Microchoice Bounds and Self Bounding Learning Algorithms

John Langford* and **Avrim Blum***
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{jcl+,avrim+}@cs.cmu.edu

September 16, 2000

Abstract

Our goal is proving tighter upper bounds on the true error rates of learned hypotheses based upon their empirical performance on training data. We prove bounds for learning algorithms that operate by making a sequence of choices. These adaptive bounds, which we call Microchoice bounds, are similar to Occam-style bounds and can be used to make learning algorithms self-bounding in the style of Freund [Fre98]. We can also combine these bounds with Freund's query-tree approach producing a version of Freund's query-tree structure that can be implemented with much more algorithmic efficiency.

1 Introduction

Sample complexity bounds tend to be too loose for application to real world learning problems using common learning algorithms. They require more examples than experimentally necessary to guarantee a reasonable accuracy with a reasonable probability. Chief amongst the reasons for this failure is that typical sample complexity bounds are worst-case in nature and do not take into account what may turn out to be a fortunate target function and/or data distribution. A tighter bound can perhaps be derived by taking properties of the observed data into account.

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

*Supported in part by NSF grant CCR-9732705

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 of each choice set, can we produce a tighter bound?

We develop a bound we call the Microchoice bound, which is a natural approach to analyzing algorithms of this form and lends itself to the construction of Self bounding Learning algorithms in the style of Freund [Fre98] in a straightforward way. The goal is to allow algorithms to bound the difference between the true and empirical errors of the output hypothesis based on the choices actually taken and the choice sets actually explored without needing to consider sets that "might have been" visited for different learning problems. By keeping track of the sizes of the choice sets visited, the bound can be calculated and output along with the answer. The simpler version of the Microchoice bound, which we describe first, can be thought of as a direct application of the Occam's razor connection developed by [BEHW87]. In particular, 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. In the second half of this paper, we show how the Microchoice bound can be combined with the query-tree approach of Freund to produce a variant of the query tree that can be used much more efficiently. Our procedures work without altering the final hypothesis produced by the learning algorithm. It is also possible to modify the learning algorithm to use a current bound in order to implement early stopping in a manner similar to that of dynamic Structural Risk Minimization [STBWA96].

Self bounding Learning algorithms were first suggested by Freund [Fre98]. Work developing approximately Self Bounding Learning algorithms was also done by Domingos [Dom98].

2 Discrete PAC Bound Review

The basic PAC model is characterized by a hypothesis set, H, composed of hypotheses, $h : X \to Y$. We will be considering only discrete sets H. The learning algorithm is given N examples $x_1, \ldots, x_N$ generated according to an unknown distribution D over the instance space X, and labeled by some unknown target function c. We will call the resulting set of labeled examples $z^N = \{z_1, \ldots, z_N\}$, where $z_i = (x_i, c(x_i))$. The function $c(x_i)$ may be probabilistic. We use P to denote the distribution on labeled examples induced by D and c. Let $E_P(h)$ denote the true error of some hypothesis h with respect to P, and let $E(h, z^N)$ denote h's empirical error on the sample.

In the realizable setting, we assume there exists some $h \in H$ with $E_P(h) = 0$. In this setting we typically wish to state that with high probability no hypothesis of high true error will have zero empirical error. Let $\Pr_{z^N}\left[\phi(z^N)\right]$ be the probability that ϕ holds over the sample set. The standard result, using the union bound, is that for any P, for any $\delta > 0$,

$$\Pr_{z^N}\left[\exists h \in H : E(h, z^N) = 0 \text{ and } E_P(h) > \epsilon\right] < \delta, \tag{1}$$

$$\text{for } \epsilon = \frac{1}{N}\left(\ln \frac{|H|}{\delta}\right).$$

A folk result, stated explicitly as Preliminary Theorem 1 in [McA98], is that one can weight the hypotheses in H according to any probability measure m, resulting in different error bounds for different hypotheses. In particular, one way to think of the standard bound above is that we take our "supply" of confidence, δ, and split it evenly among the hypotheses, giving $\delta/|H|$ to each. Then we define ϵ so that the probability a fixed hypothesis of true error ϵ has empirical error 0 is at most $\delta/|H|$. The generalization is simply that we can allocate our supply of δ according to any distribution m if we wish, giving $\delta(h) = \delta \cdot m(h)$ to each. We then define $\epsilon(h)$ so that the probability a hypothesis of true error $\epsilon(h)$ has empirical error 0 is at most $\delta(h)$, and apply the union bound. Specifically,

Theorem 2.1 *(Realizable Folk Theorem [McA98]) For any $\delta > 0$, any P, and any measure m over H,*

$$\Pr_{z^N}\left[\exists h \in H : E(h, z^N) = 0 \text{ and } E_P(h) > \epsilon(h)\right] < \delta, \quad \text{for} \quad \epsilon(h) = \frac{1}{N}\left(\ln\frac{1}{\delta} + \ln\frac{1}{m(h)}\right) \tag{2}$$

For example, the Occam's razor connection observed by [BEHW87] can be viewed in this framework by giving every hypothesis of description length l a weight of $m(h) = 2^{-2l}$. In the agnostic case, where the empirical error can be any value rather than only zero, we wish to bound the deviation between the true and empirical errors of any hypothesis. The same trick of unevenly spreading the confidence δ over the hypotheses, combined with Hoeffding bounds, produces the following result stated explicitly as Preliminary Theorem 2 in [McA98].

Theorem 2.2 *(Agnostic Folk Theorem [McA98]) For any $\delta > 0$, any P, and any measure m over H,*

$$\Pr_{z^N}\left[\exists h \in H : E_P(h) > E(h, z^N) + \epsilon(h)\right] < \delta,$$

$$\text{for} \quad \epsilon(h) = \sqrt{\frac{1}{2N}\left(\ln\frac{1}{\delta} + \ln\frac{1}{m(h)}\right)}$$

2.1 A Motivating Observation

For a given learning algorithm A, a distribution P on labeled examples induces a distribution $p(h)$ over the possible hypotheses $h \in H$ produced by algorithm A after N examples[1]. If we knew P, then we could apply the Realizable Folk Theorem with the measure $m(h) = p(h)$. This choice of $m(h)$ is optimal for minimizing the expected value of $\epsilon(h)$. In particular, notice that

$$\begin{aligned} &\inf_m \sum_{h \in H} p(h) \frac{1}{N}\left(\ln\frac{1}{\delta} + \ln\frac{1}{m(h)}\right) \\ &= \frac{1}{N}\left(\ln\frac{1}{\delta} + \inf_m \sum_{h\in H} p(h) \ln\frac{1}{m(h)}\right) \end{aligned}$$

This corresponds to minimizing the Kullback-Liebler divergence which provides a solution of $m(h) = p(h)$.

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

Interestingly, for the agnostic bound this choice of measure, $m(h) = p(h)$, does not minimize the expected value of $\epsilon(h)$, as can be seen from a quick analysis of the two hypothesis case. Instead, this choice of measure minimizes the expected value of $\epsilon(h)^2$.

$$\inf_m \sum_{h \in H} p(h)\epsilon(h)^2 = \inf_m \sum_{h \in H} p(h)\frac{1}{2N}(\ln \frac{1}{\delta} + \ln \frac{1}{m(h)})$$

From here, the analysis is the same as for the realizable case.

The point of these observations is to notice that if we can use the structure of the learning algorithm to produce a choice of $m(h)$ that approximates $p(h)$, this should result in better estimation bounds.

3 The Microchoice Bound

The Microchoice bound is essentially a compelling and easy way to compute the selection of the measure $m(h)$ 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 $|C|$. Depending on the learning algorithm, subtrees 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 [Riv87] 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 way we determine $m(h)$ (equivalently, $\delta(h)$) is as follows. We take our "supply" of δ and give it to 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 confidence parameter is allocated among the leaves. If we examine some leaf containing a hypothesis h, we see that this method gives it $\delta(h) = \delta \prod_{i=1}^{d(h)} \frac{1}{|C_i(h)|}$, where $d(h)$ is the depth of h in the choice tree and $C_1(h), C_2(h), \ldots, C_{d(h)}(h)$ is the sequence of choice sets traversed on the path to h. Equivalently, we allocate $m(h) = \prod_{i=1}^{d(h)} \frac{1}{|C_i(h)|}$ measure to each 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, we will neglect this issue, implying that the Microchoice bound 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,

$m(h)$ 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 P on labeled examples induces a probability distribution over choice sequences, inducing a probability distribution $p(h)$ over hypotheses. Ideally we would like to use $m(h) = p(h)$ in our bounds as noted above. However, we cannot calculate $p(h)$, so instead, our choice of $m(h)$ will be just an estimate. In particular, our estimate $m(h)$ 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 theorems.

Theorem 3.1 *(Realizable Microchoice Theorem) For any $\delta \in (0,1)$,*

$$\Pr_{z^N}\left[\exists h \in H : E(h, z^N) = 0 \text{ and } E_P(h) > \epsilon(h)\right] < \delta$$

$$\text{for } \ \epsilon(h) = \frac{1}{N}\left(\ln\frac{1}{\delta} + \sum_{i=1}^{d(h)} \ln|C_i(h)|\right)$$

Proof. Specialization of the Realizable Folk Theorem.

Theorem 3.2 *(Agnostic Microchoice Theorem) For any $\delta \in (0,1)$,*

$$\Pr_{z^N}\left[\exists h \in H : E_P(h) > E(h, z^N) + \epsilon(h)\right] < \delta$$

$$\text{for } \ \epsilon(h) = \sqrt{\frac{1}{2N}\left(\ln\frac{1}{\delta} + \sum_{i=1}^{d(h)} \ln|C_i(h)|\right)}$$

Proof. Specialization of the Agnostic Folk Theorem.

The point of these Microchoice bounds is that the quantity $\epsilon(h)$ is something the algorithm can calculate as it goes along, based on the sizes of the choice sets encountered. Furthermore, 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 $d \log F$ where d is the length of the list produced and F is the number of features. Notice that $d \log F$ is also the description length of the final hypothesis produced in the natural encoding, thus in this case these theorems yield similar bounds to 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 towards 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 $m(h)$. 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 $m(h) = p(h)$. In fact, the method of section 4 where we combine these results with Freund's query-tree approach can be thought of along these lines.

3.1 Examples

The Microchoice Bound is not always better than the simple sample complexity bound (1). To develop some understanding of how they compare we consider several cases.

3.1.1 Greedy Set Cover

Consider a greedy set cover algorithm for learning an OR function over F boolean features. The algorithms 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:

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

The bound of (1) is:

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

If k is small, then the Microchoice bound is a lot better, but if $k = O(F)$ then the Microchoice bound is slightly worse. 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.

3.1.2 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:

$$E_P(h) \leq E(h, z^N) + \sqrt{\frac{1}{2N}(\ln\frac{1}{\delta} + \sum_v \ln(F - d(v) + 1))} \tag{3}$$

where v ranges over the nodes of the decision tree.

3.1.3 Pruning

Decision tree algorithms for real-world learning problems often have some form of "pruning" as in [Qui86] and [Min89]. 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 towards 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 (3) 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 $\epsilon(h)$. The similarities to SRM are discussed below.

3.2 Relationship with Structural Risk Minimization

Structural Risk Minimization works with a sequence of nested hypothesis sets, $H_1 \subset H_2 \subset .... \subset H_l$. For each hypothesis set, a PAC bound on the difference between empirical and true error exists.

Theorem 3.3 *(Standard Sample Complexity Bound)*

$$\Pr_{z^N}\left[\exists h \in H_i : E_P(h) > E(h, z^N) + \epsilon(h)\right] < \delta$$

$$\textit{for} \quad \epsilon(h) = \sqrt{\frac{1}{2N}\left(\ln\frac{1}{\delta} + \ln|H_i|\right)}$$

The simplest way to improve the bound to include all hypothesis sets is with the following simple theorem.

Theorem 3.4 *(Structural Risk Minimization) Let $m(i)$ be some measure across the l hypothesis sets with $\sum_{i=1}^{l} m(i) = 1$. Then:*

$$\Pr_{z^N}\left[\exists h \in H_i \in [H_1, ..., H_l] : E_P(h) > E(h, z^N) + \epsilon(h)\right] < \delta$$

$$\text{for } \ \epsilon(h) = \sqrt{\frac{1}{2N}\left(\ln\frac{1}{\delta} + \ln|H_i| + \ln\frac{1}{m(i)}\right)}$$

The proof is simple — just apply the union bound to the standard PAC theorem.

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.

The above Microchoice bound is essentially a compelling application of the Disjoint SRM bound 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 $m(i)$ 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}.

4 Combining Microchoice with Freund's Query Tree approach

The remainder of the paper is devoted to an improvement of the Microchoice bound called Adaptive Microchoice which arises from synthesizing Freund's query trees with the Microchoice bound. This improvement is not easily expressed as a simplification of Structural Risk Minimization. First we require some background material in order to state and understand Freund's bound.

4.1 Preliminaries and Definitions

The statistical query framework introduced by Kearns [Kea93] is the same as the PAC framework, except that the learning algorithm can only access its data using statistical queries. A statistical query takes as input a binary predicate, χ, mapping examples [2] to a binary output: $(X, Y) \to \{0, 1\}$. The output of the statistical query is the average of χ over the examples seen,

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

The output is an empirical estimate of the true value $P_\chi = \mathbf{E}_P[\chi(z)]$ of the query under the distribution P. It is convenient to define

$$\alpha(\delta) = \sqrt{\frac{1}{2N} \ln \frac{2}{\delta}}.$$

Then, by Hoeffding's inequality,

$$\Pr\left[|\hat{P}_\chi - P_\chi| > \alpha(\delta)\right] < \delta.$$

4.2 Background and Summary

Freund [Fre98] 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, and distribution P, Freund defines the query tree $T_A(P, \alpha)$ as the choice tree created by considering only those choices resulting from answers $\hat{P}_\chi$ to queries χ such that $|\hat{P}_\chi - P_\chi| \leq \alpha$. The idea is that if a particular predicate, χ, is true with probability .9 on a random example 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 $2e^{-2N\alpha^2}$ by Hoeffding's inequality. So, if the entire tree contains a total of $|Q(T_A(P, \alpha))|$ queries in it, the probability *any* of these queries is off by more than α is at most $2 \cdot |Q(T_A(P, \alpha))| \cdot e^{-2N\alpha^2}$. 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 $p(h)$ distribution considered in Section 2.1, namely that to compute it, one needs to know P. So, Freund proposes an algorithmic method to find a superset 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 $\hat{P}_\chi$ is received, if it is true that $|\hat{P}_\chi - P_\chi| \leq \alpha$, then the range $[\hat{P}_\chi - 2\alpha, \hat{P}_\chi + 2\alpha]$ contains the range $[P_\chi - \alpha, P_\chi + \alpha]$. Thus, under the assumption that no

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

query in the *correct* tree is answered badly, a superset 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 superset 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 [Fre98]; we will be able to finesse that issue.

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

4.3 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 3, 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 δ' of its "supply" $\tilde{\delta}$ of confidence parameter to the event that the output of the statistical query being asked is within $\alpha(\delta')$ of the correct answer. The node then splits the remainder of its supply, $\tilde{\delta} - \delta'$, evenly among the children corresponding to choices that result from answers $\hat{P}_\chi$ with $|\hat{P}_\chi - P_\chi| \leq \alpha(\delta')$. 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.

How should δ' be chosen? Smaller values of δ' result in larger values of $\alpha(\delta')$ leading to more children in the pruned tree and less confidence given to each. Larger values of δ' result in less left over to divide among the children. Unfortunately, our algorithmic approximation (which only sees the empirical answers $\hat{P}_\chi$ and needs to be efficient) will not be able to make this optimization. Therefore, we define δ' in the *true* Microchoice query tree to be $\frac{\tilde{\delta}}{d+1}$ where d is the depth of the current node.

The algorithmic approximation uses the idea of [Fre98] of including all choices within 2α of the observed value $\hat{P}_\chi$. Unlike [Fre98], however, we do not actually create the tree; instead we just follow the path taken by the learning algorithm, and argue that the "supply" $\delta(h)$ of confidence remaining at the leaf is no greater than the amount that would have been there in the original tree. Finally, the algorithm outputs $\epsilon(h)$ corresponding to $\delta(h)$.

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 $\tilde{\delta}$ supply of confidence parameter. (If the current node is the root, then $\tilde{\delta} = \delta$ and $d(\chi) = 1$). We choose $\delta' = \tilde{\delta}/(d+1)$, ask the query χ, and receive $\hat{P}_\chi$. We now let k be the number of children of our node corresponding to answers in the range $[\hat{P}_\chi - 2\alpha(\delta'), \hat{P}_\chi + 2\alpha(\delta')]$. We then go to the child corresponding to the answer $\hat{P}_\chi$ that we received, giving this child a confidence parameter supply of $(\tilde{\delta} - \delta')/k$. This is the same as we would have given it had we allocated $\tilde{\delta} - \delta'$ to the children equally. We then continue from that child. Finally, when we reach a leaf, we output the current hypothesis h along with the error estimate $E(h, z^N) + \epsilon(\tilde{\delta})$, where $\epsilon(\tilde{\delta})$ is the error term corresponding to the remaining confidence supply $\tilde{\delta}$.

We now prove that with probability $1 - \delta$, our error estimate $E(h, z^N) + \epsilon(h)$ is correct. 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 argue that in this case, the confidence assigned to the final hypothesis in the algorithmic construction is no greater than the confidence assigned in the true query tree. In particular, assume inductively that at the current node of our empirical path the supply $\tilde{\delta}_{\text{emp}}$ is no greater than the supply $\tilde{\delta}_{\text{true}}$ given to that node in the true tree. Now, under the assumption that the response $\hat{P}_\chi$ is "good", it must be the case that the interval $[\hat{P}_\chi - 2\alpha(\tilde{\delta}_{\text{emp}}/(d+1)), \hat{P}_\chi + 2\alpha(\tilde{\delta}_{\text{emp}}/(d+1))]$ contains the interval $[P_\chi - \alpha(\tilde{\delta}_{\text{true}}/(d+1)), P_\chi + \alpha(\tilde{\delta}_{\text{true}}/(d+1))]$. Therefore, the supply given to our child in the empirical path is no greater than the supply given in the true tree.

Let $d(h)$ be the depth of some hypothesis h in the empirical path and $\hat{C}_1(h), \hat{C}_2(h), \ldots, \hat{C}_d(h)$ 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:

$$\delta'(h) = \delta \prod_{i=1}^{d(h)} \left(\frac{i}{i+1} \frac{1}{|\hat{C}_i(h)|} \right) = \delta \frac{1}{d(h)+1} \prod_{i=1}^{d(h)} \frac{1}{|\hat{C}_i(h)|} \tag{4}$$

Let $A(z^N) = h$ be the output of the learning algorithm.

We now have the following two theorems for the realizable and agnostic cases:

Theorem 4.1 *(Realizable Adaptive Microchoice Theorem) For any* $0 < \delta < 1$, *under the assumption that* A *produces only hypotheses with zero empirical error,*

$$\Pr_{z^N}\left[A(z^N) = h : E_P(h) > \epsilon(h)\right] < \delta$$

$$\textit{for } \epsilon(h) = \frac{1}{N}\left(\ln\frac{1}{\delta} + \ln(d(h)+1) + \sum_{i=1}^{d(h)} \ln(|\hat{C}_i(h)|)\right).$$

Theorem 4.2 *(Agnostic Adaptive Microchoice Theorem) For any* $0 < \delta < 1$,

$$\Pr_{z^N}\left[A(z^N) = h : E_P(h) > E(h, z^N) + \epsilon(h)\right] < \delta$$

$$\textit{for } \epsilon(h) = \sqrt{\frac{1}{2N}(\ln\frac{1}{\delta} + \ln(d(h)+1) + \sum_{i=1}^{d(h)} \ln(|\hat{C}_i(h)|)}).$$

The bounds in theorems 4.1 and 4.2 are very similar to 3.1 and 3.2 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 $\hat{C}_i(h)$.

4.4 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 $\delta' = \tilde{\delta}/(d+1)$ as before, but we now split the δ' 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 $[\hat{P}_{\chi_1} - 2\alpha(\delta'/q), \hat{P}_{\chi_1} + 2\alpha(\delta'/q)], \ldots, [\hat{P}_{\chi_q} - 2\alpha(\delta'/q), \hat{P}_{\chi_q} + 2\alpha(\delta'/q)]$ respectively. We then go to the child corresponding to the answers we actually received, and as before give the child a confidence supply of $(\tilde{\delta} - \delta')/k$. Theorems 4.1 and 4.2 hold 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.

4.4.1 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 $\delta' = \tilde{\delta}/(d+1)$, then further divide δ' into confidences of size $\delta'/(F - d(v) + 1)$, 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, $\tilde{\delta} - \delta$ 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 $\hat{P}_\chi$ with a confidence of δ then the branching factor would be approximately $m \cdot \left[\hat{P}_\chi - 2\alpha(\delta) - (\hat{P}_\chi - 2\alpha(\delta))\right] = 4m \cdot \alpha(\delta)$. 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 $[4m \cdot \alpha(\delta)]^F$. This can be reduced to F or less using a batch query.

4.5 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 dataset 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 N implying that the value of $\epsilon(h)$ 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, $\sum_{i=1}^{d(h)} \ln |C_i(h)|$.

4.6 Alternative 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 (4) and so the term logarithmic in $d(h)$ in theorems 4.1 and 4.2 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 Comparison with Freund's Work

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.

5.1 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 simple sample complexity bound (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 3.1 and 3.2. 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 subchoice 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 $|\hat{C}_i|$.

6 Conclusion

The goal of this work is to produce tighter bounds on the future error rates of a learned hypothesis. We have presented the Microchoice bound which is a kind of Occam's razor bound using the structure of the learning algorithm itself. The Microchoice bound was then synthesized with Freund's query tree approach for Self-Bounding learning algorithms to produce the Adaptive Microchoice bound. We also presented some simple techniques for improving these bounds. One technique is a method for interpolating between the Microchoice bound and Freund's query tree approach using conglomeration of choice sets. Another technique is the batch query which is especially useful for decision trees. Some examples of the application of these bounds to decision trees were given along with a new pruning technique suggested by the Microchoice bounds.

The Microchoice bound can be applied to many common learning algorithms unintrusively — the path through state space that the algorithm takes need not be affected. Some extra computation may be necessary to assess the size of the choice set from which the algorithm makes its choices. In return, the algorithm can state a bound along with its hypothesis. The Microchoice approach also shows a very natural way of performing SRM.

In a given situation, the Microchoice bound may or may not be better than another PAC bound. In practice, it may be worthwhile to output a bound which is the minimum of that given by the Microchoice bound and a more general PAC bound. This can be done in a theoretically sound manner by first mapping $\delta \to \frac{\delta}{2}$ in each bound which slightly worsens the result. The resulting $\epsilon = \min(PAC(\delta/2), Microchoice(\delta/2))$ bound will then hold with probability $1 - \delta$.

There are several remaining open questions. Is there a satisfying, natural bound for the continuous case? Can this approach be useful for commonly used learning algorithms?

References

[BEHW87] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

[Dom98] Pedro Domingos. A process-oriented heuristic for model selection. In *Machine Learning Proceedings of the Fifteenth International Conference*, pages 127–135. Morgan Kaufmann Publishers, San Francisco, CA, 1998.

[Fre98] Yoav Freund. Self bounding learning algorithms. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 247–258. ACM Press, New York, NY, 1998.

[Kea93] Michael Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing*, pages 392–401. ACM Press, New York, NY, 1993.

[McA98] David A. McAllester. Some PAC-Bayesian theorems. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 230–234. ACM Press, New York, NY, 1998.

[Min89] Jon Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, 1989.

[Qui86] J.L. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[Riv87] R.L. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.

[STBWA96] John Shawe-Taylor, Peter Bartlett, Robert Williamson, and Martin Anthony. A framework for structural risk minimization. In *Proceedings of the 9th Annual Conference on Computational Learning Theory*, pages 68–88. ACM Press, New York, NY, 1996.