Self bounding learning algorithms

Yoav Freund
AT&T Labs
180 Park Avenue
Florham Park, NJ 07932-0971 USA
yoav@research.att.com

January 17, 2000

Abstract

Most of the work which attempts to give bounds on the generalization error of the hypothesis generated by a learning algorithm is based on methods from the theory of uniform convergence. These bounds are a-priori bounds that hold for any distribution of examples and are calculated before any data is observed. In this paper we propose a different approach for bounding the generalization error after the data has been observed. A self-bounding learning algorithm is an algorithm which, in addition to the hypothesis that it outputs, outputs a reliable upper bound on the generalization error of this hypothesis. We first explore the idea in the statistical query learning framework of Kearns [10]. After that we give an explicit self bounding algorithm for learning algorithms that are based on local search.

1 INTRODUCTION

Most of the work on the sample complexity of learning is based on uniform convergence theory and attempts to give *uniform a-priori bounds*. A uniform a-priori bound is a guarantee that, with high probability over the training set, the difference between the training error and the test error is uniformly small for all hypotheses in a given class. Such bounds are attractive because they allow us to argue about *any* algorithm that can find a hypothesis with a small training error. However, the bounds that this theory yields are almost always much too pessimistic. The gap between training error and test error of learning algorithms that are used in practice is usually much smaller than predicted by this theory.

In this work we propose a way by which one can derive better bounds on the generalization error in specific situations. The bounds depend on everything: the input distribution, the relation between the input and the output and the learning algorithm. In this sense the bounds are weaker than those of uniform convergence theory. On the other hand,the bounds that we derive are in some cases tighter. We give an *algorithm*

for calculating the bound as a function of the training data. We call an algorithm that combines the learning algorithm with this bound calculating algorithm a *self-bounding* learning algorithm. We are thus replacing the a-priori guarantees of the uniform convergence theory with *a-posteriori* guarantees which provide a constructive way of calculating a reliable bound on the generalization error.

To see how this idea might be used consider the problem of bounding the generalization error of a decision tree generated by one of the practical learning algorithms for decision trees, such as C4.5 [13] or CART [5]. Bounds of this type can be used as part of a formally justified pruning method. If we base our analysis on structural risk minimization (see, for example, Mansour [11]), then we use the a-priori bound which is based on the number of all possible decision trees with a given number of nodes. However, if we take into account that we used the algorithm C4.5 to generate the tree we realize that taking into account *all* possible trees might sometimes be a gross overestimate. C4.5 proceed by repeatedly splitting the leaves of the decision tree. At each step choosing the split that seems best according to some measure of quality. Suppose now that the training data is such that at each step, one split is significantly better than the rest. In this case it is very unlikely that a different training set will change the tree, which means that in our calculations of the bound we really should be considering only a single tree! In this paper we formalize what we mean when we say that one of the splits is significantly better than the rest and bound the size of the sample that is needed to achieve such significance.

In the first part of the paper we explore the idea of self-bounding learning algorithms in the general context of the statistical query (SQ) learning model of Kearns [10]. We describe a general transformation of deterministic SQ learning algorithms into self-bounding algorithms. However, self bounding algorithms generated by this transformation would, in general, require exponentially more time and space than the original learning algorithm.

In the second part of the paper we restrict our attention to one important subset of SQ learning algorithms. This is the family of local search algorithm. This family includes gradient-descent algorithms such as "BackProp" [14] and algorithms that work by iteratively altering their model, such as the work of Heckerman et. al. [8] on learning graphical models by repeated local changes. We describe a transformation of local-search learning algorithms into self-bounding algorithms. This transformation produces a self bounding learning algorithm whose computational complexity is similar to the computational complexity of the learning algorithm on which it is based.

1.1 RELATION TO OTHER WORK

A-posteriori error bounds of the type we consider here were previously considered by Shaw-Taylor et. al. [15]. In their work they expand Vapnik's SRM framework to allow for structures that depend on the training data. Their analysis yields bounds on the difference between the training error and the true error that can be quantified only after the data has been observed. An important application of their method are the a-posteriori bounds that they derive on Vapnik's support vector machines. The difference

between this work and theirs is that our starting point is a given learning algorithm while theirs is a so-called "luckiness function", which is, in the case of support vector machines, the size of the margin. Other research that considers a-posteriori bounds include Shawe-Taylor and Williamson [16] and McAllester [12]. These analysis use as a starting point a prior distribution over the concept space.

Learning for specific distributions has been studied quite extensively, for examples see Benedeck and Itai [2] and Haussler et. al [7]. However, these works derive bounds that depend explicitly on the input distribution. This makes them useless for problems of model selection, because the input distribution is not known and estimating it is usually a harder problem than approximating the input-output relationship.

Another approach to estimating the test error of a hypothesis is cross validation. In this approach some of the training data is left outside the reach of the learning algorithm and is later used to estimate the generalization error of the final hypothesis. The advantage of our approach over cross validation is that the learning algorithm can use all of the training data.

2 ERROR BOUNDS FOR SQ ALGORITHMS

Kearns [10] introduced the statistical query (SQ) model of learning. This model is a restriction of the PAC learning model. Most of the known concept learning algorithms can be analyzed within this framework. One of the advantages of this model is that it provides a general way of transforming any learning algorithm into a version that can learn in the presence of classification noise. For our needs here we define a slightly different class of learning algorithms, which we call *Encapsulated Statistical Query* learning algorithm or **ESQ** for short.[1]

An encapsulated learning algorithm $\mathbf{A}$ consists of two parts, the first is an SQ algorithm, denoted $\mathbf{A}_{SQ}$; the second is the *statistical oracle*, denoted STAT. The oracle serves as an intermediary between the SQ learning algorithm and the training data. In order to be precise, we define the following setup. We assume that *examples* are items of the form $(x, y) \in X \times \{0, 1\}$ and are randomly generated according to some fixed but unknown distribution $\mathcal{D}$. The goal of learning is to find a function, also called the *hypothesis*, $h : X \to \{0, 1\}$, chosen from a hypothesis class H, which minimizes the *generalization error*:

$$\mathrm{err}(h) \doteq \mathbf{P}_{(x,y)\sim\mathcal{D}}\left[h(x) \neq y\right] .$$

In this paper we restrict ourselves to the case where H is a finite set. The **ESQ** learning algorithm receives as input a *training set*, which is a set of examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, drawn independently at random according to $\mathcal{D}$. The training set is accessible only to the oracle, not to the SQ algorithm. The SQ algorithm communicates with the oracle by making *statistical queries*. A statistical query is a

[1] Our definition is essentially identical to the standard simulation of a PAC learning algorithm using an SQ algorithm as described in Kearns [10] and Aslam and Decatur [1]. However, as our emphasis here is on a different set of questions, we find it better to establish a slightly different terminology.

description of a binary predicate χ which is an element of the *query class* $\mathcal{Q}$. Again, in this paper, we consider only finite query classes. The binary predicate maps instance and label pairs $(x, y) \in X \times \{0, 1\}$ into $\{0, 1\}$. The answer provided by STAT to the statistical query is a a real number $0 \leq \hat{P}_\chi \leq 1$ which is an approximation of the expected value of the predicate:

$$P_\chi = \mathbf{E}_{(x,y)\sim\mathcal{D}} \left[\chi(x, y)\right] \ .$$

We say that the approximation is α-accurate if $|\hat{P}_\chi - P_\chi| \leq \alpha$ for some $\alpha > 0$. The oracle STAT generates its answers, given the sample S, as follows:

$$\hat{P}_\chi = \text{STAT}\left(S, \chi, S\right) = \frac{1}{m}\sum_{i=1}^{m} \chi(x_i, y_i) \ . \tag{1}$$

Note that the *same* sample is used for answering all of the queries. As the sample size is m the answers of the oracle are always of the form i/m for some integer $0 \leq i \leq m$. We require that the SQ learning algorithm is a deterministic algorithm (given the answers to its queries), and that it always halts after a finite amount of time and outputs a hypothesis h^* in H.

In this paper our goal is *not* to characterize hypotheses classes which are "learnable", i.e. for which there exist an **ESQ** algorithm that can efficiently generate an hypothesis whose generalization error is small. Instead, our goal here is to give better bounds on the generalization error of h^* for a *given* algorithm $\mathbf{A}$ and a *fixed* distribution $\mathcal{D}$.

A natural estimator of $\text{err}(h^*)$ is the number of mistakes it makes on the training set:

$$\widehat{\text{err}}(h^*) = \frac{1}{m}\left|\{1 \leq i \leq m \text{ such that } h(x_i) \neq y_i\}\right| \ .$$

This estimator has non-zero bias and variance. We would like to have some reliable bound on its accuracy. More precisely, we are seeking a procedure that, given a reliability parameter $\delta > 0$, generates a real number $\epsilon > 0$ such that the probability of a training sample S of size m for which $\text{err}(h^*) > \widehat{\text{err}}(h^*) + \epsilon$ is smaller than δ.

Before we describe our method for generating such bounds. We consider two existing approaches for generating *a-priori* bounds. In the standard analysis, the difference between the estimated error and the true error is bounded using uniform convergence arguments. Most of these bounds depend only on the complexity of the hypothesis class H. As we restrict ourselves here to finite hypotheses classes, we can use the simple application of Hoeffding's bounds [9] (see for instance Blumer et. al [3]) and get that

$$\mathbf{P}_{S\sim\mathcal{D}^m}\left[\text{err}(h^*) - \widehat{\text{err}}(h^*) > \epsilon\right] \leq |H| e^{-2m\epsilon^2} \ . \tag{2}$$

A different approach is used by Kearns in [10]. There the assumption is made that if the answers to all the statistical queries are α-accurate then the generalization error of the hypothesis output by the SQ learning algorithm is guaranteed to be smaller than

$\epsilon(\alpha)$. An argument similar to the one used for Equation 2 can be applied to show that

$$\begin{aligned}
&\mathbf{P}_{S\sim\mathcal{D}^m}\left[\text{err}(h^*) > \epsilon(\alpha)\right] \\
&\leq \quad \mathbf{P}_{S\sim\mathcal{D}^m}\left[\exists\chi\in\mathcal{Q}||\hat{P}_\chi - P_\chi| > \alpha\right] \\
&\leq \quad 2|\mathcal{Q}|e^{-2m\alpha^2}\,. \qquad (3)
\end{aligned}$$

The advantage of the bounds given in Equations (2) and (3) is that the probability of failure can be calculated before the sample is observed and without making any assumption about the distribution $\mathcal{D}$. On the other hand, this same property is also the reason for a main disadvantage of these bounds, which is that they are usually overly pessimistic. The reason is that for any particular distribution $\mathcal{D}$ there might exist a much better bound on the error. While the information that we get about $\mathcal{D}$ from the sample S is very partial it might still be used to improve the bound. In the rest of this section we show how this can be done for general deterministic **ESQ** algorithms.

Let **A** be an **ESQ** learning algorithm. Consider the set of states of **A** in which it either makes a statistical query or outputs a hypothesis and stops, we call such states query-states and output-states respectively. We use s to denote a particular state, $\chi(s)$ to denote the query associated with a query state and $h(s)$ to denote the hypothesis associated with an output state. We now define the query-tree $\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha)$. The query tree is a structure that represents the possible ways that the **ESQ** algorithm **A** might run for a given distribution $\mathcal{D}$ and accuracy $\alpha > 0$. The internal nodes of this tree are associated with query-states and the leaves are associated with output-states. The root of the tree corresponds to the first query made by the algorithm. Given a query χ the desired accuracy α and the distribution $\mathcal{D}$ we define the following finite set of rational fractions as the set of "legal answers":

$$L(\chi,\alpha) \doteq \{\hat{P}_\chi = i/m \text{ such that } |i/m - P_\chi| \leq \alpha\}\,.$$

The children of a query state s in which the query χ is made consist of query states or output states that immediately follow the state s if $\hat{P}_\chi \in L(\chi(s),\alpha)$. A recursive procedure that calculates $\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha)$ is given in Figure 1.

We call the *set* of queries in $\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha)$ the *query span* of **A** and denote it by $\text{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))$.[2] Similarly, we call the set of final hypotheses in $\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha)$ the *hypothesis span* and denote it by $\text{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))$. Using these definitions we can get the following improved bound on the generalization error of any **ESQ** learning algorithm.

Theorem 1 *Let* **A** *be an* **ESQ** *learning algorithm that receives a training sample S and outputs a final hypothesis h^*. Assume that S consists of m examples, drawn independently at random from the distribution $\mathcal{D}$ over $X\times\{0,1\}$.*

Then $\forall\alpha > 0, \forall\epsilon > 0$, the generalization error of the final hypothesis is bounded by

$$\mathbf{P}_{S\sim\mathcal{D}^m}\left[err(h^*) - \widehat{err}(h^*) > \epsilon\right]$$

[2] Note that it is quite possible that the same query appears in several places in the tree, in which case the size of the query span will be smaller than the number of internal nodes in the query tree.

Input:
A: An **ESQ** learning algorithm
$\mathcal{D}$: A distribution over the space of labeled examples.
m: The size of the training set
α: A reliability parameter.

Output:
The query tree $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$, of **A**, when started in state s.

Procedure **GenerateTree** $(\mathbf{A}, \mathcal{D}, s, \alpha, m)$

1. If s is an output state, return the leaf node s.
2. Otherwise, let s be the root of a tree.
3. Let P_χ be the "correct" answer to the query $\chi(s)$:
$$P_\chi = \mathbf{E}_{(x,y)\sim\mathcal{D}}\left[\chi(x,y)\right] \ .$$
4. For each integer j such that
$P_\chi - \alpha \leq j/m \leq P_\chi + \alpha$ do:
   (a) Let $s' = \text{NEXT}\,(\mathbf{A}, s, j/m)$
   (b) If s' is equal to one of the existing children of s, skip to the next loop iteration.
   (c) Add the tree **GenerateTree** $(\mathbf{A}, \mathcal{D}, s', \alpha, m)$ as a child of s.
5. Return the tree rooted at s.

Figure 1: A recursive definition of the query tree rooted at the state s of an **ESQ** learning algorithm **A**. The tree $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ is calculated by a call to **GenerateTree** with the first query state of **A** as input. The notation NEXT $(\mathbf{A}, s, a)$ represents the state which **A** reaches following s if the answer for the query $\chi(s)$ it issues in state s is $a \in [0, 1]$.

$$\leq \quad 2|Q(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))|e^{-2m\alpha^2} + |H(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))|e^{-2m\epsilon^2} \ .$$

Proof: From Hoeffding's bound we get that

$$\begin{aligned} &\mathbf{P}_{S\sim\mathcal{D}^m}\left[\exists\chi \in \mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha)) : |\hat{P}_\chi - P_\chi| > \alpha\right] \\ &\leq \quad 2|\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))|e^{-2m\alpha^2} \ . \end{aligned}$$

This gives the first term in the bound. On the complement event all the estimates for all queries in $\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))$ are in $L(\chi,\alpha)$. In this case, by definition, all the queries made by the algorithm are in $\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha)$ and $h^* \in \mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))$. Using Hoeffding's bound a second time we get that the probability that $\mathrm{err}(h^*) - \widehat{\mathrm{err}}(h^*) > \epsilon$ given that $\hat{P}_\chi \in L(\chi,\alpha)$ for all the queries asked is at most

$$|\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))|e^{-2m\epsilon^2}$$

which gives the second term in the bound, completing the proof of the theorem. ■

Comparing this bound to the bound given in Equations (2) and (3) we find that if $\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha))$ is much smaller than $\mathcal{Q}$ and if $\mathrm{Q}(\alpha)$ is not too large we get a much smaller probability of failure. Equivalently, if we fix the probability of failure, then under the same conditions, Theorem 1 guarantees a smaller value of ϵ than Equation (2).

We return to the example discussed in the introduction regarding generalization bounds for C4.5. Let us assume, for the sake of simplicity, that the input consists of n binary features and that the depth of the tree is d. Assume also that if two splits have the same quality then the algorithm has some fixed rule for choosing one of the splits over the other. In most practical decision tree algorithms, such as CART and C4.5, the quality of a split can be calculated as a continuous function of the answers to a small number of statistical queries. The queries are of the form: "What is the probability of a positive/negative/any example reaching leaf X?". More formally, the predicates for these queries are conjunctions of at most d out of the n binary features or their negation, together with a possible condition on the label, thus the number of possible queries in $\mathcal{Q}$ is $O(2^d d^n)$. Let us call the "ideal" decision tree the tree that is generated if $\alpha = 0$ i.e., if $\hat{P}_\chi = P_\chi$ for all the queries. As the quality measures are continuous functions of the answers to the queries, there must exist some sufficiently small $\alpha_m > 0$ which guarantees that the algorithm generates the ideal tree. In this case $|\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha_m))| = 1$. If we use the bound $\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D},\alpha_m)) \subseteq \mathcal{Q}$ we get the following bound on the probability of failure:

$$e^{-2m\epsilon^2} + O(2^d d^n)e^{-2m\alpha_m^2} \ .$$

But we can do even better, as the number of queries involved in generating one particular decision tree is $O(2^d n)$ the bound can be improved to

$$e^{-2m\epsilon^2} + O(2^d n)e^{-2m\alpha_m^2} \ . \tag{4}$$

Compare this bound to the bound that we get by using Equation (2). The number of decision trees of depth d over n binary features is $O(n^{(2^d)})$, thus we get that the probability of failure is at most:

$$n^{(2^d)} e^{-2m\epsilon^2} \ . \tag{5}$$

Of course, whether the bound in (4) is better or worse than the bound in (5) depends on the value of α_m and the value of α_m depends on the distribution of the data. If the decisions made while growing the tree are close to deterministic then the value of α_m is large and we get a superior bound. In practice, we would like to choose α and ϵ to balance the two terms in Theorem 1.

How can we use the bound given in Theorem 1 in practice? Even if we have direct access to the learning algorithm **A** it seems that we need to identify the distribution $\mathcal{D}$ before we can calculate $\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha_m))$ and apply the theorem. It thus seems we have ended up with a problem that is harder than the original learning problem, which was to approximate the input-output relationship!

However, while in most cases, calculating $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ exactly is hard, it is not as hard, in principle, to calculate a superset of $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$. Similarly to the definition of $L(\chi, \alpha)$, let $\hat{L}(\chi, \alpha)$ be the segment $[\hat{P}_\chi - \alpha, \hat{P}_\chi + \alpha]$ which *can* be calculated using only the training set S. Let us denote by $\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha)$ the tree expansion of the algorithm where the answer to each query is any number in $\hat{L}(\chi, \alpha)$ rather than $L(\chi, \alpha)$. This tree can be computed by a procedure very similar to **GenerateTree** $(\mathbf{A}, \mathcal{D}, s, \alpha, m)$, described in Figure 1, where the distribution $\mathcal{D}$ is replaced with the sample S and the exact answer P_χ is replaced by the approximate answer $\hat{P}_\chi = \mathrm{STAT}(S, \chi)$ as defined in Equation (1). The following lemma describes the sense in which $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha)$ is an approximation of $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$. We say that the tree $\mathcal{T}_1$ is a *pruning* of the tree $\mathcal{T}_2$ if both trees have the same root and the nodes of $\mathcal{T}_1$ are a subset of the nodes of $\mathcal{T}_2$. Clearly, if $\mathcal{T}_1$ is a pruning of $\mathcal{T}_2$, then $\mathrm{H}(\mathcal{T}_1) \subseteq \mathrm{H}(\mathcal{T}_2)$ and $\mathrm{Q}(\mathcal{T}_1) \subseteq \mathrm{Q}(\mathcal{T}_2)$

Lemma 2 *For any* **ESQ** *algorithm* **A**, *any distribution* $\mathcal{D}$ *and any* $\alpha > 0$, *if for all* $s \in \mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ $\hat{P}_\chi \in L(\chi, \alpha)$, *then* $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ *is a pruning of* $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha)$

Proof: Consider the roots of the query trees $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ and $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha)$. Both roots correspond to the initial state s_0. From the assumptions we get that the answer to the query $\chi(s_0)$ satisfies $\hat{P}_\chi \in L(\chi(s_0), \alpha)$. Thus $\hat{P}_\chi - 2\alpha \leq P_\chi + \alpha - 2\alpha = P_\chi - \alpha$ and $\hat{P}_\chi + 2\alpha \geq P_\chi - \alpha + 2\alpha = P_\chi + \alpha$. Thus $\hat{L}(\chi, \alpha) \subset L(\chi, \alpha)$. As a result the children of s_0 in $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ are a subset of the children of s_0 in $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha)$.

Applying the same argument now to the children of s_0 in $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ and continuing inductively completes the proof. ■

As $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha) \subseteq \widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha)$, the hypothesis and query spans of $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha)$ are supersets of the corresponding spans for $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha)$ and we can use them to calculate a bound similar to the one given in Theorem 1.

In Figure 2 we describes **SB-ESQ** - a general self-bounding algorithm for **ESQ** learning algorithms. **SB-ESQ** starts by calling **A** with the sample S to calculate the

Input:
A : An **ESQ** learning algorithm
$S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$: a training set
N_0 : An upper bound on the size of the query space $|\mathcal{Q}|$.
$\delta =$ A reliability parameter.

Output:
h^* = a hypothesis.
error-bound = an upper bound on err(h^*).

Globally defined objects: The sets H and Q.

Main Algorithm:

1. Call **A** with sample S to generate the final hypothesis h^*.
2. Calculate $\widehat{\mathtt{err}}(h^*)$ using the sample S.
3. Set
$$\alpha_1 = \sqrt{\frac{1}{2m} \ln \frac{4N_0}{\delta}} \quad .$$
4. For $i = 2, 3, \ldots$ until $\alpha_i = \alpha_{i-1}$
   (a) Calculate $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_{i-1})$
   (b) Set
$$\alpha_i = \sqrt{\frac{1}{2m} \ln \frac{4|\mathtt{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_{i-1}))}{\delta}} \quad .$$
5. Output the error bound:
$$\widehat{\mathtt{err}}(h^*) + \sqrt{\frac{1}{2m} \ln \frac{2|\mathtt{H}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i))}{\delta}}$$

Figure 2: the self bounding learning algorithm **SB-ESQ**.

final hypothesis h^*, it then calculates the training error of that hypothesis. The remainder of the algorithm is devoted to calculating a bound on the difference between the training error and the generalization error. In order to calculate the bound the algorithm needs to receive, as input, an upper bound N_0 on the size of the set of possible queries. In the lack of any additional information we use the size of the query space i.e. set $N_0 = |\mathcal{Q}|$. Given N_0 the algorithm iteratively calculates bounds $\alpha_1, \alpha_2, \ldots$ on the errors of the answers to the algorithm's queries. Eventually the bounds converge and the loop is exited, as is proven in the following lemma

Lemma 3 *For any setting of the inputs of* **SB-ESQ**, *and for each* $i \geq 1$ $\alpha_{i+1} \leq \alpha_i$ *and* $\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_{i+1})$ *is a pruning of* $\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_i)$. *In addition there exists a finite value of* i *for which* $\alpha_i = \alpha_{i-1}$.

Proof:

From the assumption that $N_0 \geq |\mathcal{Q}|$ we get that $|\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_1))| \leq N_0$ and thus that $\alpha_2 \leq \alpha_1$. It is easy to verify that this implies that $\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_2)$ is a pruning of $\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_1)$. Thus $|\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_2))| \leq |\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_1))|$ and thus $\alpha_3 \leq \alpha_2$. We can continue this argument by induction to prove the first part of the lemma.

For the second part, note that $|\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_i))|$ is a non-negative integer and a non-increasing sequence of non-negative integers must converge in a finite amount of time. Thus for some finite i $|\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_{i-1}))| = |\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, \alpha_i))|$ and $\alpha_i = \alpha_{i+1}$. ■

Given the final bound α_i and the corresponding tree $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i)$, **SB-ESQ** calculates H, which is, with high probability, a superset of the set of hypothesis from which h^* is likely to be chosen. Given the size of H, **SB-ESQ** calculates the bound on the generalization error of h^*.

From Lemma 3 we know that the query tree that is generated by **SB-ESQ** on iteration $i + 1$ is a pruning of the tree generated on iteration i. This implies that if the loop on command 4 is stopped before convergence is reached the result is a larger query tree, a larger hypothesis set H and thus an inferior bound. However, if the computation time is limited, then it might be worthwhile to stop the loop before convergence and use the inferior bound.

In what sense can we say that the bound that is output by the algorithm is reliable? We say that a self bounding learning algorithm is *sound* if the probability of training samples on which the bound that is output by the algorithm is incorrect can be made arbitrarily small. Formally, we use the following definition:

Definition 4 *A self bounding learning algorithm* A *is called* sound *if for any distribution* $\mathcal{D}$ *over labeled examples, for any sample size* m *and for any* $\delta > 0$ *the following event has probability at most* δ *over the random draws of the training set* S.

The event is: "The algorithm A*, given the training set* S*, outputs a hypothesis* h^* *and a bound* b *such that* $b < err(h^*)$*"*

We now state and prove the main theorem of the paper

Theorem 5 *The self-bounding learning algorithm* **SB-ESQ** *is sound.*

To prove the theorem we need to first define the value η_*. The value of η_* is closely related to the values of the random variables $\alpha_1, \alpha_2, \ldots$. However, it is defined in terms of the actual distribution $\mathcal{D}$ and not the sample S, and is thus *not* a random variable in the context of Theorem 5. This is the critical technical property of η_* that we use in the proof of the theorem. The following Lemma defines η_* and proves its existence.

Lemma 6 *For any* **ESQ** *learning algorithm* **A***which receives as inputs any constants m and δ, and any distribution of examples $\mathcal{D}$ there exists a real number $0 \leq \eta_* \leq 1$ such that*

$$\eta_* = \sqrt{\frac{1}{2m} \ln \left(\frac{4|Q(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))|}{\delta} \right)} .$$

Proof: We define a sequence of positive reals as follows

$$\eta_1 = \sqrt{\frac{1}{2m} \ln \left(\frac{4|\mathcal{Q}|}{\delta} \right)} .$$

Given the value of η_i for some $i \geq 1$ we define η_{i+1} as follows:

$$\eta_{i+1} = \sqrt{\frac{1}{2m} \ln \left(\frac{4|\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_i))|}{\delta} \right)} .$$

Replacing α_i by η_i in the proof of Lemma 3 we find that this is a non-increasing sequence that converges after a finite amount of steps. Thus $\eta_i = \eta_{i+1}$ for some finite value of i. Setting $\eta_* = \eta_{i+1}$ we get the statement of the lemma. ■

Note that

$$2|\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))|e^{-2m\eta_*^2} = \delta/2 .$$

We now prove that, with high probability, the estimates α_i generated by **SB-ESQ** are all lower bounded by η_*.

Lemma 7 *If the training set S is such that for all $\chi \in Q(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))$, $|\hat{P}_\chi - P_\chi| \leq \eta_*$*
then for all $i \geq 1$,$\alpha_i(S) \geq \eta_*$ and $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))$ is a pruning of $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i(S)))$

Proof: We prove the lemma by induction over i, similar to the one used to analyze the sequence η_i in the proof of Lemma 6. We use the notation η_i defined in that proof. We denote by $\alpha_i(S)$ the value of α_i that is generated by **SB-ESQ** when its input is S.

Recall the definitions of $\alpha_1(S)$ and η_1:

$$\eta_1 = \sqrt{\frac{1}{2m} \ln \left(\frac{4|\mathcal{Q}|}{\delta} \right)} ,$$

and

$$\alpha_1(S) = \sqrt{\frac{1}{2m} \ln \left(\frac{4N_0}{\delta} \right)} ,$$

As $N_0 \geq |\mathcal{Q}|$ we get that $\alpha_1(S) \geq \eta_1 \geq \eta_*$.

Assuming that $\alpha_i(S) \geq \eta_*$ we show that $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)$ is a pruning of $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i(S))$ and that $\alpha_{i+1}(S) \geq \eta_*$.

From the assumption that for all $\chi \in \mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))$, $|\hat{P}_\chi - P_\chi| \leq \eta_*$, combined with Lemma 2 it follows that $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)$ is a pruning of $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\eta_*)$. On the other hand, from the induction assumption we have that $\alpha_i(S) \geq \eta_*$ which implies that $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\eta_*)$ is a pruning of $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i(S))$. Thus $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)$ is a pruning of $\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i(S))$ which implies that $|\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, \eta_*))| \leq |\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i(S)))|$.

Combining the last inequality with the definitions of η_* and $\alpha_{i+1}(S)$:

$$\eta_* = \sqrt{\frac{1}{2m} \ln \left(\frac{4|\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))|}{\delta} \right)} \, ,$$

and

$$\alpha_{i+1}(S) = \sqrt{\frac{1}{2m} \ln \frac{4|\mathrm{Q}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i(S)))|}{\delta}} \, ,$$

we get that $\alpha_{i+1}(S) \geq \eta_*$ thus proving the next step of the induction and completing the proof of the lemma. ■

We now have all the ingredients we need in order to prove the main theorem of this section:

Proof of Theorem 5: We bound the probability that the error bound generated by **SB-ESQ** , denoted b is incorrect by separating this event into two as follows:

$$\begin{aligned} & \mathbf{P}_{S \sim \mathcal{D}^m} [b > \mathrm{err}(h^*)] \qquad (6) \\ & \quad \leq \quad \mathbf{P}_{S \sim \mathcal{D}^m} \left[\exists \chi \in \mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)) \text{ s.t. } |\hat{P}_\chi - P_\chi| \geq \eta_* \right] \\ & + \mathbf{P}_{S \sim \mathcal{D}^m} \left[b > \mathrm{err}(h^*) | \forall \chi \in \mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)), |\hat{P}_\chi - P_\chi| < \eta_* \right] \end{aligned}$$

We first bound the probability of the first term. Using Hoeffding's bound we get that this probability is bounded by

$$\begin{aligned} & \mathbf{P}_{S \sim \mathcal{D}^m} \left[\exists \chi \in \mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)) \, \middle| \, |\hat{P}_\chi - P_\chi| \geq \eta_* \right] \\ & \quad \leq \quad 2|\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))| e^{-2m\eta_*^2} = \delta/2 \, . \qquad (7) \end{aligned}$$

Next we bound the second term. From the definition of $\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)$ together with the assumption that $\forall \chi \in \mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)) \, |\hat{P}_\chi - P_\chi| < \eta_*$ we get, by using Hoeffding's bound a second time that for any $\epsilon > 0$

$$\begin{aligned} & \mathbf{P}_{S \sim \mathcal{D}^m} [\mathrm{err}(h^*) \geq \widehat{\mathrm{err}}(h^*) + \epsilon] \qquad (8) \\ & \quad \leq \quad \mathbf{P}_{S \sim \mathcal{D}^m} \left[\exists h \in \mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)) \text{ s.t. } \mathrm{err}(h) > \widehat{\mathrm{err}}(h) + \epsilon \right] \\ & \quad \leq \quad |\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))| e^{-2m\epsilon^2} \end{aligned}$$

On the other hand, from the assumption that $\forall \chi \in \mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))\ |\hat{P}_\chi - P_\chi| < \eta_*$ together with Lemma 7 we get that $|\mathrm{H}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i))| \geq |\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))|$. If we set ϵ in Equation (8) to $\sqrt{(1/2m)\ln(2|\mathrm{H}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i))|/\delta)}$ we get that

$$\begin{aligned} & \mathbf{P}_{S \sim \mathcal{D}^m}\left[\mathrm{err}(h^*) \geq \widehat{\mathrm{err}}(h^*) + \epsilon\right] \\ \leq\ & |\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))| \frac{\delta}{2|\mathrm{H}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i))|} \leq \delta/2\ . \end{aligned} \tag{9}$$

We have thus bound both terms of the RHS of Equation (6) by $\delta/2$, which completes the proof of the theorem. ■

2.1 DISCUSSION

How good is the bound generated by **SB-ESQ**? It is easy to see that as $\mathrm{H}(\widehat{\mathcal{T}}_{\mathbf{A}}(S, 2\alpha_i)) \subseteq H$ the bound that is generated by the **SB-ESQ** is at least as good as the bound that can be calculated by Equation (2) if the reliability δ is replaced by $\delta/2$. Thus by losing a factor of 2 in the worst-case reliability we gain the potential to find much better bounds.

When are these new bounds likely to be better? We know that they cannot be better in general if the task is to learn any concept from a given finite concept class C when the distribution of the inputs is arbitrary. However, it is instructive to consider the performance of **SB-ESQ** on a generic learning algorithm **Gen** which finds a hypothesis $h \in H$ by minimizing the training error. More precisely, assume that **Gen**, given the training sample S, first calculates the training error of each hypothesis in $h \in H$ and then outputs a hypothesis which has the smallest training error. Consider first the *query span* $\mathrm{Q}(\mathcal{T}_{\mathbf{Gen}}(\mathcal{D}, \eta))$ of **Gen**. As **Gen** always calculates the errors of *all* of the concepts in H the size of its query span is $|H|$ independent of the distribution $\mathcal{D}$ and of η. Next consider the hypothesis span of **Gen**. Unlike the query span, the hypothesis span depends on the distribution $\mathcal{D}$. Let C_η be the set of concepts whose expected error is within η of the minimal expected error over all the hypothesis in H. In this case it is easy to see that $\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta)) = C_\eta$, which indicates that a tighter bound than the one given in Equation (2) might be possible. Assume that the sample size m is sufficiently large that α_1, as calculated by **SB-ESQ** is smaller than $\eta/2$. In this case the calculated hypothesis span $\mathrm{H}(\widehat{\mathcal{T}}_{\mathbf{Gen}}(S, 2\alpha_1))$ will be a subset of C_η and thus the bound that **SB-ESQ** outputs is:

$$\widehat{\mathrm{err}}(h^*) + \sqrt{\frac{1}{2m} \ln \frac{2|C_\eta|}{\delta}}\ .$$

If $2|C_\eta| < |C|$ then this bound is better than the one that is calculated a-priori using Equation 2. We believe that in many real-world cases $|C_\eta| \ll |C|$ for reasonably large values of η. The analysis here is closely related to the distribution-specific analysis of Haussler et. al [7]. It extends their work in that it considers the bound that can be calculated when the distribution is unknown and only a sample from it is available.

Clearly, in learning algorithms that are more sophisticated than **Gen** the query-span will depend on the distribution $\mathcal{D}$ and, as a result, improved bounds might be possible. The case for C4.5 was sketched earlier in this section. In this case it seems that the exact description of the distributions on which the bounds will be improved is as complex as the learning algorithm itself and thus is of little use. On the other hand, we believe that algorithms such as C4.5 that use the answers they get for their initial statistical queries in order to choose which queries to make later on are more likely to enjoy improved bounds than algorithms which always make the same sequence of queries, such as **Gen**.

Even when the bounds generated by **SB-ESQ** are much better than the a-priori bounds, it is clear that their calculation, while doable in principle, is in most cases computationally infeasible. This is because in most cases many nodes in the query tree will have more than one child and thus the size of the query tree would be exponential in the number of queries made by the **ESQ** learning algorithm. This implies that *each iteration* of the loop in command 4 of **SB-ESQ** takes exponentially more time than running the learning algorithm itself!

It is thus desirable to find cases in which the computation of query span and hypothesis span can be done in time that is comparable to the running time of the algorithm. One such case is described in the next section.

3 ERROR BOUNDS FOR LOCAL SEARCH

In this section we describe a self bounding learning algorithm which is constructed specifically for algorithms based on local search, this specialized algorithm is much more efficient than **SB-ESQ** but the bounds it yields are weaker. A local search learning algorithm is a learning algorithm which searches for the best hypothesis by following a neighborhood structure that connects similar hypotheses. In this section we consider two types of local search algorithms, steepest descent and beam search. A steepest descent learning algorithm starts with some hypothesis h_0 and, on each iteration, makes statistical queries to estimate the errors of some "neighboring" hypotheses. It then replaces its hypothesis with the neighboring hypothesis that has the smallest estimated error and moves on to explore the neighbors of the new hypothesis. The popular "BackProp" learning algorithm for neural networks (see e.g. [14]), when run in batch mode, is a special case of this algorithm. In this case the local neighborhood is a small ball around the hypothesis in parameter space (weight space). Here we assume that the space of hypotheses is discretized. For example, if the model is described by real valued parameters (such as the weights in a neural network) then we can discretize it by restricting the accuracy of the parameters.

A beam search algorithm is local search algorithm which is a generalization of steepest descent. A beam search algorithm maintains a *set* of prospective hypotheses and explores the neighborhoods of all of these hypotheses in parallel.

Formally, we define the beam search learning algorithm **Beam** as follows. Let G be an undirected graph where each node corresponds to a hypothesis denoted by h_i. An edge between h_1 and h_2 embodies the prior assumption that $|\mathrm{err}(h_1) - \mathrm{err}(h_2)|$ is

likely to be small. The graph G has a designated initial node h_0. We denote the set of nodes which are connected to h_0 by a path of length at most t by G_t. **Beam** operates on a set of nodes which we call the "pool". We denote the pool on the tth iteration by $\widehat{P}_t$. The initial pool is $\widehat{P}_0 = \{h_0\}$. The minimal estimated error of the hypotheses in $\widehat{P}_t$ is denoted $\widehat{\text{minerr}}(\widehat{P}_t)$. The subset of $\widehat{P}_t$ whose estimated errors are within some small constant $\Delta > 0$ from $\widehat{\text{minerr}}(\widehat{P}_t)$ are placed in the "live" pool $\widehat{P}_t^\diamond$. The algorithm proceeds by iteratively expanding the live pool and identifying the best hypothesis in the expanded pool. After T such iterations the algorithm outputs a hypothesis with the smallest estimated error in $\widehat{P}_T$, denoted h^*. A detailed description of **Beam** is given in Figure 3. Note that **Beam** is an **ESQ** algorithm. The queries that are made by this algorithm are all requests to estimate the error of a specific hypothesis.

The only knowledge about the identity of the final hypothesis that we have before receiving the training set is that h will be a node in G_T. The best a-priori upper bound on the number of queries in $\mathcal{Q}$ is also $|G_T|$. The a-priori bound that we get using this prior knowledge and Equation (2) is

$$\mathbf{P}_{S\sim D^m}\left[\text{err}(h^*) - \widehat{\text{err}}(h^*) > \epsilon\right] \leq |G_T| e^{-2m\epsilon^2} \ . \tag{10}$$

The parameter Δ of the beam search algorithm is the "search tolerance" of the algorithm. The case $\Delta = 0$ corresponds to steepest descent. Setting $\Delta > 0$ can sometimes help avoid local minima in the path of steepest descent. The cost of increasing Δ is an increase in the size of the search pool and thus the time and space complexity of the algorithm. We shall now show that increasing Δ can serve an additional purpose - calculating an improved upper bound on the generalization error $\text{err}(h^*)$.

We now describe a transformation of **Beam** into a self-bounding algorithm which we call the inner-outer algorithm and denote **InOut**. The bounds on the generalization error that **InOut** yields are inferior to the bounds that we can get by applying **SB-ESQ** to **Beam**. On the other hand, **InOut** is much more efficient.

Let η_* be the value defined in Lemma 6 for the algorithm **Beam** and the distribution $\mathcal{D}$. Consider the sets of nodes that can be reached in the graph when all of the error estimates are within η_* of their correct value. More precisely, let $\mathcal{P}_t$ be the set of pools that correspond to a state of **Beam** after t iterations which is in $\mathcal{T}_{\textbf{Beam}}(\mathcal{D}, \eta_*)$. We denote by I_t (the inner set) the intersection of all pools in $\mathcal{P}_t$ and by O_t (the outer set) the union of all of these pools.

Intuitively, O_t is the subset of G that is reached by **Beam** for a significant fraction of training sets and $I_t(\alpha)$ is the subset of G that is reached by **Beam** for most training sets. A more precise statement is given in the following lemma:

Lemma 8 *Assume that a training sample of size m is used to estimate the errors of hypotheses of the beam search algorithm. Then with probability at least $1-|O_T(\eta_*)|e^{-2m\eta_*^2}$*

$$I_t \subseteq \widehat{P}_t \subseteq O_t \quad \forall 1 \leq t \leq T$$

Proof: It follows directly from the definitions that for all $2 \leq t \leq T$, $I_t \subseteq O_t$, $I_{t-1} \subseteq I_t$, and $O_{t-1} \subseteq O_t$. Thus all the sets $I_1, \ldots, I_T$ and $O_1, \ldots, O_{T-1}$ are subsets of O_T. This

Input: a binary labeled training set: $\{(x_1, y_1), \ldots, (x_m, y_m)\}$
A search graph G with an initial node h_0
A search tolerance $\Delta \geq 0$
Number of iterations T

1. Initialize the first pool $\widehat{P}_0 = \{h_0\}$, $\widehat{P}_0^{\diamond} = \widehat{P}_0$.
2. Repeat for $t = 1, 2, \ldots, T$
   (a) Let $\widehat{P}_t$ be the union of $\widehat{P}_{t-1}$ and the set of neighbors of the nodes in $\widehat{P}_{t-1}^{\diamond}$.
   (b) Estimate the error of the nodes in $\widehat{P}_t \backslash \widehat{P}_{t-1}$ using the training set.
   (c) Calculate the best estimated error: $\widehat{\texttt{minerr}}(\widehat{P}_t) = \min_{h \in \widehat{P}_t} \widehat{\texttt{err}}(h)$.
   (d) Let $\widehat{P}_t^{\diamond}$ include all $h \in \widehat{P}_t$ such that $\widehat{\texttt{err}}(h) \leq \widehat{\texttt{minerr}}(\widehat{P}_t) + \Delta$.
3. Output a hypothesis $h^* \in \widehat{P}_T$ for which $\widehat{\texttt{err}}(h^*) = \widehat{\texttt{minerr}}(\widehat{P}_T)$.

Figure 3: the beam search learning algorithm **Beam**.

implies that $O_T = \mathrm{Q}(\mathcal{T}_{\mathbf{Beam}}(\mathcal{D}, \eta_*))$. Applying Hoeffding's bound we get that with the probability stated in the lemma all the estimates $\widehat{\mathrm{err}}(h)$ that **Beam** receives are within η_* from the expected value $\mathrm{err}(h)$. The statement of the lemma follows from this and from the definitions of I_t and O_t. ■

The self bounding beam search algorithm **InOut** computes a superset of O_T. To do this it iteratively updates two pools, similar to the one pool used in **Beam**. The inner pool, $\widehat{I}_{i,t}$ and the outer pool $\widehat{O}_{i,t}$ which are approximations of the sets I_t and O_t respectively. The algorithm is described in Figure 4. Similarly to **SB-ESQ**, **InOut** receives as input an upper bound N_0 on the size of G_T and iteratively computes improved bounds until convergence is reached. Also, similarly to **SB-ESQ**, the loop of command 3 in **InOut**can be stopped before convergence and the only effect of this is an increase in the generated error bound.

Note that the number of pools generated by a direct application of **SB-ESQ** to **Beam** is exponential in the iteration number t. On the other hand, **InOut** maintains only two pools. This is why we claim that **InOut** is more efficient. The cost of this efficiency is that the outer pool is a rougher approximation of the query span of **Beam** than the approximation one gets from applying **SB-ESQ** to **Beam**.

Theorem 9 *The algorithm* **InOut** *is sound.*

The proof of soundness for the inner-outer algorithm is similar to the proof of Theorem 5 and consists of two parts. First we show that if all of the error estimates for $h \in O_T$ are within η_* of their correct value, then the pools that are generated by **InOut** have some desirable properties. In the second part we show that when the desirable properties hold, then, with high probability, the error bound that the algorithm outputs for its final hypothesis is correct.

The first part of the proof is contained in the following lemma. We denote by Imin_t and Omin_t the minimal (true) error among the elements of I_t and O_t respectively. With a slight abuse of notation we use $\widehat{P}_t^{\diamond} \in \mathcal{P}_t$ to denote the fact that $\widehat{P}_t^{\diamond}$ is the live subset of some reachable pool $\widehat{P}_t \in \mathcal{P}_t$. We use this notation to define inner and outer *live* sets as follows:

$$I^{\diamond}{}_t \doteq \bigcap_{\mathcal{P}_t} \widehat{P}_t^{\diamond}, \qquad O^{\diamond}{}_t \doteq \bigcup_{\mathcal{P}_t} \widehat{P}_t^{\diamond} .$$

We first prove a claim that holds for each iteration of the internal loop of command 3(c).

Lemma 10 *If $\alpha_i \geq \eta_*$ and $|err(h) - \widehat{err}(h)| \leq \eta_*$ for all $h \in O_T = Q(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))$. Then the following claims hold for all $t \in \{1, \ldots, T\}$*

1. $\widehat{I}_{i,t} \subseteq I_t$
2. $O_t \subseteq \widehat{O}_{i,t}$
3. $$Omin_t \geq \widehat{minerr}(\widehat{O}_{i,t}) - \eta_*$$

Input:
$S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$: A training set.
G: The search graph.
$\Delta \geq 0$: A search tolerance.
T : Number of iterations
δ : A reliability parameter
N_0 An upper bound on the number of nodes in G_T

1. Run **Beam** with inputs S, G, Δ and T to generate h^*
2. Compute $\widehat{\mathtt{err}}(h^*)$.
3. Repeat for $i = 1, 2, \ldots$ until $\alpha_i = \alpha_{i-1}$.
   (a) Set
   $$\alpha_i = \sqrt{\frac{1}{2m} \ln \frac{4|N_{i-1}|}{\delta}}$$
   (b) Initialize $\widehat{I}_{i,0}$ and $\widehat{O}_{i,0}$ to $\{h_0\}$.
   (c) Repeat for $t = 1, 2, \ldots, T$
      i. Let $\widehat{I}_{i,t}$ be the union of $\widehat{I}_{i,t-1}$ and the set of neighbors of all the nodes in $\widehat{I}^{\diamond}_{i,t-1}$. Similarly compute $\widehat{O}_{i,t}$ from $\widehat{O}_{i,t-1}, \widehat{O}^{\diamond}_{i,t-1}$.
      ii. Estimate the error of the (new) nodes in $\widehat{I}_{i,t}$ and $\widehat{O}_{i,t}$ from S.
      iii.
      $$\widehat{\mathtt{minerr}}(\widehat{I}_{i,t}) = \min_{h \in \widehat{I}_{i,t}} \widehat{\mathtt{err}}(h) \;\; ; \;\; \widehat{\mathtt{minerr}}(\widehat{O}_{i,t}) = \min_{h \in \widehat{O}_{i,t}} \widehat{\mathtt{err}}(h) \;\; .$$
      iv. $\widehat{I}^{\diamond}_{i,t} = \left\{ h \in \widehat{I}_{i,t} \;\; : \widehat{\mathtt{err}}(h) \leq \widehat{\mathtt{minerr}}(\widehat{O}_{i,t}) + \Delta - 2\alpha_i \right\}.$

      $$\widehat{O}^{\diamond}_{i,t} = \left\{ h \in \widehat{O}_{i,t} \;\; : \widehat{\mathtt{err}}(h) \leq \widehat{\mathtt{minerr}}(\widehat{I}_{i,t}) + \Delta + 2\alpha_i \right\}.$$

   (d) Set $N_i = |\widehat{O}_{i,T}|$
4. let $\widehat{HF} = \left\{ h \in \widehat{O}_{i,T} \;\; : \widehat{\mathtt{err}}(h) \leq \widehat{\mathtt{minerr}}(\widehat{I}_{i,T}) + 4\alpha_i \right\}.$
5. Output the error bound: $\widehat{\mathtt{err}}(h^*) + \sqrt{\frac{1}{2m} \ln \frac{2|\widehat{HF}|}{\delta}}$

Figure 4: The self bounding beam search algorithm **InOut**

and

$$\mathit{Imin}_t \leq \widehat{\mathit{minerr}}(\widehat{I}_{i,t}) + \eta_*$$

4.

$$\widehat{I}^{\diamond}_{i,t} \subseteq {I^{\diamond}}_t \text{ and } \widehat{O}^{\diamond}_{i,t} \supseteq {O^{\diamond}}_t$$

Proof: For a set of nodes A we denote by $[A]$ the union of A with the immediate neighbors of the nodes in A. We use $\bigcup_{\mathcal{P}_t} \widehat{P}_t$ and $\bigcap_{\mathcal{P}_t} \widehat{P}_t$ to indicate the union and intersection of all pools on iteration t which can be generated are all within the allowed tolerance of η_*.

The proof consists of an induction over t within an induction over i.

We fix any $i \geq 1$ and prove the lemma by induction over t. The base case, $t = 1$ is trivial. We prove the claims for $t + 1$ one by one.

1. Claim 1 for $t + 1$ follows from claims 1 and 4 for t as follows:

$$\begin{aligned}
\widehat{I}_{i,t+1} &= \widehat{I}_{i,t} \cup [\widehat{I}^{\diamond}_{i,t}] \subseteq I_t \cup [{I^{\diamond}}_t] \\
&= \bigcap_{\mathcal{P}_t} \widehat{P}_t \cup \left[\bigcap_{\mathcal{P}_t} \widehat{P}^{\diamond}_t\right] \subseteq \bigcap_{\mathcal{P}_t} \widehat{P}_t \cup \bigcap_{\mathcal{P}_t} [\widehat{P}^{\diamond}_t] \\
&\subseteq \bigcap_{\mathcal{P}_t} \left(\widehat{P}_t \cup [\widehat{P}^{\diamond}_t]\right) = \bigcap_{\mathcal{P}_t} \widehat{P}_{t+1} = I_{t+1}
\end{aligned}$$

2. Claim 2 for $t + 1$ follows from claims 2 and 4 for t as follows:

$$\begin{aligned}
\widehat{O}_{i,t+1} &= \widehat{O}_{i,t} \cup [\widehat{O}^{\diamond}_{i,t}] \supseteq O_{i,t} \cup [{O^{\diamond}}_{i,t}] \\
&= \bigcup_{\mathcal{P}_t} \widehat{P}_t \cup \left[\bigcup_{\mathcal{P}_t} \widehat{P}^{\diamond}_t\right] \supseteq \bigcup_{\mathcal{P}_t} \widehat{P}_t \cup \bigcup_{\mathcal{P}_t} [\widehat{P}^{\diamond}_t] \\
&= \bigcup_{\mathcal{P}_t} \left(\widehat{P}_t \cup [\widehat{P}^{\diamond}_t]\right) = \bigcup_{\mathcal{P}_t} \widehat{P}_{t+1} = O_{i,t+1}
\end{aligned}$$

3. Claim 3 for $t + 1$ follows from claims 1 and 2 for $t + 1$ as follows. As $\widehat{I}_{i,t+1} \subseteq I_{t+1}$ the node h^* which minimizes $\widehat{\mathrm{err}}(h)$ in $\widehat{I}_{i,t+1}$ is also an element in I_{t+1}, from the assumptions $\mathrm{err}(h^*) \leq \widehat{\mathrm{err}}(h^*) + \eta_*$. Combining these two observations we get $\mathrm{Imin}_{t+1} \leq \widehat{\mathrm{minerr}}(\widehat{I}_{i,t}) + \eta_*$. The other inequality is proved in a similar way.

4. Claim 4 for $t + 1$ follows from claims 1,2 and 3 for $t + 1$ as follows. Fix $\widehat{O}_{i,t+1}$ and $\widehat{I}_{i,t+1}$ and let $\widehat{P}_{t+1}$ be any pool in $\mathcal{P}_{t+1}$. As ${I^{\circ}}_{i,t} \doteq \bigcap_{\mathcal{P}_t} \widehat{P}^{\diamond}_t$, we have to show that $\widehat{I}^{\diamond}_{i,t+1}$, as defined in statement 3(c)[iv], is a subset of $\widehat{P}^{\diamond}_{t+1}$ for all $\widehat{P}^{\diamond}_{t+1} \in \mathcal{P}_{t+1}$.

From the definition of O_{t+1} we know that $\widehat{P}_{t+1} \subseteq O_{t+1}$ and thus $\widehat{\text{minerr}}(\widehat{P}_{t+1}) \geq \text{Omin}_{t+1} - \eta_*$.
From claim 3 we know that $\text{Omin}_{t+1} \geq \widehat{\text{minerr}}(\widehat{O}_{i,t+1}) - \eta_*$.
Combining the inequalities we get that $\widehat{\text{minerr}}(\widehat{P}_{t+1}) + \Delta \geq \widehat{\text{minerr}}(\widehat{O}_{i,t+1}) + \Delta - 2\eta_*$. From the assumption that $\alpha_i \geq \eta_*$ we get that the threshold used for calculating $\widehat{I}^{\diamond}_{i,t+1}$ is at most the threshold used for defining $\widehat{P}^{\diamond}_{t+1}$ for *all* $\widehat{P}_{t+1} \in \mathcal{P}_{t+1}$. On the other hand, from claim 1 $\widehat{I}_{i,t+1} \subseteq I_{t+1}$ and combining this with the definition of I_{t+1} we get that $\widehat{I}_{i,t+1} \subseteq \widehat{P}_{t+1}$ for *all* $\widehat{P}_{t+1} \in \mathcal{P}_{t+1}$. Finally, combining these two arguments we get that $\widehat{I}^{\diamond}_{i,t+1} \subseteq \widehat{P}^{\diamond}_{t+1}$ for all $\widehat{P}^{\diamond}_{t+1} \in \mathcal{P}$ and thus $\widehat{I}^{\diamond}_{i,t+1} \subseteq {I^{\diamond}}_{t+1}$. The other inclusion is proved in a similar way.

■

Given Lemma 10, we can now prove, under the same assumption, a lower bound on α_i:

Lemma 11 *If $|err(h) - \widehat{err}(h)| \leq \eta_*$ for all $h \in O_T = Q(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))$, then $\alpha_i \geq \eta_*$ for all $i \geq 1$.*

Proof: We prove the claim by induction over i. Recall that

$$\eta_* = \sqrt{\frac{1}{2m} \ln \frac{4|\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))|}{\delta}} \,,$$

and that $\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*)) = O_T$. Recall also that

$$\alpha_i = \sqrt{\frac{1}{2m} \ln \frac{4|N_{i-1}|}{\delta}} \,,$$

Thus if $N_{i-1} \geq |\mathrm{Q}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \eta_*))|$ then $\alpha_i \geq \eta_*$ as required. As $N_0 \geq |G_T|$ the bound holds for α_1.

Assuming that $\alpha_i \geq \eta_*$ we get from claim 2 of Lemma 10 for $t = T$ that $O_T \subseteq \widehat{O}_{i,T}$. As $N_i = |\widehat{O}_{i,T}|$ we get that $N_i \geq O_T$ and thus $\alpha_{i+1} \geq \eta_*$. ■

Combining the claims of Lemmas 10 and 11 we get that, under the same assumptions, the three claims of Lemmas 10 hold for all i. Using these properties we can prove that **InOut** is sound.

proof of Theorem 9: There are two modes of failure and we bound the probability of each one of them by $\delta/2$.

As in the proof of Theorem 5, we have that with probability at least $1 - \delta/2$, $|\mathrm{err}(h) - \widehat{\mathrm{err}}(h)| \leq \eta_*$ for all $h \in O_T$.

We now assume that $|\mathrm{err}(h) - \widehat{\mathrm{err}}(h)| \leq \eta_*$ for all $h \in O_T$. From the definitions of O_T and Imin_T we get that under this assumption

$$\widehat{P}_T \subseteq O_T \tag{11}$$

and

$$\widehat{\text{minerr}}(\widehat{P}_T) \leq \text{Imin}_T + \eta_* \ . \tag{12}$$

As $\widehat{\text{err}}(h^*) = \widehat{\text{minerr}}(\widehat{P}_T)$ and $|\text{err}(h^*) - \widehat{\text{err}}(h^*)| \leq \eta_*$, we find that h^* is in the set

$$A := \{h \in O_T \ : \ \text{err}(h) \leq \text{Imin}_T + 2\eta_*\}$$

Using Hoeffding bound we get that the probability that $|\text{err}(h^*) - \widehat{\text{err}}(h^*)| > \epsilon$ is at most

$$2|A|e^{-2m\epsilon^2} \tag{13}$$

We now show that, under the same assumption A is a subset of the set $\widehat{HF}$ which is calculated on line 5 of Figure 4. From Lemma 10 we know that

$$O_T \subseteq \widehat{O}_{i,T} \tag{14}$$

and that

$$\text{Imin}_T \leq \widehat{\text{minerr}}(\widehat{I}_{i,T}) + \eta_* \ . \tag{15}$$

Assume that $h \in A$, then $h \in O_T$ which implies that $h \in \widehat{O}_{i,T}$. On the other hand $h \in A$ implies that $\text{err}(h) \leq \text{Imin}_T + 2\eta_*$ which implies that $\widehat{\text{err}}(h) \leq \widehat{\text{minerr}}(\widehat{I}_{i,T}) + 4\eta_*$. Combining the two implied conditions we get the definition of $\widehat{HF}$ and thus $A \subseteq \widehat{HF}$.

Finally, if we plug the choice of ϵ made by the algorithm into Equation 13, we get that the probability that $\text{err}(h^*) \geq \widehat{\text{err}}(h^*) + \sqrt{\frac{1}{2m} \ln \frac{2|\widehat{HF}|}{\delta}}$ is at most

$$|A|e^{-2m\epsilon^2} = |A| \exp\left(-2m \frac{1}{2m} \ln \frac{2|\widehat{HF}|}{\delta}\right) = \frac{|A|}{|\widehat{HF}|} \frac{\delta}{2} \leq \frac{\delta}{2}$$

which completes the proof. ■

The same argument that we made about **SB-ESQ** can be used here to show that the upper bound generated by **InOut** with reliability $1 - \delta$ are at least as good as the a-priori bounds that are generated using Equation 2 with reliability $1 - \delta/2$.

Once again, the interesting question is *when* would these new a-posteriori bound be significantly *better*? Clearly, a necessary condition is that O_T is much smaller than G_T. However, it is not hard to construct cases in which O_T is small but $\widehat{O}_T$ is, with high probability, equal to G_T. In this case there exists a very good bound on the error but the bound given by the self bounding algorithm is not better than the a-priori uniform bound.

On the other hand, note that the the updates of the inner and outer pools $\widehat{I}_t$ and $\widehat{O}_t$ differ from the updates to $\widehat{P}_t$ only in the way that the live pools are calculated in statement 3(d). Thus if α is sufficiently small, the inner and outer pools remain similar to $\widehat{P}_t$ for many iterations, in which case $\widehat{O}_t$ is a good approximation of O_t and the bounds that the algorithm yields are almost as good as the bounds that can be achieved knowing O_t.

We cannot, at this time, give a formal characterization of the cases in which the inner-outer algorithm is guaranteed to yield good bounds. However, to justify our belief that such cases are likely to occur in practice , we sketch a simple example.

Let us assume that the search graph G is a grid in R^d in which neighbors are nodes that differ from each other by at most $\epsilon > 0$ in each coordinate (this can be used as an approximation of gradient descent search on hypothesis described by d real valued parameters, such as a neural network with d weights). Consider the *true* expected errors of the nodes in the search graph. Assume that there is only a single path of steepest descent and that each node h along this path has the following two properties. (1) there is only one neighbor of h which has a smaller error that that of h and (2) the errors of the other neighbors of h are all larger than the error of h by at least some constant $a > 0$. Clearly, if we have access to the true errors of the nodes then a steepest descent algorithm could follow the path, as we are only given estimated errors, we have to use a beam search algorithm with $\Delta > 0$ to avoid possible local minima in the estimated errors along the path. Suppose that the error estimates are all within $a/8$ from their true value, then it is easy to verify that running the beam search algorithm with $\Delta = a/4$ will proceed along the path of steepest descent and after T iterations, reach some node h^* along that path. Consider now what bounds we can give on the true error of h^*. An a-priori bound will depend on the size of the reachable graph G_T. As the graph is a grid in R^d then $|G_T| = O(T^d)$ and the bound that we get on $\mathrm{err}(h^*) - \widehat{\mathrm{err}}(h^*)$ is

$$\alpha = \sqrt{\frac{1}{2m} \ln \frac{|G_T|}{\delta}} = O\left(\frac{d}{m} \ln \frac{T}{\delta}\right)) \ . \tag{16}$$

On the other hand, if

$$\sqrt{\frac{1}{2m} \ln \frac{2|G_T|}{\delta}} \leq a/4 \ ,$$

then running **InOut** would result in an outer pool $\widehat{O}_{1,T}$ that includes only nodes along the path of steepest descent and thus $|O_{1,T}| \leq T$ and the bound the algorithm outputs is

$$\sqrt{\frac{1}{2m} \ln \frac{2T}{\delta}} \tag{17}$$

i.e. an improvement by a factor of $\sqrt{d}$ over the bound given in Equation (16). While it is clear that this examples can be generalized in various ways, it is unclear whether there exists a simple general characterization of the situations in which **InOut** generates superior upper bounds.

4 CONCLUSIONS AND SUGGESTED FUTURE DIRECTIONS

This work represents an initial attempt to define and analyze self-bounding learning algorithms. We believe that self bounding algorithms can give superior bounds on the

generalization error of practical algorithms. As the bounds depend on the distribution, it is hard to give a simple characterization of the situations in which the bounds achieved will be better than the a-priori ones. Also, it is not clear whether there is a notion of an "optimal" self bounding algorithm. Experiments with self bounding algorithms are needed in order to determine whether its potential for yielding improved bounds is realized in practice.

The analysis given in this paper is restricted to deterministic **ESQ** learning algorithms which use a finite class of queries and output a binary hypothesis from a finite class. It is desirable to extend the analysis to more general learning algorithms and to more general learning problems. A related issue is to use more refined measures of complexity for the query span and for the hypothesis span. It might be, for instance, that the algorithm might generate many different hypotheses, but that all of these hypotheses make very similar predictions on most of the training set. In this case we would like to use a better characterization of the complexity of the hypothesis span than its size.

A different issue is how to make the self bounding algorithm more efficient. One method that seems attractive is to use sampling to estimate the size of $\mathrm{H}(\mathcal{T}_{\mathbf{A}}(\mathcal{D}, \alpha_i))$. It is desirable to find conditions under which sampling would give good estimates. Ideally, one would like to be able to test these conditions using only the training set.

Estimates of the generalization error of a hypothesis are often used *within* learning algorithms. Principled approaches such as MDL and SRM yield bounds that are used in order to select the best balance between training error and complexity. The self bounding analysis suggests a different principled approach for selecting the right complexity. The advantage of this approach is that it allows the algorithm to choose a very complex hypotheses when such a choice is justified for the specific distribution being learned, even when a-priori bounds, which consider the worst-case distribution, would suggest avoiding such high complexity.

The learning algorithm might use approximations of the query tree in order to reduce its generalization error. For example, it seems likely that taking the majority vote over the hypotheses span of an algorithm can reduce the generalization error because it decreases the differences between the hypotheses in the span. This might provide a new way for analyzing ensemble methods such as Bagging C4.5 [4] and Randomized C4.5 [6] which can be seen as taking the majority vote over samples from the hypothesis span of C4.5.

ACKNOWLEDGEMENTS

Special thanks to David McAllester for inspiring discussions that led to this work. I thank Fernando Pereira and Rob Schapire for helpful comments and suggestions.

References

[1] Javed A. Aslam and Scott E. Decatur. On the sample complexity of noise-tolerant learning. *Information Processing Letters*, 57(4):189–195, 26 February 1996.

[2] Gyora M. Benedek and Alon Itai. Learnability with respect to fixed distributions. *Theoretical Computer Science*, 86(2):377–389, September 1991.

[3] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, April 1987.

[4] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[6] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Unpublished manuscript, 1998.

[7] David Haussler, Michael Kearns, H. Sebastian Seung, and Naftali Tishby. Rigorous learning curve bounds from statistical mechanics. *Machine Learning*, 25:195–236, 1996.

[8] D. Heckerman, D. Geiger, and D.M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[9] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[10] 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, 1993.

[11] Yishay Mansour. Pessimistic decision tree pruning based on tree size. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 195–201, 1997.

[12] David A. McAllester. Some pac-bayesian theorems. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998.

[13] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[14] David E. Rumelhart and James L. McClelland, editors. *Parallel Distributed Processing*. MIT Press, 1986.

[15] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. A framework for structural risk minimisation. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 68–76, 1996.

[16] John Shawe-Taylor and Robert C. Williamson. A pac analysis of a bayesian estimator. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pages 2–9, 1997.