properties:

(1) If one bound is loose, it does not greatly harm the final true error bound.
(2) The final true error bound can be tighter than either individual true error bound.

Showing that technique (2) works is just an application of the union bound. Given any two bounds on the true error rate, we can apportion $\frac{\delta}{2}$ confidence to each bound. Then both bounds will hold with probability δ which implies that the smaller of the two true error bounds holds.

11.2. General Approaches for Combined Bounds

Showing that a more general technique works must start with a discussion of confidence intervals. Fundamentally, a bound can be viewed as a set of outcomes. Let X be space of outcomes, then a bound $\phi \subseteq X$ is a subset. The probability that this generalized bound is violated is given by:

$$\Pr_{x \sim P}(x \in \phi)$$

Typically, we parameterize ϕ with both P and δ to get:

$$\Pr_{x \sim P}(x \in \phi_P(\delta)) \leq \delta$$

We can expand the definition of a high probability set to include a *randomized* high probability set. In particular, let $\phi_P(w, \delta)$ satisfy:

$$\forall w : \Pr_{x \sim P}(x \in \phi_P(w, \delta)) \leq \delta$$

Then, statement such as:

$$\Pr_{w \sim Q, x \sim P}(x \in \phi_P(w, \delta)) \leq \delta$$

In fact, we can make a stronger statement. If

$$E_{w \sim Q} \Pr_{x \sim P}(x \in \phi_P(w, \delta)) \leq \delta \tag{11.2.1}$$

then

$$\Pr_{w \sim Q, x \sim P}(x \in \phi_P(w, \delta)) \leq \delta \tag{11.2.2}$$

.

Randomized confidence intervals are useful here because we can regard the the draw of the "test" set as constructing a randomized interval for the "training" set.

As long as constraint 11.2.1 is obeyed, the bound will hold with probability at least δ. A P-value Inversion of the bound will then yield the following theorem:

LEMMA 11.2.1. *(Exact test and train bound) Let S_{test} be the set of test examples and S be the set of training examples. Let $\phi_D(\delta)$ be any bound satisfying $\Pr_{S \in D^m}(S \in \phi_D(\delta)) \leq \delta$. Let $f(S_{test})$ be any function satisfying: $E_{S_{test} \sim D^{m_{test}}} \Pr_{S \sim D^m}(S \in \phi_D(f(S_{test}) * \delta)) \leq \delta$, then:*

$$\Pr_{S, S_{test} \sim D^{m+m_{test}}} (S \in \phi_D(f(S_{test}) * \delta)) \leq \delta$$

PROOF. Linearity of expectation. □

There are many possible choices of the function $f(S_{\text{test}})$, each of which leads a different combined training and testing bound. Typically, it will be important to invert this bound into a P-value form where we take a worst case over all D just as in section 3.4.

The functional form of $f(S_{\text{test}})$ becomes more constrained when we only work with a bound on the test error cumulative distribution rather than the exact distribution.

11.3. Approximations in Combinations

The inexact nature of bounds forces us to impose a monotonic structure on the function $f(S_{\text{test}})$. For simplicity, we will restrict to functions of the form $f(\hat{e}_{\text{test}}(h))$ where $\hat{e}_{\text{test}}(h)$ is the test error on hypothesis h. This simplification is not necessary and this technique can be extended to arbitrary test set based techniques.

We can consider any upper bound $\theta_D(\delta)$ on the true error rate, $e_D(h)$, as inducing a cumulative distribution on the test set events.

This cumulative distribution is *not* the cumulative distribution of the underlying (Binomial) probability. To construct this distribution, let:

$$F_\theta(\hat{e}_{\text{test}}(h)) = \inf\{\delta : \hat{e}_{\text{test}}(h) \in \theta_D(\delta)\}$$

Intuitively, $F_\theta(\hat{e}_{\text{test}}(h))$ is the smallest δ such that the test error $\hat{e}_{\text{test}}(h)$ is rejected.

LEMMA 11.3.1. *The function $F_\theta(\hat{e}_{test}(h))$ is a cumulative distribution function.*

PROOF. In order to show that the function is a cumulative distribution function, we must show that it varies between 0 and 1 for all values of $\hat{e}_{\text{test}}(h)$. Since $\theta_D(\delta)$ is an upper bound, the following inequality holds:

$$\forall S_{\text{test}} \; F_\theta(\hat{e}_{\text{test}}(h)) \geq \text{Bin}(m, m * \hat{e}_{\text{test}}(h), e_D(h))$$

This inequality implies the value of $F_\theta(\hat{e}_{\text{test}}(h))$ is always at least as large as the CDF of the underlying Binomial distribution. Note, that different S_{test} are implicitly aliased under this technique. We also have the inequality $F_\theta(\hat{e}_{\text{test}}(h)) \leq 1$ because all true error rate upper bounds are vacuous above a true error rate bound of 1. □

We have shown that F_θ is a cumulative distribution function over the value of the empirical error. Given the upper bound cumulative, F_θ, we can look at distributions satisfying:

$$E_{\hat{e}_{\text{test}}(h) \sim F_\theta} \Pr_{S \sim D^m} (S \in \phi(f(\hat{e}_{\text{test}}(h)) * \delta)) \leq \delta$$

If we are guaranteed that $f(\hat{e}_{\text{test}}(h))$ decreases monotonically then equation 11.2.2 will hold. This is the essence of our theorem.

THEOREM 11.3.2. *(Approximate test and train bound) Let $\phi_D(\delta)$ be any bound satisfying* $\Pr_{S\sim D^m}(S \in \phi_D(\delta)) \le \delta$. *Let $f(\hat{e}_{test}(h))$ be any monotonic decreasing function satisfying:*

$$E_{\hat{e}_{test}(h)\sim F_\theta} \Pr_{S\sim D^m}(S \in \phi_D(f(\hat{e}_{test}(h)) * \delta)) \le \delta$$

, then:

$$\Pr_{S,S_{test}\sim D^{m+m_{test}}}(S \in \phi_D(f(\hat{e}_{test}(h)) * \delta)) \le \delta$$

PROOF. Note that $\Pr_{S\sim D^m}(S \in \phi_D(f(\hat{e}_{\text{test}}(h)) * \delta))$ is a monotonic decreasing function of $\hat{e}_{\text{test}}(h)$. For any monotonic decreasing function $g(x)$ and any two cumulative distribution functions $F_1(x)$ and $F_2(x)$ satisfying $\forall x\ F_1(x) \le F_2(x)$ we have:

$$E_{x\sim F_1(x)}g(x) \le E_{x\sim F_2(x)}g(x)$$

Let $F(x)$ be the cumulative distribution of the Binomial and note that the definition of a bound implies: $\forall x\ F_\theta(x) \ge F(x)$. Applying these inequalities, we get:

$$\delta \ge E_{\hat{e}_{\text{test}}\sim F_\theta} \Pr_{S\sim D^m}(S \in \phi_D(f(\hat{e}_{\text{test}}(h)) * \delta))$$
$$\ge E_{\hat{e}_{\text{test}}\sim F} \Pr_{S\sim D^m}(S \in \phi_D(f(\hat{e}_{\text{test}}(h)) * \delta))$$

Given this, an application of theorem 11.2.1 completes the proof. □

The only constraint that we must check in applying a combined training and test set bound is the monotonicity constraint. Heuristically, this is satisfied for the functions graphed in the pictures of techniques (1)-(3) because the set of excluded events increases monotonically along the x axis as it decreases along the y axis.

An explicit mathematical form for technique (3) can be given by considering the bound-based cumulative distributions, F_θ, and a similar distribution for the training set, F_ϕ. In particular, we can define the rejection region to be $\{S_{\text{test}}, S : F_\theta F_\phi \le t(\delta)\}$ where $t(\delta)$ is a function satisfying $\Pr_{S_{\text{test}},S\sim F_\theta,F_\phi}(F_\theta(\hat{e}_{\text{test}}(h))F_\phi(S)) \le t) \le \delta$. The monotonic constraint is satisfied by this construction because F_θ is implicitly monotonic decreasing with $F_\phi(S)$ given a constant t. We will use technique (3) for combining training and test sets in the experiments. Appendix section 16.4 details the programming interface to calculate this bound.

11.4. Conclusion

The theorems in this section give us a very general tools for composing training set and test set based bounds. We used these general tools to construct a particular approach which will be tested in the next chapter.

There are two significant questions which need to be answered.

(1) Is the improvement of the particular approach worthwhile over a simple disjunction (technique 2 in the introduction)? Later empirical results will show that it can be worthwhile on real data, but the computational cost is non-negligible.
(2) Is the approach we used for combining bounds "best"? It is difficult to compare different approaches because strict dominance does not typically occur. Nonetheless, there are many other ways to compose train set and test set bounds and satisfy theorem 11.3.2. Perhaps some other way is better on natural learning problems.

Part 3

Experimental Results

CHAPTER 12

Decision Trees

Are Sample Complexity bounds quantitatively tight? The real challenge lies with continuous valued classifiers and will be addressed in the next chapter. Before worrying about continuous valued classifiers it is worthwhile to consider performance on a discrete valued classifier. To do this, we will apply a (discrete valued) decision tree to learning problems in the UCI machine learning database.

The results of this analysis are interesting - competitive bounds are achieved in some cases and the best sample complexity bounds are never more than an order of magnitude worse than a reasonable holdout based approach using the same resources.

Most practitioners of applied machine learning currently use a different technique for free parameter optimization: holdout sets. The simplest form of this technique is to separate the examples into a "training set" and "testing set". The training set is used by the learning algorithm to output a hypothesis. The hypothesis is then tested on the test set to generate an estimate of the future error rate. The principal advantage of the holdout technique is the (simple theoretical) guarantee that with high probability the estimate will not be much higher or lower than the true error rate. Here the quantity "much" depends upon the size of the holdout set and the "high probability" can be chosen as desired by the person applying the bound.

The principal disadvantage of the holdout approach is that not all of the examples are used for training. This is often not a significant problem but it *can* be important for certain learning algorithms and problems which exhibit phase transitions (see figure 10.4.1 on page 88 for an example). Near a phase transition , extra examples can exponentially decrease the expected error rate of the output hypothesis. More sophisticated techniques such as Leave One Out Cross Validation, K-fold Cross Validation, and Progressive Validation [**3**] attempt to remove this disadvantage. Each of these alternative holdout techniques cannot fully remove the disadvantage and some of them threaten the advantage (a tight bound on the true error). In addition, a new disadvantage is introduced: significantly more computation is required.

The holdout technique is fundamentally unsatisfactory for one important reason: If a holdout set is used multiple times, the theoretical guarantees become progressively weaker. In particular, this implies that designing a learning algorithm which makes multiple internal decisions based upon the result of evaluating a future error bound for holdout sets may require many examples. In fact, enough examples are required that a "multiply used holdout set" is described by the same math as a training set based sample complexity bound.

We will compare the following bounds on a decision tree:

(1) The discrete hypothesis bound 4.2.1.

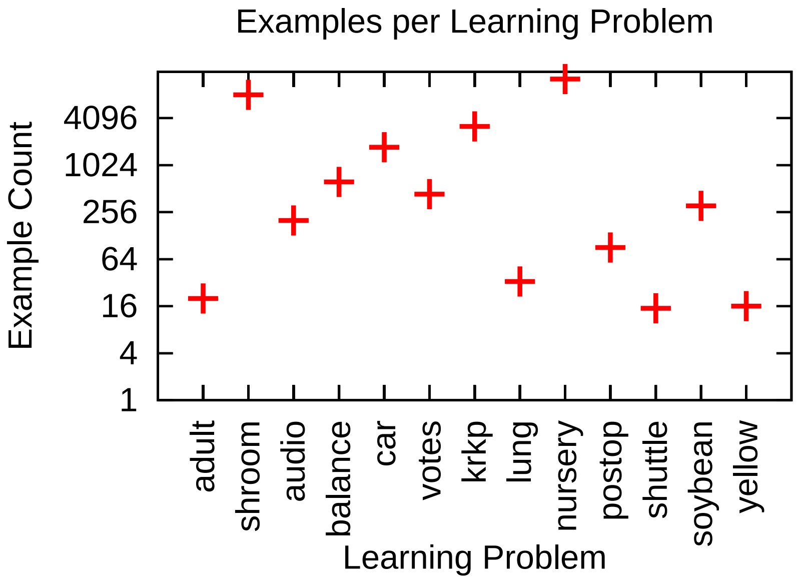


FIGURE 12.1.1. A plot of the sizes over various learning problems in terms of the number of examples. The learning problems used range over 3 orders of magnitude in size.

(2) The Occam's Razor bound 4.6.1 with the Hoeffding Binomial tail bound. This bound might be thought of as an old "state of the art" bound.
(3) The Microchoice bound 5.2.2. In addition, we will prune the decision tree according to the microchoice bound.
(4) The Shell bound 8.1.2 and the Sampled Shell bound 8.2.1.
(5) A simple holdout bound 4.1.1 using a holdout set of size 20%.
(6) A combined training and testing bound using the holdout bound 4.1.1 and microchoice bound 5.2.2.
(7) A combined training and testing bound using the holdout bound 4.1.1 and the (stochastic) shell bound (8.2.1) 8.1.2.

We will first discuss the decision tree and bound calculation implementation. Then present results

12.1. The Decision Tree Learning Algorithm

The bounds are applied to the results of decision trees learned on UCI database problems (see figure 12.1.1). The decision tree algorithm is a variant of ID3 which works in two phases: a tree of minimum empirical error is discovered and then pruned with a criteria which arises naturally from the Microchoice bound. The first phase works in a recursive fashion by maximizing the information gain across all splitting criteria. It is assumed that each example consists of n features each of which takes a small number of discrete values. The splitting criteria at each internal node are of the form "feature f has value v" which implies that a binary tree is produced. Leaves of the tree are labeled with the label most common amongst the examples in the training set which reach the leaf. The pruning pass prunes according to the criterion: minimize the upper bound on true error implied by the Microchoice bound. Pruning starts with the internal nodes closest to the leaves and proceeds up the tree toward the root node. This pruning criteria falls in the category of pessimistic criteria [**38**]. The implementation has been somewhat computationally optimized by careful caching of information. The examples are tokenized into integers for fast comparison and analysis of splitting criteria at each node only loops over the examples once. Sub-calculations of the Microchoice bound are cached for use in pruning.

12.1.1. Pruning. The proof of the Microchoice bound works by assigning a uniform weight to each choice in a choice space. Since the choice space of every node includes the choice of making a leaf, the Microchoice bound incidentally also proves a tighter bound for every pruning of the output tree. We choose to prune when pruning reduces the upper bound on the generalization error. We prune starting with internal nodes nearest to the leaves and working toward the root node. As will be shown by the experiments, other bounds are sometimes tighter than the Microchoice bound. This implies that the Microchoice bound is not always the optimal pruning criteria. However, the Microchoice bound is fast to calculate so it is used here. In practice, it may be desirable to prune according to the criteria "minimize $\alpha\hat{e}_S(h) + (1-\alpha)be(h)$" where $\bar{e}(h)$ is some high probability upper bound on the true error $e_D(h)$. We use $\alpha = 0$ with the bound produced by the Microchoice bound.

12.1.2. Uniform Sampling from Decision Trees. To use the Sampling Shell bound with Structural Risk Minimization, we need to sample uniformly from the set of all decision trees of a particular size. The number of binary structures of size i is the ith Catalan number, $C_i = \frac{1}{i+1}\binom{2i}{i}$. This implies there are C_i different tree structures with i internal nodes. Sampling uniformly from the set of all tree structures with i internal nodes can then be done with a recursive process. For a particular node in a tree, assume that j internal nodes need to be created. If $j = 0$, we have a leaf. For $j \geq 1$, we can construct a distribution over the number of nodes that are left branches of the tree by calculating $C_0, \ldots, C_{j-1}$ and normalizing. Draw from this distribution to get j_l: the number of internal nodes in the left sub-tree. Let $j_r = j - j_l - 1$ be the number of internal nodes in the right sub-tree and recurse. This construction is not yet a decision tree because we have not placed tests in each node. We make another uniform pick from the set of available tests at a particular node. We allow only tests on features with at least two different feature values remaining after tests by parent nodes. This approach is not quite uniform in the set of decision trees because when there are i internal nodes and $F < i$ Boolean features, some trees have a depth greater than F which is not possible in a binary decision tree. In practice this does not make a difference because the number of decision trees with a too-large depth is an exponentially small fraction of the total number of trees. When running the algorithm we did not ever encounter a sampled binary tree structure with depth greater than F. Leaves have a label picked uniformly from the set of labels. A particular label implies some empirical error count amongst all examples which reach the leaf. By adding the leaf error rates together we find the empirical error of a random hypothesis from the hypothesis set.

12.1.3. Fast Sampling. In order for the Sampling Shell bound to be a significant improvement we need the number of samples l to satisfy $l = O(m)$. This is not tractable using the above sampling technique so we "cheated". At some critical sub-tree size s we switch from sampling to exact enumeration. (The exact value of s is discussed later.) Exact enumeration produces a multiset with elements corresponding to an error rate $\hat{e}_i$ in the sub-tree and a count c_i associated with each element. We recurse up the tree with the entire error multiset rather than just one error. At an internal node we have a multiset of errors from the left child and from the right child. The multiset of errors at an internal node is the cross product of all errors in the left child and right child because the choice of left sub-tree is independent of the choice of right sub-tree. Let S_l be the multiset of errors in the left child and S_r be the multiset of errors in the right child. The multiset of errors for the internal node is then given by:

$$\{\hat{e}_i + \hat{e}_j, c_i * c_j : \hat{e}_i, c_i \in S_l, \hat{e}_j, c_j \in S_r\}$$

The multiset produced is passed recursively to the parent and each element of the set of errors at the root node is considered an "independent" sample for the purposes of applying the Sampling Shell bound.

The power of this techniques comes from the fact that the set of hypotheses sampled is exponential in the number of leaves. However, we never need to represent the exponentially many different hypotheses explicitly because there are only $m+1$ possible empirical errors. Using multisets, we achieve sampling time similar to the simple uniform sampling approach of the previous section but with exponentially more (dependent) samples. The drawback to this approach is that the samples are no longer independent so it is not "fair" to use them as independent samples in a theoretical sense. We pretend each fast-sample is independent anyway and apply the Sampling Shell bound. It is worth noting that the samples returned by this technique are not biased and sometimes have lower variance than independent samples.

The choice of the critical subtree size s is done in an anytime fashion. The time t required for learning the decision tree is first calculated. Then, starting with a size of $s = 0$, we sample from the decision tree, increasing the size by one after each sample. When more than $t/10$ time has been spent on sampling, we cease incrementing s. If s is the size of the tree, then we stop and apply the exact Shell bound. Otherwise, we decrement s and sample repeatedly until as much time has been spent on sampling as was spent on learning the decision tree. The union of all the multisets is returned and the Sampling Shell bound is used.

12.2. Bound Application Details

Before introducing the results, we will mention a few important details about this bound.

12.2.1. Structural Risk Minimization. A naive application of the Shell bound would not prove useful because the size of the hypothesis space can be extremely large. Instead, we must combine it with Structural Risk Minimization (SRM) to achieve useful results. In SRM, you start with a bound for each hypothesis space, H_i, in a sequence of nested hypothesis spaces, $H_1 \subseteq \cdots \subseteq H_n$. These bounds on individual hypothesis spaces are combined to create a bound which applies for all hypothesis spaces. The particular nesting we use is "H_i = all decision trees with i or fewer internal nodes".

Since the size of the decision tree hypothesis spaces increases exponentially with the index i , we choose $p_i = \frac{1}{2^i}$. This choice has the property that $\frac{1}{p_i}$ is always small in comparison to $|H_i|$, implying that the SRM bound is never much worse than a simple application of the underlying bound.

12.2.2. Computation. The computational cost of calculating some of these bounds is nontrivial. There are two basic parts to this computation;

(1) Gathering the information in order to calculate the bound. This is sometimes infeasible for shell bounds and (later) PAC-Bayes bounds.
(2) Combining the information in order to calculate the bound. For the shell bound, the amount of computation is like $O(m^{1.5})$ where m is the number of training examples. For the combined test and shell bound, the computation is approximately $O(m^2)$.

We typically avoid the difficulties inherent in (1) using tricks such as monte carlo sampling followed by bounding the deviation of the monte carlo sample. In particular, we use the anytime computation trick of section 12.1.3 here.

To avoid difficulties inherent in problem (2), we use fast bounds (see section 3.2) on the Binomial tail as necessary.

12.3. Results & Discussion

12.3.1. Holdout bound. Our goal is to bound the true error of the hypothesis output by our learning algorithm. To do this, we apply sample complexity bounds to the results of the decision tree on UCI database problems. The problems chosen from the UCI database are those for which a discrete decision tree is applicable. All bounds are calculated with a probability of failure of $\delta = 0.1$. As mentioned in the introduction, there are two approaches. The commonly used approach is to first divide the example set into two sets, m_{test} and m_{train}. Then, train using the examples in m_{train} and test on the m_{test} examples. We chose an 80/20 split of the data into $m_{\text{train}}/m_{\text{test}}$. We will compare each bound with the simple holdout approach because this is the commonly used baseline.

12.3.2. Comparison with a standard confidence interval approach. When attempting to calculate a confidence interval on the true error rate given the holdout set, many people follow a standard statistical prescription:

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

This approach is motivated by the fact that for any *fixed* true error rate, the distribution of empirical errors will behave 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. In particular, 12.3.2 shows that the confidence interval is not confined to the interval $[0, 1]$. It is difficult to give an interpretation to intervals with boundaries less than 0 or greater than 1. In addition, this approach is sometimes highly

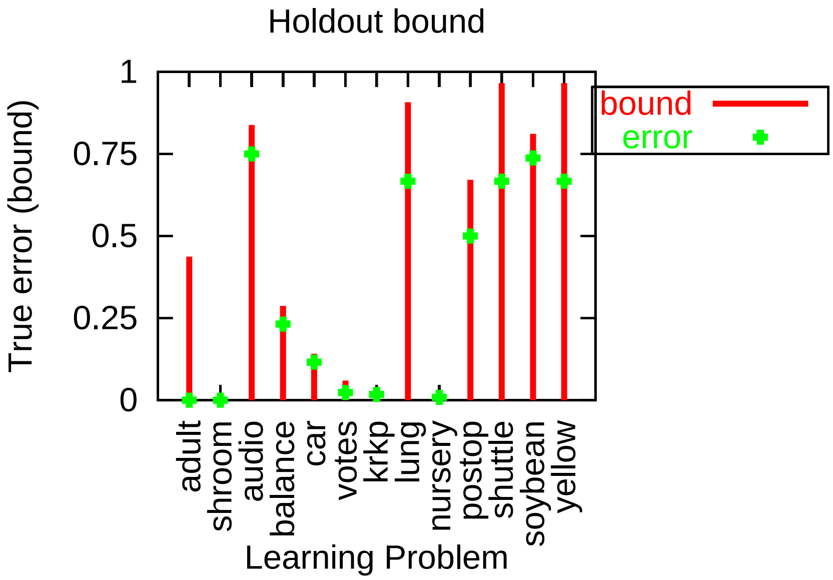


FIGURE 12.3.1. This is a graph of the true error upper bound give by the holdout bound (4.1.1). In this figure (and all others) we use $\delta = 0.1$ probability of failure for the tail. Here "error" is the test error. The red lines can be interpreted as the region where the true error rate might exist given that we are in a high probability case.

overoptimistic. When the test error is 0, our confidence interval should *not* have size 0 for any finite m.

In contrast, the holdout bound approach uses the underlying Binomial distribution directly. This implies:

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

The bootstrap [**15**] is sometimes used as a confidence interval. The assumption under which this works is essentially equivalent to an assumption of "enough" data. For finite amounts of data, the bootstrap "confidence intervals" will necessarily be violated on datasets with phase transitions such as 10.4.1. This is discussed more in the next section.

12.3.3. Comparison with point estimators. Point estimators are techniques for directly estimating the value of the true error. In theory, there should be no need to compare point estimators with confidence interval bounds such as those discussed here because the goals are simply different: point estimators attempt to estimate the value of the true error while confidence intervals confine the value of the true error to an interval with high probability. However, point estimators are often *used* for more than estimating true error. It is a common practice to use

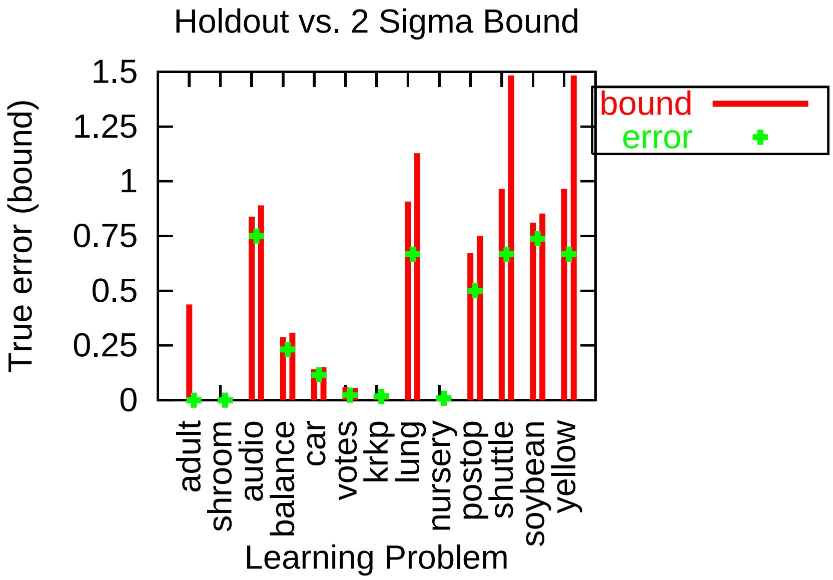


FIGURE 12.3.2. This is a graph of the confidence intervals implied by the holdout bound (4.1.1) on the left, and the approximate confidence intervals implied using the common two sigma rule motivated by asymptotic normality on the right. For this graph only, the upper and lower bounds of the holdout bound have a maximum 2.5% failure rate (each), rather than a 10% failure rate. This is done in order to make the results more comparable with the 2-sigma approach. The holdout bound is better behaved in the sense that the confidence interval is confined to the interval $[0, 1]$ and it is never over-optimistic.

point estimators in deciding which of two learning algorithms (or learning algorithm parameters) is better.

There are several several point estimators in use, including:

(1) Holdout test set error rate.
(2) The bootstrap.

One commonly used point estimator is the bootstrap. In typical use, the bootstrap which functions like this:

Repeat many times:

(1) Pick m examples uniformly from the set of m examples.
(2) Train on the m examples
(3) Test on examples not included in the training set.

After the above computation, the training and test errors are combined according some formula (which often varies) to get an estimate of the true error rate of a hypothesis learned on all of the m original examples.

There is one immediate observation: the resampling process typically results in about $m(1 - \frac{1}{e})$ unique examples being included in the resampled subset. This has very strong implications because there exist learning problems with “phase

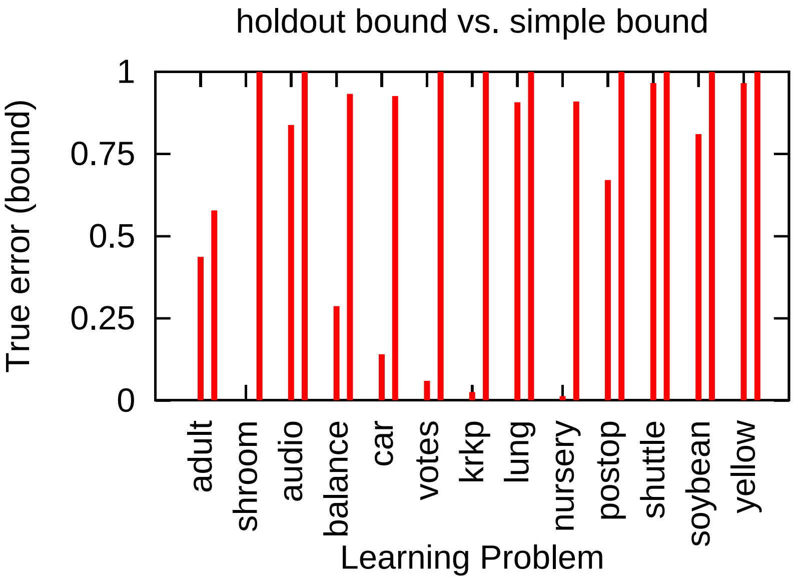


FIGURE 12.3.3. This is a plot comparing confidence intervals built based upon the holdout bound (4.1.1) on the left and the simple training bound using the Hoeffding inequality (4.2.3) on the right. The results are clearly unsatisfactory for the simple training bound.

transitions" where the accuracy of the learned hypothesis (even for the best possible learning algorithm) as a function of m decreases suddenly when m reaches some critical threshold. This implies that point estimators *cannot* always be accurate on dataset with a phase transition like 10.4.1. When learning a hypothesis on $m(1-\frac{1}{e})$ examples results in a true error rate of 0.5, it could be the case that learning on m examples results in a true error rate of 0 or it could be the case that the true error rate will be 0.5.

Given that the bootstrap can sometimes fail to predict the true error rate, reasoning about which algorithm is preferable based upon the bootstrap output is questionable. One alternative to this is reasoning with the criteria:

Pick the learning algorithm with the lower upper bound.

Assuming that examples are independent, this approach can *never* fail arbitrarily badly (with high probability).

There are still questionable issues with this approach such as: "What if the upper bound is not tight?" It *could* be the case that a better learning algorithm has a worse upper bound implying that the worse algorithm will be picked according to this criteria. One solution to this dilemma is to always involve some small amount of holdout examples in your bound calculation. Used judiciously, these holdout examples can guarantee that the bound-based criteria never becomes too loose.

12.3.4. Simplistic bounds vs. the Holdout bound. Figure 12.3.3 compares a bound based upon theorem 4.2.3 and theorem 4.1.1. It is remarkably pessimistic about the prospect of training set based bounds because the confidence intervals are essentially vacuous. This bound can be improved always by using

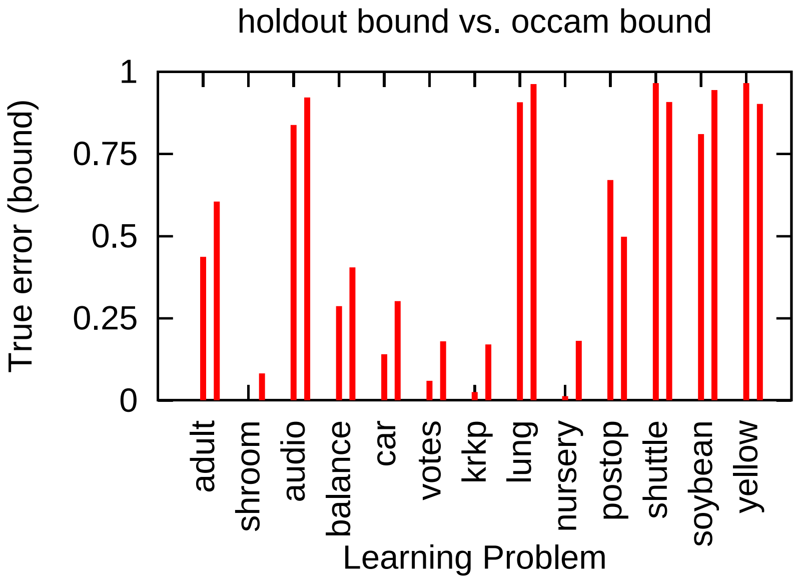


FIGURE 12.3.4. This is a plot comparing confidence intervals built based upon the holdout bound (4.1.1) on the left and an "Occam's Razor" style bound (4.6.1) using the Hoeffding inequality (4.2.3) on the right. This training set bound is sometimes useful on the very small datasets. The particular description length was built using a microchoice-like description language.

exact (rather than approximate) calculations of the Binomial tail. It can also be improved in practice by using a nonuniform "prior" over the hypothesis space.

12.3.5. Occam vs. the Holdout bound. A "prior" (in the sense of theorem 4.6.1) does not help much with the visible confidence intervals, although an examination of the calculations suggests that improvements do exist - they just aren't enough to make the confidence intervals nonvacuous in figure 12.3.4. Note that the "prior" used here is the Microchoice prior. Next, we will get rid of the approximation.

12.3.6. Microchoice. For the first time, we observe confidence intervals which are nonvacuous on a training set in figure 12.3.5. This is encouraging, and a comparison with the holdout approach indicates that the training set based confidence intervals are actually superior on datasets with a small number of examples (and thus with a *very* small holdout set).

12.3.7. Shell Bound. The Shell bound performs better than the Microchoice bound in figure 12.3.6. The information and computation requirements needed to calculate the shell bound are quite large, but the resulting bound is noticeably tighter, especially on problems with more examples. This bound is strong evidence that training set based bounds can be made competitive with test set based bounds. However, it is unnecessary to choose between these approaches since we

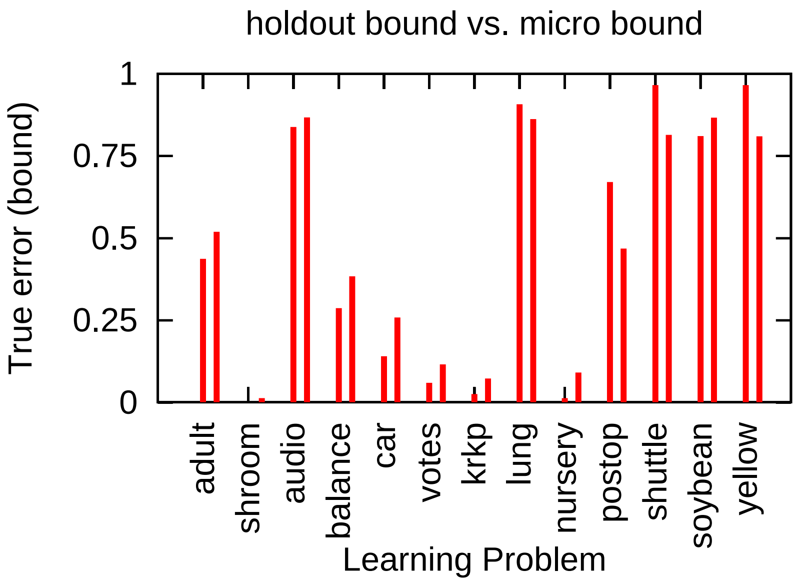


FIGURE 12.3.5. This is a plot comparing confidence intervals built based upon the holdout bound (4.1.1) on the left and the microchoice bound (5.2.2) on the right. The most significant improvement over the last bound is using a Binomial tail bound calculation rather than the Hoeffding approximation. The effect of this improved bound is quite significant when the training error is low. We achieve a tighter upper bound on 4 learning problems.

can construct a bound which uses information from both training and test set based bounds.

12.3.8. Combined Microchoice and holdout bound. The combined Microchoice and Holdout bound performs only slightly worse than the best of the either bound and is sometimes better than either bound individually for the problems reported in figure 12.3.7. This particular combined bound is (perhaps) the most practical result of this thesis since it is easy to calculate the necessary information and reasonably easy to calculate the value of the bound.

12.3.9. Combined Shell and Holdout Bound. The combined shell and holdout bound gives the best results of all in figure 12.3.8. The downside of using the shell bound is that significantly more computation and information is required in order to calculate the bound. The computational cost for the bound is $O(m^2)$ which makes it impractical to apply this bound beyond about $m = 10000$ with current computers.

12.4. Discussion

It is difficult to answer the question “which bound is tighter?” in a theoretical way, because every bound has a worst case. For example, the Occam’s Razor bound is worse than the Simple bound when the hypothesis chosen happens to be one of

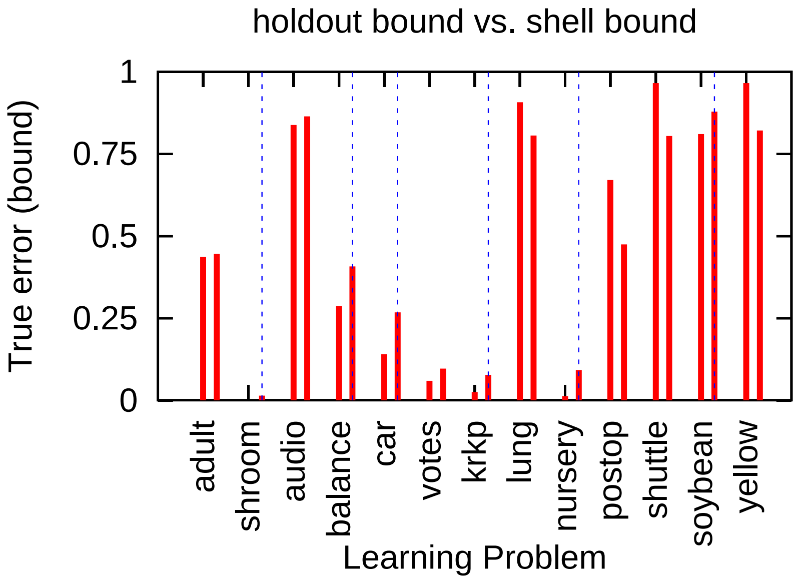


FIGURE 12.3.6. This is a plot comparing confidence intervals built based upon the holdout bound (4.1.1) on the left and the shell upper bound theorem (8.1.2 or 8.2.1) on the right. The light-dashed line indicates which of the results were obtained using the fast sampling technique. The shell-based upper bound is lower than the holdout upper bound on several of the 13 problems and is never significantly looser. Note that the shell bound is computationally intensive with $O(m^{1.5})$ running time plus the time required to gather the necessary information.

the last H. Our results show there is no total ordering amongst the bounds although there is a noticeable rough ordering:

$$\text{Simple} > \text{Occam} > \text{Microchoice} \; > \text{Shell} \simeq \text{Holdout}$$

This ordering is approximately as expected based on theoretical considerations. The Simple bound can never be much better than Occam Bound and the Occam bound can be arbitrarily tighter than the Simple bound. A similar statement holds for the Microchoice Bound and the Shell bound. The Occam bound is only significantly looser than the Microchoice bound because we used the Hoeffding approximation to the Binomial tail. The Shell bound is not always the best, but it does behave well in comparison to the more standard holdout approach.

It is interesting to note that the sampling shell bound is *not* better than the Microchoice bound on these learning problems, even with fast sampling techniques. Apparently, the looseness introduced by bounding the sampling error is not countered by the improvement in tightness.

Empirically, we can observe a very noticeable behavior. For problems with less than 100 examples the sample complexity bounds are superior to the holdout bound. Between 100 and 1000 examples, the behavior changes with the holdout bound

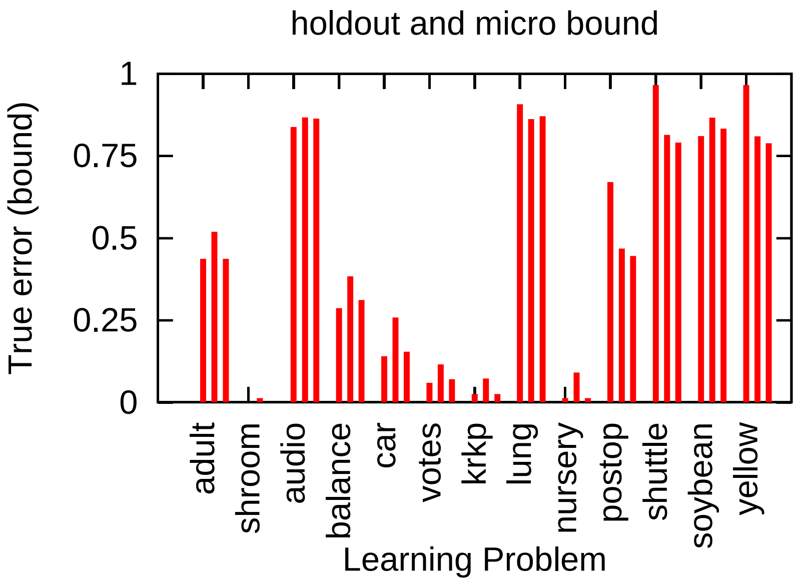


FIGURE 12.3.7. This is a plot comparing confidence intervals built based upon the holdout bound (4.1.1) on the left and a combination (theorem 11.2.1) of the holdout and microchoice (theorem 5.2.2) bounds. The middle column is the microchoice bound and the right column is the combined bound.
The resulting combined bound consistently performs well on all data sets and is sometimes superior to either bound individually. The computational time of this bound is $O(m^{1.5})$ in general.

generally winning, although not necessarily by much. Above 1000 examples, the holdout bound is significantly and consistently tighter than the sample complexity bounds. This behavior strongly suggests that the sample complexity bounds are loose. Each of these bounds is "tight" in one sense or another, but there may exist some as yet undiscovered observable property prevalent in practical machine learning algorithms which allows us to create a tighter bound. In particular, the problem of correlated hypotheses has yet to be solved in a convincing manner.

Also note that the holdout bound is *not* the tightest bound we report. In general, we have the following ordering:

$$\text{Holdout} > \text{Holdout} + \text{Micro} > \text{Holdout} + \text{Shell}$$

The combined bounds seem to have the best behavior in practice.

There are several directions of future investigation which could further strengthen any of these approaches. For the sample complexity approach, it would be useful to address the non-independence of samples in the fast sampling method used for the Sampling Shell bound. We tested the simplest of holdout techniques so another natural extension is to test other holdout techniques. This was not done here, because the theory of these other techniques is lacking.

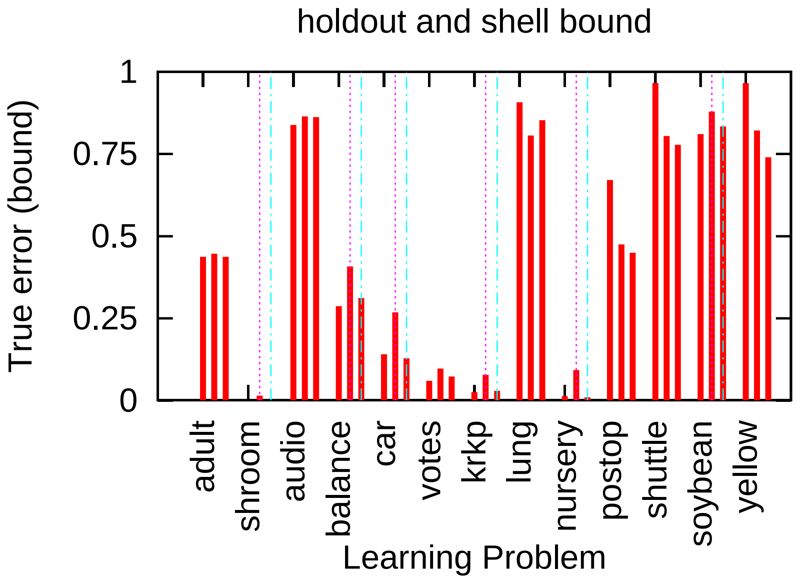


FIGURE 12.3.8. This is a plot comparing confidence intervals built based upon the holdout bound (4.1.1) on the left and a combination (theorem 11.2.1) of the holdout (theorem 4.1.1) and shell bounds (theorem 8.1.2 and 8.2.1). The middle column is the shell bound or sampling shell bound (if a dashed line is present). The right column is the combined bound

The computational cost of this bound is very nontrivial taking $O(m^2)$ time in general.

PROBLEM 12.4.1. (Open) Address the looseness introduced by hypotheses with a strong correlation. For example, two decision trees which differ in only one leaf probably don't have significantly different error rates. Using the union bound over these decision trees introduces unnecessary slack. Note that VC dimension and covering number analysis address this, but (unfortunately) the formulas are either unevaluatable or introduce so much slack that the quantitative results are worse rather than better.

CHAPTER 13

Neural Networks

This work is joint with Rich Caruana and was published at NIPS [**32**].

Estimating the true error rate of a continuous valued classifier can be surprisingly difficult. For example, all known bounds on the true error rate of artificial neural networks tend to be extremely loose and often result in the meaningless bound of "always err" (error rate = 1.0). Figure 13.2.1 demonstrates this.

The approach here is to *not* bound the true error rate of a neural network. Instead, we bound the true error rate of a related distribution over neural networks which we create by analyzing one neural network. The stochastic bound approach proves much more fruitful than trying to bound the true error rate of an individual network. The best current approaches [**?**][**28**] often require 1000, 10000, or more examples before producing a nontrivial bound on the true error rate. We produce nontrivial bounds on the true error rate of a stochastic neural network with less than 100 examples.

Our approach uses a PAC-Bayes bound such as theorem (6.2.1). The approach can be thought of as a redivision of the work between the experimenter and the theoretician: we make the experimenter work harder so that the theoretician's true error bound becomes much tighter. This "extra work" on the part of the experimenter is significant, but tractable, and the resulting bounds are *much* tighter.

An alternative viewpoint is that the classification problem *is* finding a hypothesis with a low upper bound on the future error rate. We present a post-processing phase for neural networks which results in a classifier with a much lower upper bound on the future error rate. The post-processing can be used with any artificial neural net trained with any optimization method; it does not require the learning procedure be modified, re-run, or even that the threshold function be differentiable. In fact, this post-processing step can easily be adapted to other continuous valued learning algorithms.

The post-processing step finds a "large" distribution over classifiers, which has a small *average* empirical error rate. Given the average empirical error rate, it is straightforward to apply the PAC-Bayes bound in order to find a bound on the *average* true error rate. We can find this large distribution over classifiers by performing a simple noise sensitivity analysis on the learned model. The noise model allows us to generate a distribution of classifiers with a known, small, average empirical error rate. We refer to the distribution of neural nets that results from this noise analysis as a "stochastic" neural net model.

Why do we expect the PAC-Bayes bound to be a significant improvement over standard covering number and VC bound approaches? There exist learning problems for which the difference between the lower bound and the PAC-Bayes upper bound is tight up to $O\left(\frac{\ln m}{m}\right)$ where m is the number of training examples. This is superior to the guarantees which can be made for typical covering number bounds where the gap is, at best, known up to an (asymptotic) constant. The guarantee that PAC-Bayes bounds are sometimes quite tight encourages us to apply them here.

13.1. Theoretical setup

We first present a modern neural network bound (the "competition"), then specialize the PAC-Bayes bound to a stochastic neural network. A stochastic neural network is simply a neural network where each weight in the neural network is drawn from some distribution whenever it is used. The reason for constructing a stochastic neural network is that it will have a *much* lower true error upper bound than the neural network. Furthermore, this will be accomplished without increasing the empirical error rate more than marginally.

13.1.1. Neural Network bound. We will compare a specialization of the best current neural network true error rate bound [**28**] with our approach. The neural network bound is described in terms of the following parameters:

(1) A margin, $0 < \theta < 1$.
(2) A function ϕ defined by $\phi(x) = 1$ if $x < 0$, $\phi(x) = 0$ if $x > 1$, and linear in between.
(3) A_i, an upper bound on the sum of the magnitude of the weights in the ith layer of the neural network
(4) L_i, a Lipschitz constant which holds for the ith layer of the neural network. A Lipschitz constant is a bound on the magnitude of the derivative.
(5) d, the size of the input space.

With these parameters defined, we get the following bound.

THEOREM 13.1.1. *(2 Layer Feed-Forward Neural Network bound) For all* $\delta \in (0, 1]$

$$\Pr_D \left(\exists h \in H : \; e(h) > \inf_\theta \; b(\theta) \right)$$

where $b(\theta) = \frac{1}{m} \sum \phi \left(\frac{yh(x)}{\theta} \right) + \frac{2\sqrt{2\pi}}{\theta} 32 \sqrt{\frac{d+1}{m}} L_1 L_2 A_1 A_2 + \frac{\sqrt{\frac{1}{2} \ln \frac{2}{\delta}} + 2}{\sqrt{m}}$

PROOF. Given in [**28**] □

The theorem is actually only given up to a universal constant. "32" might be the right choice, but this is just an educated guess by the author [**42**]. The neural network true error bound above is (perhaps) the tightest known bound for general feed-forward neural networks and so it is the natural bound to compare with.

This 2 layer feed-forward bound is not easily applied in a tight manner because we can't calculate a priori what our weight bound A_i should be. This can be patched up using the principle of structural risk minimization. In particular, we can state the bound for $A_1 = \alpha^j$ where j is some non-negative integer and $\alpha > 1$ is a constant. If the jth bound holds with probability $\frac{\delta}{j(j+1)}$, then all bounds will hold simultaneously with probability $1 - \delta$, since

$$\sum_{j=1}^{\infty} \frac{1}{j(j+1)} = 1$$

Applying this approach to the values of both A_1 and A_2, we get the following theorem:

COROLLARY 13.1.2. *(2 Layer Feed-Forward Neural Network bound) For all* $\delta \in (0,1]$

$$\Pr_D\left(\exists h \in H, j, k: \;\; e(h) > \inf_{\theta} \; b(\theta, j, k)\right)$$

where $b(\theta,j,k) = \frac{1}{m}\sum \phi\left(\frac{yh(x)}{\theta}\right) + \frac{2\sqrt{2\pi}}{\theta}32\sqrt{\frac{d+1}{m}}L_1 L_2 \alpha^j \beta^k + \frac{\sqrt{\frac{1}{2}\ln\frac{2j(j+1)k(k+1)}{\delta}}+2}{\sqrt{m}}$

PROOF. Apply the union bound to all possible values of j and k as discussed above. □

In practice, we will use $\alpha = \beta = 1.1$ and report the value of the tightest applicable bound for all j, k.

13.1.2. Stochastic Neural Network bound.

We will specialize a PAC-Bayes bound (6.2.1) for application to a stochastic neural network with a choice of the "prior". Our "prior" will be zero on all neural net structures other than the one we train and a multidimensional isotropic gaussian on the values of the weights in our neural network. The multidimensional gaussian will have a mean of 0 and a variance in each dimension of b^2. This choice is made for convenience and happens to provide good results.

The optimal value of b is unknown and dependent on the learning problem so we will wish to parameterize it in an example dependent manner. We can do this using the same trick as for the original neural net bound. Use a sequence of bounds where $b = c\alpha^j$ for c and α some constants and j a nonnegative number. For the jth bound set $\delta \to \frac{\delta}{j(j+1)}$. The union bound will imply that all bounds hold simultaneously with probability at least $1 - \delta$.

Assuming that our "posterior" Q is also defined by a multidimensional gaussian with the mean and variance in each dimension defined by w_i and s_i^2, we can specialize to the following corollary:

COROLLARY 13.1.3. *(Stochastic Neural Network bound) Let k be the number of weights in a neural network, w_i be the i the weight and s_i be the variance of the ith weight. Then, we have:*

(13.1.1)

$$\Pr_D\left(\exists q(h): \;\; KL(\hat{e}_q(h)||e_q(h)) \geq \inf_j \frac{\sum_{i=1}^k \left[\ln\frac{c\alpha^j}{s_i} + \frac{s_i^2+w_i^2}{2c^2\alpha^{2j}} - \frac{1}{2}\right] + \ln\frac{j(j+1)m}{\delta}}{m-1}\right) \leq \delta$$

PROOF. Analytic calculation of the KL divergence between two multidimensional Gaussians and the union bound applied for each value of j. □

We will choose $\alpha = 1.1$ and $c = 0.2$ as reasonable default values.

One more step is necessary in order to apply this bound. The essential difficulty is evaluating $\hat{e}_q(h)$. This quantity is observable although calculating it to high precision is difficult. We will use the Monte Carlo sampling technique of section 6.3.1 in order to bound the value of $\hat{e}_q(h)$ and then use the bound on this value in the PAC-Bayes bound. We use $n = 1000$ evaluations of the empirical error rate of the stochastic neural network.

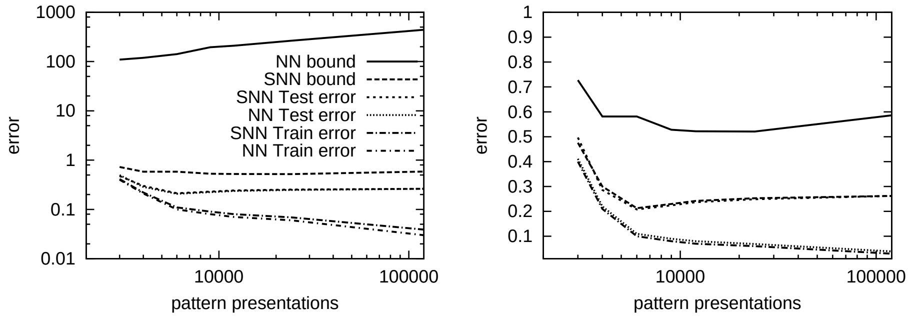


FIGURE 13.2.1. Plot of errors and true error bounds for the neural network (NN) and the stochastic neural network (SNN). The graph exhibits overfitting after approximately 6000 pattern presentations. The slope of the neural network true error bound is positive because the size of the weights is gradually increasing. Note that a true error bound of "100" implies that a factor of 100^2 more examples are required in order to make a non-vacuous bound. The graph on the right expands the vertical scale by excluding the poor true error bound.

13.1.3. Distribution Construction algorithm. One critical step is missing in the description: How do we calculate the multidimensional gaussian, Q? The variance of the posterior gaussian needs to be dependent on each weight in order to achieve a tight bound since we want any "meaningless" weights to not contribute significantly to the overall sample complexity. We use a simple greedy algorithm to find the appropriate variance in each dimension.

1. Train a neural net on the examples
2. For every weight, w_i, search for the variance, s_i^2, which reduces the empirical accuracy of the trained network by 1% (for example) while holding all other weights fixed.
3. The stochastic neural network defined by $\{w_i,\ s_i^2\}$ will generally have a too-large empirical error. Therefore, we calculate a global multiplier $\lambda < 1$ such that the stochastic neural network defined by $\{w_i,\ \lambda s_i^2\}$ decreases the empirical accuracy by only 1%.
4. Then, we evaluate the empirical error rate of the resulting stochastic neural net with 1000 samples from the stochastic neural network.

13.2. Experimental Results

How well can we bound the true error rate of a stochastic neural network? The answer is *much* better than we can bound the true error rate of a neural network.

Our experimental results take place on a synthetic data set which has 25 input dimensions and one output dimension. Most of these dimensions are useless—simply random numbers drawn from a $N(0, 1)$ Gaussian. One of the 25 input dimensions is dependent on the label. First, the label y is drawn uniformly from $\{-1, 1\}$, then the special dimension is drawn from a $N(y, 1)$ Gaussian. Note that this learning problem can not be solved perfectly because some examples will be drawn from the tail of the gaussian.

The "ideal" neural net to use in solving this problem is a single node perceptron. We will instead use a 2 layer neural net with 2 hidden nodes. This overly large neural net will result in the potential for significant overfitting which makes the bound prediction problem interesting. It is also somewhat more "realistic" if the neural net structure does not exactly fit the learning problem.

All of our data sets will use just 100 examples. Constructing a non-vacuous bound for a continuous hypothesis space at 100 examples is quite challenging as indicated by figure 13.2.1. Conventional bounds are hopelessly loose while the stochastic neural network bound is still not as tight as might be desired. There are several notable things about this figure.

(1) The SNN upper bound is *2-3* orders of magnitude lower than the NN upper bound.
(2) The SNN performs better than expected. In particular, the SNN true error rate is closer than 1% of the NN true error rate. This is surprising considering that we fixed the difference in empirical error rates at 1%.
(3) The SNN bound has a minimum at 12000 pattern presentations which weakly predicts the overfitting point of 6000 for both the SNN and the NN.

The comparison between the neural network bound and the stochastic neural network bound is not quite "fair" due to the form of the bound. In particular, the stochastic neural network bound can never return a value greater than "always err". This implies that when the bound is near the value of "1", it is difficult to judge how rapidly extra examples will improve the stochastic neural network bound. We can judge the sample complexity of the stochastic bound by plotting the value of the numerator in equation 13.1.1. Figure 13.2.2 plots the complexity versus the number of pattern presentations in training.

The stochastic bound is a radical improvement on the neural network bound but it is not yet a perfectly tight bound. Given that we do not have a perfectly tight bound, one important consideration arises: does the minimum of the stochastic bound predict the minimum of the true error rate (as predicted by a large holdout data set). In particular, can we use the stochastic bound to determine when we should cease training? The stochastic bound depends upon (1) the complexity which increases with training time and (2) the training error which decreases with training time. This dependence results in a minima which for our problem occurs at approximately 12000 pattern presentations. The point of minimal true error (for the stochastic and deterministic neural networks) occurs at approximately 6000 pattern presentations indicating that the stochastic bound weakly predicts the point of minimum error. The neural network bound has no such minimum.

Is the choice of 5% increased empirical error optimal? In general, the "optimal" choice of the extra error rate depends upon the learning problem. Since the stochastic neural network bound (corollary 13.1.3) holds for all multidimensional gaussian distributions, we are free to optimize the choice of distribution in anyway we desire. Figure 13.2.2 shows the resulting bound for different choices of q. The bound has a minimum at 0.03 extra error indicating that a slightly lower bound is possible if we accept a larger training error. Also note that the complexity always decreases with increasing entropy in the distribution of our stochastic neural net. The existence of a minimum in Figure 13.2.2 is the "right" behavior: the increased empirical error rate is significant in the calculation of the true error bound.

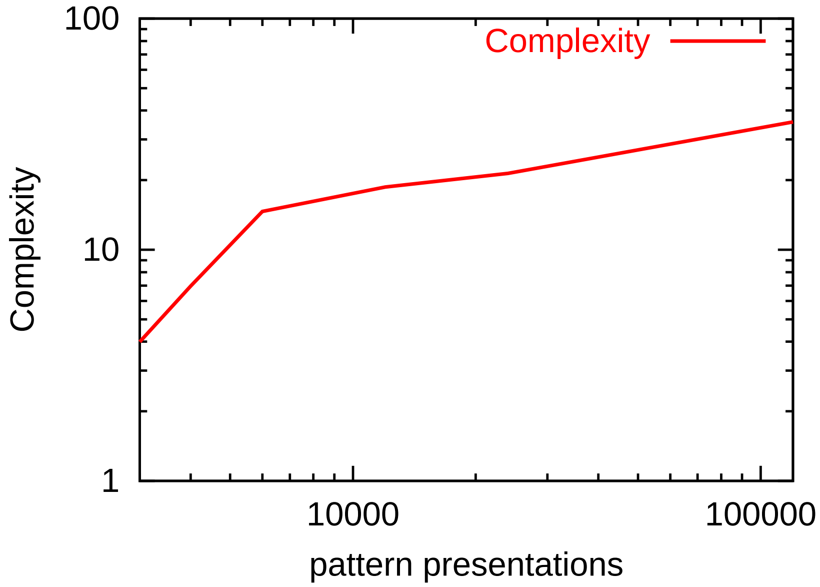
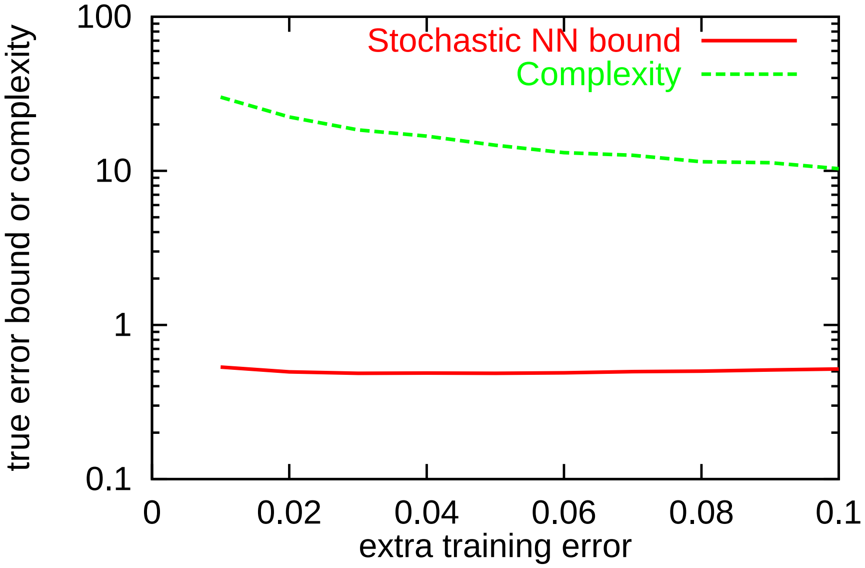


FIGURE 13.2.2. We plot the "complexity" of the stochastic network model (numerator of 13.1.1) vs. training epoch. Note that the complexity increases with more training as expected and stays below 100, implying non-vacuous bounds on a training set of size 100.



Plot of the stochastic neural net (SNN) bound for "posterior" distributions chosen according to the extra empirical error they introduce.

13.3. Conclusion

PAC-Bayes bounds give excellent results on a stochastic neural network. The stochastic neural network bound is radically tighter ($2 - 3$ orders of magnitude) bound on the true error rate of a classifier while increasing the empirical and true error rates only a small amount.

Although, the stochastic neural net bound is not completely tight, it is not vacuous with just 100 examples and the minima of the bound weakly predicts the point where overtraining occurs.

PROBLEM 13.3.1. (Open) To what extent do these results extend to other learning problems and other continuous learning algorithms? Work is under-way (most notably in [**48**]) to evaluate both of these questions.

CHAPTER 14

Conclusion & Challenges

The basic conclusion of this thesis is that we can achieve bounds tight enough to yield useful results on real learning problems with standard learning algorithms or simple variants on standard learning algorithms. The evidence of this conclusion is reported in the last two chapters.

To accomplish this goal, considerable theoretical work was completed. This includes:

(1) Derivation of the microchoice bounds and adaptive microchoice bounds in Section 5.
(2) Improvement and simplification of the PAC-Bayes bound in Section 6.
(3) Improvement of the margin bound to achieve benefit from averaging 7.
(4) Derivation of the Shell bounds 8.
(5) An investigation into how to repair the covering number approach 9.
(6) An analysis of progressive validation 10.
(7) An analysis for the combination of training and test set bounds 11.

In particular, microchoice bounds (Section 5), PAC-Bayes bounds (Section 6), shell bound (Section 8), and combined bounds (Section 11) have proved useful for practical application.

It is worth emphasizing that all of the bounds reported here rest upon only an assumption of example independence implying wide applicability.

Bibliography

[1] Peter Bartlett, “The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is More Important than the Size of the Network”, IEEE Transactions on Information Theory, Vol. 44, No. 2, March 1998.

[2] Gyora M. Benedek and Alon Itai, “Learnability by Fixed Distributions”, Proceedings of the 1988 Workshop on Computational Learning Theory.

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

[4] Avrim Blum and Ron Rivest, “Training a 3-Node Neural Network is NP-Complete”, Neural Networks, 5(1):117-127, 1992.

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

[6] L. Breiman, "Bagging Predictors" Machine Learning, Vol. 24, No . 2, pp. 123-140.

[7] H. Chernoff, “A Measure of the asymptotic efficiency of tests of a hypothesis based on the sum of observations”, Annals of Mathematical Statistics, 23:493-507, 1952

[8] N. Christianini, J. Shaw-Taylor, “Support Vector Machines”, Cambridge University Press, 2000 ISBN: 0 521 78019 5

[9] Claude Cohen-Tannoudji, Bernard Diu, Frank Laloe, Bernard Dui, “Quantum Mechanics”

[10] Thomas Cover and Joy Thomas, “Elements of Information Theory” Wiley, New York 1991.

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

[12] Pedro Domingos. “A Process-Oriented Heuristic for Model Selection, Machine Learning Proceedings of the Fifteenth International Conference, "Morgan Kaufmann Publishers, San Francisco, CA, 1998, pp 127-135.

[13] R. M. Dudley, “A course on empirical processes”, Lecture Notes in Mathematics 1097, 2-141. Springer-Verlag, New York.

[14] A. Ehrenfucht, D. Haussler, M. Kearns, and L. Valiant, “A General Lower Bound on the Number of Examples Needed for Learning”, Information and Computation 82(3), pp. 247-261, 1989.

[15] B. Efron and R. Tibshirani, “An Introduction to the Bootstrap”, Chapman & Hall, London, 1993.

[16] Yoav Freund. “Predicting a binary sequence almost as well as the optimal biased coin”, COLT 1996. http://citeseer.nj.nec.com/freund96predicting.html

[17] Yoav Freund. Self-Bounding Learning Algorithms. COLT 98. http://www.research.att.com/~yoav/papers/lsearch.ps.gz

[18] Yoav Freund and Robert E. Schapire, “A Decision Theoretic Generalization of On-line Learning and an Application to Boosting” Eurocolt 1995

[19] Oded Goldreich, “The Foundations of Cryptography - Volume 1”, ISBN 0-521-79172-3 Cambridge University Press, 2001

[20] David Haussler, “Mathematical perspectives on Neural networks ”, Lawrence Erlbaum Associates, 1995.

[21] David Haussler, Michael Kearns, and Robert Schapire, “Bounds on the Sample Complexity of Bayesian Learning Using Information Theory and the VC dimension”, Machine Learning 14, pp. 83-113, 1994.

[22] David Haussler, Michael Kearns, H. Sebastian Seung, and Naftali Tishby, “Rigorous Learning Curve Bounds from Statistical Mechanics” Machine Learning,25, 1996, pages 195–236.

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

[24] T. Jaakkola, M. Meila, T. Jebara, "Maximum Entropy D iscrimination\char" NIPS 1999.

[25] Adam Kalai, "Probabilistic and On-line methods in Machine Learning".thesis, CMU-CS-01-132.

[26] Michael Kearns. Efficient Noise-Tolerant Learning From Statistical Queries, Proceedings of the 25th ACM Symposium on the Theory of Computing, pp. 392-401, 1993, ACM Press. http://www.research.att.com/~mkearns/papers/sq-noise.ps.Z

[27] 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.

[28] V. Koltchinskii and D. Panchenko, "Empirical Margin Distributions and Bounding the Generalization Error of Combined Classifiers", preprint, http://citeseer.nj.nec.com/386416.html

[29] P.Kontkanen, P. Myllymaki, T. Silander, H.Tirri, and P. Grunwald, Predictive Distributions and Bayesian Networks, Journal of Statistics and Computing 10, pp. 39-54, 2000.

[30] John Langford and Avrim Blum 1999. Microchoice Bounds and Self Bounding learning algorithms. COLT99. http://www.cs.cmu.edu/~jcl/papers/microchoice/mc.ps

[31] 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

[32] John Langford and Rich Caruana, (Not) Bounding the True Error NIPS2001.

[33] John Langford and David McAllester, "Computable Shell Decomposition Bounds" COLT 2000.

[34] John Langford, Matthias Seeger, and Nimrod Megiddo, "An Improved Predictive Accuracy Bound for Averaging Classifiers" ICML2001.

[35] John Langford and Matthias Seeger, "Bounds for Averaging Classifiers." CMU tech report, CMU-CS-01-102, 2001.

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

[37] Tom Mitchell, "Machine Learning", McGraw Hill, 1997.

[38] Yishay Mansour, "Pessimistic Decision Tree Pruning Based on Tree Size", ICML1997.

[39] David McAllester, "PAC-Bayesian Model Averaging" COLT 1999.

[40] Colin McDiarmid, "On the method of bounded differences", In \emph {Surveys in Combinatorics 1989,} pages 148-188. Cambridge University Press, 1989.

[41] Jon Mingers, An Empirical Comparison of Pruning Methods for Decision Tree Induction, Machine Learning, 1989, vol. 4, 227-243.

[42] Dimitry Panchenko, personal communications, 2001

[43] David Pollard, "Convergence of Stochastic Processes", Springe r-Verlag 1984.

[44] J.L. Quinlan, Induction of Decision Trees, Machine Learning, 1986, vol. 1, pp 81-106.

[45] R.L. Rivest, Learning Decision Lists. Machine Learning, 1987, vol. 2, pp 229-246.

[46] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee, "Boosting the Margin: A new explanation for the effectiveness of voting methods" The Annals of Statistics, 26(5):1651-1686, 1998.

[47] Tobias Scheffer and Thorsten Joachims, "Expected Error Analysis for Model Selection", ICML 1999.

[48] Matthias Seeger, "PAC-Bayesian Generalization Error Bounds for Gaussian Processes", Tech Report, Division of Informatics report EDI-INF-RR-0094. http://www.dai.ed.ac.uk/homes/seeger/papers/gpmcall-tr.ps.gz

[49] Aad W. van der Vaart and Jon A. Wellner, "Weak Convergence and Empirical Processes", Springer-Verlag 1996.

[50] L.G. Valiant. "A Theory of the Learnable." Communications of the ACM 27(11):1134-1142, November 1984

[51] 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.

[52] Vladimir N. Vapnik, "Statistical Learning Theory", Wiley, December 1999.

[53] Jon Baxter. A Model of Inductive Bias Learning. 1998. http://wwwsyseng.anu.edu.au/~jon/papers/ltolchap.ps.gz

CHAPTER 15

Appendix: Definitions

Term	Definition
x	The “input” which we can predict with.
y	The “output” which we want to predict.
(x, y)	An “example” or “sample” which is an <input,output> pair
S	A set of samples.
m	The number of samples.
m_{test}	The number of samples in the testing set.
m_{train}	The number of samples in the training set.
h	A hypothesis = a function from x to y
D	An (unknown) distribution over (x, y) pairs.
$\text{Bin}(m, k, p)$	The probability that a Binomial with m coins and bias p has k or fewer heads.
$\bar{e}(m, k, \delta)$	An upper bound on $\text{Bin}(m, k, p)$.
$p(h)$	A distribution over hypotheses not dependent on the training set.
$q(h)$	A distribution over hypotheses dependent on the training set.
$\hat{e}_S(h)$	Error rate on the sample set S
$e_D(h)$	$\Pr_{(x,y)\sim D}(h(x) \neq y)$ = true error rate = future error rate
$\hat{e}_{\text{test}}(h)$	Error rate on a test set
δ	The probability that a bound fails. Typically small.

CHAPTER 16

Appendix: Manual

This chapter is intended as a manual for the practical application of learning-theory based bounds. In particular, we introduce a program “bound” available at:

http://www.cs.cmu.edu/˜jcl/programs/bound/bound.html

16.1. Test Error Bound Calculation

We can use the holdout bound to calculate upper and lower true error bounds with the program “bound”. Here is an example of the application.

```
11:32PM z-12: echo "test_examples 600
test_errors 192
delta 0.025" | bound
Applying varying approximation tail bound
delta 0.025
lower_delta 0.5
test_examples 600
test_errors 192
approximation automatic
true_error = 0.32 0.358976827934 0.31926717516
```

There are several arguments passed to the program. These include:

(1) test_examples <int>: the number of test examples.
(2) test_errors <int>: the number of test errors.
(3) delta <float>: the probability of failure of the upper bound.
(4) lower_delta <float>: the probability of failure of the lower bound.

The output of the program is several lines stating the used and assumed arguments and a final line of the form:

```
true_error = <error rate> <upper_bound> <lower_bound>
```

The output of this particular application implies that the true error rate of the chosen hypothesis is less than 0.35897 with high confidence (over the drawn sample set).

For simplicity, the arguments can also be placed into a file:

```
11:35PM z-14: cat test_error
test_examples 800
test_errors 192
delta 0.025
lower_delta 0.025
11:35PM z-15: bound test_error
Applying varying approximation tail bound
```

```
delta 0.025
lower_delta 0.025
test_examples 800
test_errors 192
approximation automatic
true_error = 0.24 0.28252880089 0.20065891277
```

The output of this program can be interpreted as a confidence interval.

16.2. Training Set Bound Calculation

Many of the training set based bounds effectively reduce the value of δ by some fixed amount. The program 'bound' will automatically reduce the value of δ with two arguments.

1. log_prior <float>: the log (base e) of the "prior" (in the sense of theorem 4.6.1) of the hypothesis.
2. log_hypothesis_size <float>: the log of the number of hypotheses (in the sense of theorem 4.2.1)

Each of these arguments acts independently to reduce the value of δ. The value of δ will only be reduced on *training* example based bounds. There are two more arguments necessary for application of one of the simple training set based bound:

1. training_examples <int>: the number of training examples.
2. training_errors <int>: the number of training errors.

Here is an application of a simple training set based bound:

```
12:07AM z-26: cat train_error
train_examples 100
train_errors 3
log_prior -50
delta 0.3
12:07AM z-27: bound train_error
Applying varying approximation tail bound
delta 0.3
lower_delta 0.5
train_examples 100
train_errors 3
log_hypothesis_size 0
log_prior -50
approximation automatic
true_error = 0.03 0.466502545401 9.31322574615e-10
```

16.3. Shell Bound Calculation

The program 'bound' can calculate a shell bound or a sampling shell bound. To do these calculations, two extra parameters must be specified:

1. error_log_count <int> <float>: <float> should be the log (base e) of the number of hypotheses with <int> empirical error.
2. sample_space_log_size <float>: <float> should be the size of space of hypotheses uniformly sampled from if the evaluation is inexact (and not passed if the evaluation is exact)

Suppose wanted to use the shell bound and knew that e^5 hypotheses had 25 training error and e^{30} had 50 training errors. Then, we might apply the bound as:

```
12:07AM z-28: cat shell_error
train_examples 100
train_errors 3
error_log_count 3 1
error_log_count 25 5
error_log_count 50 30
delta 0.3
12:07AM z-29: bound shell_error
Applying varying approximation tail bound
Applying shell bound
delta 0.3
lower_delta 0.5
train_examples 100
error_log_count 3 1
error_log_count 25 5
error_log_count 50 30
approximation automatic
true_error = 0.03 0.144696196541 0.0266506755725
```

Now, suppose that we wanted to use the sampling shell bound and sampled $e^5 + e^{10}$ times observing e^5 hypotheses with training error 25 and e^{10} with training error 50. Then, we might apply 'bound' as follows:

```
12:23AM z-36: cat sampling_shell_error
train_examples 100
train_errors 3
error_log_count 3 1
error_log_count 25 5
error_log_count 50 10
sample_space_log_size 30
delta 0.3
12:23AM z-37: bound sampling_shell_error
Applying varying approximation tail bound
Applying shell bound
delta 0.3
lower_delta 0.5
train_examples 100
sample_space_log_size 30
error_log_count 3 1
error_log_count 25 5
error_log_count 50 10
approximation automatic
true_error = 0.03 0.405203092843 0.0266506755725
```

16.4. Combined Bound Calculation

The program 'bound' has been instrumented to use technique (3) from above. In order to apply the combined training and test error bound, you must simply supply the necessary information for both the training and test sets.

```
12:25AM z-42: cat train_n_test_error
test_examples 10
test_errors 5
delta 0.3
train_examples 20
train_errors 0
log_prior -1
 1:16AM z-43: bound train_n_test_error
Applying varying approximation tail bound
delta 0.3
lower_delta 0.5
test_examples 10
test_errors 5
train_examples 20
train_errors 0
log_hypothesis_size 0
log_prior -1
approximation automatic
true_error = 0.5 0.104335444048 0.369755505584
```