Microchoice Bounds and Self Bounding Learning Algorithms

John Langford
Computer Science Department
Carnegie Mellong University
Pittsburgh, PA 15217
jcl+@cs.cmu.edu

Avrim Blum*
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15217
avrim+@cs.cmu.edu

Abstract

We prove adaptive bounds for learning algorithms that operate by making a sequence of choices. These adaptive bounds, which we call Microchoice bounds, can be used to make these algorithms self-bounding in the style of Freund [Fre98]. Furthermore, we can combine these bounds with Freund's more sophisticated query-tree approach, producing a modified query-tree structure that yields similar bounds to those in [Fre98] but that permits a much more efficient algorithmic approximation.

1 Introduction

PAC bounds tend to be too loose for application to real world learning problems using common learning algorithms. They require more examples then expirementally necessary to guarantee a reasonable accuracy with a reasonable probability. Chief amongst the reasons for this failure is that typical PAC 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 where they take a sequence of steps each from a small (in comparison to the overall hypothesis space) space of choices. 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 space of possible steps to take. Taking into account the number of choices made and the size of each choice space can yield a tighter bound than the standard PAC bound if the algorithm halts before exploring a large portion of the available hypothesis space.

The Microchoice bound 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. Briefly, this bound allows algorithms to bound the difference between the true and empirical errors of the hypotheses produced based on the choices actually taken and the choice spaces actually explored, without needing to consider spaces that "might have been" visited under different circumstances. Thus, by keeping track of the sizes of the choice spaces visited, the bound can be calculated and output along with the answer. This procedure works without altering the final hypothesis produced by the 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]. In fact, the Microchoice bound can be viewed as a compelling choice of nested hypothesis spaces on which to apply Structural Risk minimization.

Self bounding Learning algorithms were first suggested by Yoav Freund [Fre98]. Work developing approximately Self Bounding Learning algorithms was also done by Pedro Domingos [Dom98]. 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 on the query tree that allows for algorithmic approximation in a much more efficient manner.

2 Discrete PAC bound Review

The basic PAC model is characterized by a hypothesis space, H, composed of hypotheses, $h : X \to Y$. We will be considering only discrete spaces 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 $z_i = (x_i, c(x_i))$. We use P to denote the distribution on labeled examples induced by D and c, and assume there is some underlying loss function $l : Y \times Y \to [0, 1]$ that describes the loss of predicting y when the true answer is y'. 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, in order to give us confi-

*Supported in part by NSF grant CCR-9732705

dence in algorithms that produce hypotheses of zero empirical error. The standard result, using the union bound, is that for any P, for any $\delta > 0$,

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

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

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 δ and split it evenly among the hypotheses, giving $\delta/|H|$ to each, then defining 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 m if we wish, giving $\delta(h) = \delta \cdot m(h)$ to each. We then define $\epsilon(h)$ such 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,*

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

$$\textit{for } \epsilon(h) = \frac{1}{N}\left(\ln\frac{1}{\delta} + \ln\frac{1}{m(h)}\right)$$

In the agnostic case, we simply 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,*

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

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

2.1 A motivating observation

For a given learning algorithm L, a distribution P on labeled examples induces a distribution $p(h)$ over the possible hypotheses $h \in H$ produced by algorithm L after N examples. If we were looking from the outside in and 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

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

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

Now notice that $\frac{\partial}{\partial m(h)} \ln\frac{1}{m(h)} = -\frac{\partial}{\partial m(h)} \ln m(h)$. This implies that the Lagrangian corresponding to minimizing the value of $\epsilon(h)$ will be the same up to a sign as the problem of minimizing $\sum p(h) \ln m(h)$. This corresponds to minimizing the Kullback-Liebler divergence which provides a solution of $m(h) = p(h)$.

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.

3 The Microchoice bound

Informally, the Microchoice bound is an a. posteriori bound based upon the following reasoning. "I have an a. priori bound on the true error in mind, but I do not have the means to compute it explicitly. What I can do is run the learning algorithm on the data available, and based upon the trace of the algorithm, compute a bound on the a. priori quantity."

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, $c_1, \ldots, c_d$, from a sequence of choice spaces, $C_1, \ldots, C_d$, finally producing a hypothesis, $h \in H$. Specifically, the algorithm first looks at the choice space C_1 and the data z^N to produce choice $c_1 \in C_1$. The choice c_1 then determines the next choice space C_2 (different initial choices produce different choice spaces 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 space C_3, and so on. These choice spaces 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 space C has branching factor $|C|$. Eventually, some choice space will be a *leaf*, containing no further choices and just a single hypothesis to be produced as output.

For example, the Decision List algorithm of Rivest [Riv87] applied to a space 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 space of $4n$ possible rules to put at the next level, then a choice space of size $4n - 2$, 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 spaces 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 spaces it has encountered so far.

Another way to view this process is that we can't 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 can't 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 space 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* $0 < \delta < 1$,

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

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

Proof. Specialization of the Realizable Folk Theorem.

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

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

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

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 spaces 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 $d \log n$ where d is the length of the list produced and n is the number of features. Notice that $d \log n$ is also the description length of the final hypothesis produced in the natural encoding, thus in this case these theorems yield similar bounds to Occam's razor or SRM.

More generally, the Microchoice bound is similar to Occam's razor or SRM kinds of 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 Another example

Decision trees over discrete spaces (say, $\{0,1\}^n$) 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 space 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 space size is $K(n-d)$ giving a bound noticeably worse then the bound for the decision list. This motivates a slightly different decision algorithm, one which considers only one leaf node at a time, adding a new test or deciding to never add a new test at this node. In this case, there are $(n - d(v) + 1)$ choices for a node v at depth $d(v)$ implying a bound of:

$$E_P(h) \le E(h, z^N)$$
$$+\sqrt{\frac{1}{2N}(\ln\frac{1}{\delta} + \sum_{v \in Nodes(h)} \ln(n - d(v) + 1)))}$$

which can be found by realizing that every node in h exists from a choice of some choice space.

4 Combining with Freund's Query Tree approach

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 is 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,

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

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 will be convenient to define

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

Thus, 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 $|\hat{P}_\chi - P_\chi| \leq \alpha$. If a particular predicate, χ is true with probability $.9$ 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 $2|Q(T_A(P, \alpha))|e^{-2N\alpha^2}$. In other words, this is an upper bound on the probability the algorithm ever "falls off the tree". The point of this is that one 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 space).

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 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, continuing recursively down the tree.

A drawback of this method is that it can easily take exponential time to produce the approximate tree (exponential in the running time of the original 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 will allow 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 in a sense to finesse that issue.

In order to apply the Microchoice approach, we will 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 to the choice tree are now reasonably straightforward. We begin by describing the *true* pruned tree and then giving 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 $\hat{P}_\chi$ such that $|\hat{P}_\chi - P_\chi| \leq \alpha(\delta')$. 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. In the ideal tree, we simply choose δ' so as to maximize the amount of confidence parameter given to each (unpruned) child. For example, if the world is nice and it is possible using $\delta' \leq \tilde{\delta}/2$ for the node to have only one unpruned child, then δ' will be the minimum value such that this occurs.

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 will not actually create the tree; instead we will 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 $(\tilde{\delta} - \delta')/k$. I.e., this is the same as we would have given it had we allocated the remaining $\tilde{\delta} - \delta'$ 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 $E(h, z^N) + \epsilon(h)$, where $\epsilon(h)$ is the error term corresponding to the confidence supply $\delta(h)$ remaining.

We now prove that with probability $1 - \delta$, our error estimate $E(h, z^N) + \epsilon(h)$ is correct. To do this, it helps to define a modified version of the true query tree in which at each node, the parameter δ' is required to be $\delta' = \tilde{\delta}/(d(\chi) + 1)$. With probability $1 - \delta$, all queries in *that* tree receive good answers, where "good" is defined with respect to the α parameters in that tree, *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 modified true query tree. In particular, assume inductively that at the current node of our empirical path the supply $\tilde{\delta}_{emp}$ is no greater than the supply $\tilde{\delta}_{true}$ given to that node in the (modified) true tree. Now, under the assumption that the response $\hat{P}_\chi$ is "good", it must be the case that the interval $[\hat{P}_\chi - 2\alpha(\tilde{\delta}_{emp}/(d+1)), \hat{P}_\chi + 2\alpha(\tilde{\delta}_{emp}/(d+1))]$ contains the interval $[P_\chi - \alpha(\tilde{\delta}_{true}/(d+1)), P_\chi + \alpha(\tilde{\delta}_{true}/(d+1))]$. Therefore, the supply given to our child in the empirical path is no greater than the supply given in the (modified) true tree.

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

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

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 $0 < \delta < 1$,*

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

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

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 n queries, one for each variable; we then choose the variable 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. (In particular, if we required each node in the query tree to contain just a single query, this could result in an unfortunately high branching factor just for the purpose of "remembering" the answers received so far.)

Extending the algorithmic construction to allow for batch queries is easily done. If a node has b queries $\chi_1, \ldots, \chi_b$, we choose $\delta' = \tilde{\delta}/(d+1)$ as before, but we now split the δ' evenly among all b queries. We then let k be the number of children corresponding to answers to the queries $\chi_1, \ldots, \chi_b$ in the ranges $[\hat{P}_{\chi_1} - 2\alpha(\delta'/b), \hat{P}_{\chi_1} + 2\alpha(\delta'/b)], \ldots, [\hat{P}_{\chi_b} - 2\alpha(\delta'/b), \hat{P}_{\chi_b} + 2\alpha(\delta'/b)]$ 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 ith choice space in the batch tree rather than the size in the single-query-per-node tree.

4.4.1 Example: Decision trees

When growing a decision tree, it is natural to make a batch of queries then make a decision about which feature to place in a node, repeating the process to grow the tree structure. As in the decision tree example above, if we have n features and are considering adding a node at depth $d(v)$ there are $n - d(v) + 1$ possible features which could be chosen for placement in a particular node. The decision of which feature to use is made by comparing the results of $n - d(v) + 1$ 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 $\delta'/(n - d(v) + 1)$, placing each divided confidence on one of the $n - d(v) + 1$ statistical queries. We now may be able to eliminate some of the $n - d(v) + 1$ 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 space. 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 space will be of size 1 if there are enough examples.

5 Comparison with Freund's work

Freund's approach for selfbounding 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 subspace from which a choice is made. Because many common learning algorithms work by a process of making choices from small subspaces, this is often easy computationally. 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 somewhat between the Adaptive Microchoice bound and Freund's bound by a process of conglomerating the the subspaces of the Microchoice bound. Consider a particular choice space, $\hat{C}_i$ with elements, c_i. Each c_i indexes another choice space, $\hat{C}_{i+1}(c_i)$. If the computational resources exist to calculate $\hat{C}_{i,i+1} = \cup_{c_i \in \hat{C}_i} \hat{C}_{i+1}(c_i)$, then $|\hat{C}_{i,i+1}|$ can be used in place of $|\hat{C}_i||\hat{C}_{i+1}|$ in the adaptive microchoice bound. In the extreme case where all $\hat{C}_{i+1}(c_i)$ are identical, this is like the batch case.

6 Conclusion

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

It is not clear the the Microchoice bound will always be better than a typical PAC bound when applied to a problem. 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 theoretically by first mapping $\delta \rightarrow \frac{\delta}{2}$ in each bound, slightly worsening the result. The resulting Min(PAC,Microchoice) bound will then hold with probability $1 - \delta$.

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

References

[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.

[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.