Some Weak Learning Results

David P. Helmbold
Department of Computer Science
University of California at Santa Cruz
email: dph@cis.ucsc.edu

Manfred K. Warmuth
Department of Computer Science
University of California at Santa Cruz
email: manfred@cis.ucsc.edu

Abstract

An algorithm is a weak learner if with some small probability it outputs a hypothesis with error slightly below 50%. This paper presents sufficient conditions for weak learning.

Our main result requires a "consistency oracle" for the concept class $\mathcal{F}$ which decides for a given set of examples whether there is a concept in $\mathcal{F}$ consistent with the examples. We show that such an oracle can be used to construct a computationally efficient weak learning algorithm for $\mathcal{F}$ if $\mathcal{F}$ is learnable at all. We consider consistency oracles which are allowed to give wrong answers and discusses how the number of incorrect answers effects the oracle's use in computationally efficient weak learning algorithms.

We also define "weak Occam algorithms" which, when given a set of m examples, select a consistent hypothesis from some class of $2^{m-(1/p(m))}$ possible hypotheses. We show that these weak Occam algorithms are also weak learners. In contrast, we show that an Occam style algorithm which selects a consistent hypotheses from a class of $2^{m+1}-2$ hypotheses is not a weak learner.

1 Introduction

To help introduce many of the notions used in this paper we use the problem of learning DFAs over a binary alphabet as an example learning problem. A learning algorithm is given random bitstrings (called instances) which are labeled with 0 or 1 depending on whether they are rejected or accepted by some hidden DFA. The labeled bitstrings are called examples and each possible hidden DFA defines a concept consisting of all bitstrings which it accepts (or equivalently, the associated indicator function). The set of possible concepts is called the concept class. Assume the algorithm is given two parameters: n, a length bound on the bitstrings, and s, a size bound (number of states) for the hidden DFA. After seeing a reasonable number of instances labeled with the hidden target concept, the learning algorithm outputs a hypothesis which is intended to approximate the set of bitstrings of length at most n which are accepted by the hidden target DFA. We assume that the instances are generated according to a fixed but arbitrary probability distribution and define the error of the output hypothesis as the probability of the symmetric difference between the hypothesis and the target.

A *strong* learning algorithm takes parameters $\epsilon > 0$ and $\delta > 0$ and must, with probability at least $1-\delta$, output a hypothesis having error at most ϵ. To generate this hypothesis, the algorithm is allowed to examine a number of examples equal to some polynomial $p(n, s, 1/\epsilon, 1/\delta)$. Learning algorithms are called computationally efficient if both their running time and the running time for evaluating the output hypotheses on instances of length at most n is bounded by a polynomial in all four parameters.

This notion of learning was introduced by Valiant [Val84]. Not too many concept classes have been shown to be efficiently strongly learnable and a less stringent definition of learning was given in [KV89]. A *weak* learning algorithm must, after seeing $m = p_1(n, s)$ many examples, output a hypothesis that has error at most $\frac{1}{2} - 1/p_2(m)$ with probability at least[1] $1/p_3(m)$, where p_1, p_2 and p_3 are polynomials.

Surprisingly, it has been shown that any computationally efficient weak learning algorithm can be used to build a computationally efficient strong learning algorithm [Sch90, Fre90]. Thus to determine whether a concept class is efficiently learnable it suffices to construct efficient weak learning algorithms. In this paper we give sufficient conditions for weak learning.

COLT'92-7/92/PA,USA

[1] In the original definition of weak learning the paramenter $1-\delta$ is used is place of $1/p_3(m)$. The two definitions are polynomially equivalent (see Lemma 3.4 of [HKLW91]).

A standard way to construct a learning algorithm for some concept class $\mathcal{F}$ is to find an "Occam algorithm" which, when given any sequence of m examples labeled by some concept in the class, outputs a consistent hypothesis from a "small" hypothesis class (which may be different from the concept class) [BEHW87, BEHW89]. If the size of the hypothesis class used by an Occam algorithm is bounded by $p(n,s)m^{\alpha}$, where α is a constant strictly less than 1, then there is a polynomial sample size bound that makes this algorithm a strong learning algorithm for $\mathcal{F}$ [BEHW87].

Here we define "weak" Occam algorithms that given a sample of size $m = p_1(n,s)$ output a consistent hypothesis from a set of at most $2^{m-(1/p_2(m))}$ hypotheses, where p_1 and p_2 are any polynomials,[2] and show that they are weak learning algorithms. If a weak Occam algorithm is computationally efficient and its possible hypotheses can be efficiently evaluated then it is a computationally efficient weak learning algorithm. Weak Occam algorithms can be viewed as compressing a sample of size m (containing m labeling bits) into $m - (1/p_2(m))$ "bits" identifying a hypothesis. In contrast, we present an Occam algorithm whose hypotheses class has size $2^{m+1}-2$ (indexed by almost $m+1$ bits) that always outputs hypotheses of error $\frac{1}{2}$ for some target concept in $\mathcal{F}$. This means that weak Occam algorithms which compress m examples to $m+1$ bits do not always lead to weak learning algorithms.

A *weak consistency oracle* for a concept class $\mathcal{F}$ is given parameters n, s, and a sequence of $m = p(n,s)$ labeled examples whose instances are from X_n. In polynomial time, the oracle determines whether or not there is a concept in $\mathcal{F}_s$ which is consistent with the examples. The consistency oracle's answer is a simple yes/no decision, making it (apparently) much weaker than an oracle which returns an $f \in \mathcal{F}_s$ consistent with the examples.

We show that if a polynomial time weak consistency oracle is available then it can be used to construct a computationally efficient weak learning algorithm for $\mathcal{F}$ when $\mathcal{F}$ is learnable at all, i.e. the Vapnik-Chervonenkis dimension [VC71] of $\mathcal{F}$ grows polynomially in n and s [BEHW89]. Previously a direct construction of a strong learning algorithm using the consistency oracle was given in [HLW]. However this algorithm is only computationally efficient if the VC dimension of $\mathcal{F}$ is a constant independent of n and s.

Our computationally efficient weak learning algorithm is the last in a series of algorithms. The series starts with a good prediction algorithm which is a special case of Vovk's Aggregating Strategy[3] [Vov90]. We modify that algorithm so that its chance of making a mistake on the last trial is reduced. We next introduce a second modification so that the algorithm now predicts based on the dichotomies (possible labelings) of the sample. The final modification incorporates random sampling into the algorithm, resulting in an efficient prediction algorithm when a weak consistency oracle is available.

Our techniques imply that an even weaker "one-sided consistency oracle" can be used to construct computationally efficient weak learning algorithms. The one-sided consistency oracle takes a sequence S of $m = p_1(n,s)$ labeled instances and returns "yes" if the labeled instances are consistent with a concept in $\mathcal{F}$. If the labeling is not consistent, then the oracle may return either "yes" or "no", however the oracle must return "no" to at least $2^{m-(1/p_2(m))}$ of the possible labelings of S. The one-sided consistency oracle for $\mathcal{F}$ can be viewed as a consistency oracle for a larger class $\mathcal{H}$ which contains $\mathcal{F}$ and has the property that for each sequence S, the number of labelings consistent with concepts in $\mathcal{H}$ is at most $2^{m-(1/p_2(m))}$.

In future research we would like to explore even less powerful consistency oracles that may still lead to efficient weak learning algorithms. For example we would like to relax the one-sidedness of the current oracle and explore probabilistic consistency oracles. In general, we see two potential benefits from this line of research. First we hope that polynomial versions of the discussed one-sided consistency oracles can be found for concept classes that have not previously been known to be learnable.

Second our results have applications to cryptography that seem promising. For example, there is no efficient weak learning algorithm for DFA's given standard cryptographic assumptions [KV89]. Our results show that, under the same cryptographic assumptions, a one-sided consistency oracle for DFAs can not exist. For any fixed set of m instances, the number of labelings corresponding to DFAs with at most s states is at most $m^{O(s \log s)}$ [Sau72], since the VC dimension of s-state DFAs is $O(s \log s)$. When m is polynomial in s this number of labelings consistent with DFAs is much smaller than the 2^m vertices in the Boolean m-cube. Yet (given the cryptographic assumptions) there is no polynomial-time algorithm that answers "yes" on the $m^{O(s \log s)}$ labelings consistent with s-state DFAs and "no" on only

$$2^m - 2^{m-1/p_2(m)} = 2^m(1 - \frac{1}{2^{1/p_2(m)}})$$

other labelings.

The following section contains an introduction to our notation and a description of the learning models considered. Section 3 contains the weak Occam results. Section 4 describes the Information Gain Prediction Algorithm which is a special case of Vovk's Aggregating Strategy [Vov90]. In Section 5, a modified prediction algorithm which has a reduced chance of error on the last trial is presented. We define a class of priors over the dichotomies of a sample and analyze the performance of an algorithm using those priors in Section 6. In Section 7 we describe an efficient version of the algorithm in Section 6 and prove our main theorem.

[2] We only consider polynomials that are nonnegative on all natural numbers.

[3] An alternate efficient weak prediction algorithm that uses fewer consistency queries than the one described in this paper is given in [HW92].

2 Notation and Models

Throughout, lg and log denote the binary and natural logarithms respectively and $\mathbf{N}$ denotes the natural numbers.

Let X be a set of *instances* called the *domain*, and $\mathcal{F}$ be a set of subsets of X called the *concept class* ($\mathcal{F} \subseteq 2^X$). We use subsets of X and their corresponding indicator functions interchangeably, so each *concept* $f \in \mathcal{F}$ maps X to $\{0,1\}$.

Lower case bold letters, such as $\boldsymbol{x}$ and $\boldsymbol{y}$ denote (finite) sequences of instances and $|\boldsymbol{x}|$ is the length of the sequence $\boldsymbol{x}$. For $1 \leq t \leq |\boldsymbol{x}|$, we use x_t to denote the tth component of $\boldsymbol{x}$, $\boldsymbol{x}^t$ to denote the t-vector $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ and $\boldsymbol{x}^0$ denotes the empty sequence Λ. We will often superscript sequences solely to emphasize their length.

Examples are elements of $X \times \{0,1\}$ and *samples* are sequences of examples. The *sample of f on $\boldsymbol{x}$* is denoted by $\mathrm{sam}_f(\boldsymbol{x})$ and is the sequence of examples[4] $(x_1, f(x_1)), \ldots, (x_{|\boldsymbol{x}|}, f(x_{|\boldsymbol{x}|}))$. We define $\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x}) = \{\mathrm{sam}_f(\boldsymbol{x}) : f \in F\}$ and call the elements of $\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x})$ the *dichotomies* of $\mathcal{F}$ induced by $\boldsymbol{x}$. We also use $\mathrm{sam}_*(\boldsymbol{x})$ to denote the set of $2^{|\boldsymbol{x}|}$ samples where the first example contains $\boldsymbol{x}_1$, and second example contains $\boldsymbol{x}_2$, and so on.

If S is a sample of $m \geq 1$ examples, then S^- denotes the sample of $m-1$ examples obtained by deleting the last example of S. If e is an example then "S, e" is the sample of $m+1$ examples obtained by adding e to the end of S. We use Λ to denote the empty sequence (of samples or examples).

We say that a function f is *consistent* with a sample S if there is an $\boldsymbol{x}$ (the sequence of instances in the sample) such that $S = \mathrm{sam}_f(\boldsymbol{x})$. Every $f \in F$ is consistent with the empty sample.

We use $\mathbf{E}_{f \in \mathcal{P}}[a(f)]$ to denote the expectation of the random variable a under distribution $\mathcal{P}$, and $\mathbf{Pr}_{f \in \mathcal{P}}[\mathrm{condition}(f)]$ to denote the probability under the distribution $\mathcal{P}$ of the set containing all f satisfying the condition. Throughout, $\mathcal{P}$ denotes the prior probability distribution on $\mathcal{F}$. For each $f \in \mathcal{F}$, $\mathcal{P}(f)$ represents the extent to which the learner initially (before seeing any examples) believes that f is the target function to be learned. Distribution $\mathcal{D}$ always denotes a probability distribution on X. We use U to denote various uniform distributions – in particular $U_{[0,1]}$ is the uniform distribution on the continuous interval $[0,1]$ and $U(\boldsymbol{x}^m)$ is the uniform distribution on the $m!$ permutations of sequence $\boldsymbol{x}$.

The *volume with respect to $\mathcal{P}$ of sample S* is written $V^{\mathcal{P}}(S)$ and denotes $\mathbf{Pr}_{f \in \mathcal{P}}[f \text{ is consistent with } S]$. Note that the empty sample has unit volume and that the volume of a sample depends only on the examples in the sample and not the order in which they appear. Furthermore, for any sample S and $x \in X$, $V^{\mathcal{P}}(S) = V^{\mathcal{P}}(S,(x,0)) + V^{\mathcal{P}}(S,(x,1))$.

For nonempty S, the *information gain* of S, $I^{\mathcal{P}}(S)$, is defined as $-\lg(V^{\mathcal{P}}(S)/V^{\mathcal{P}}(S^-))$. This quantity can be thought of as the information gained by the last example of S after seeing the other examples of S. We define $I^{\mathcal{P}}(S)$ to be infinite when $V^{\mathcal{P}}(S) = 0$ and $V^{\mathcal{P}}(S^-) > 0$. When $V^{\mathcal{P}}(S) = V^{\mathcal{P}}(S^-) = 0$ then $I^{\mathcal{P}}(S)$ is undefined.

A (randomized) *prediction algorithm*[5] A takes a sample, an instance, and a random number[6] in $[0,1]$ as input and outputs a prediction from the set $\{0,1\}$. Thus $A : (X \times \{0,1\})^* \times X \times [0,1] \to \{0,1\}$.

A (randomized) *learning algorithm* A for a concept class $\mathcal{F}$ on X receives as input a sample of some concept $f \in \mathcal{F}$ and random number $r \in [0,1]$. A outputs the representation of a concept h in a second concept class $\mathcal{H}$ on X that approximates f. $\mathcal{H}$ is called the *class of hypotheses*. Let $A(\mathrm{sam}_f(\boldsymbol{x}^{m-1}), x_m, r)$ denote (the representation of) the hypothesis output by algorithm A when run on the example sequence $\mathrm{sam}_f(\boldsymbol{x}^{m-1})$, instance x_m, and randomization r. Each learning algorithm implicitly defines a (deterministic) *evaluation algorithm* that takes as input the representation of a hypothesis and an instance $x \in X$, and outputs the value of the hypothesis on x. There are trivial learning algorithms that simply output $\langle \mathrm{sam}_f(\boldsymbol{x}), r \rangle$ as the representation of the hypothesis. In that case the evaluation algorithm does all the "work".

The performance of prediction and learning algorithms can be evaluated in several ways. For learning algorithms we are primarily interested in how well the algorithm's hypothesis approximates the function being learned. For prediction algorithms we look at both the expected number of incorrect predictions made over a sequence of trials and the probability of an incorrect prediction on the mth trial.

The error with respect to distribution $\mathcal{D}$ on X of a learning algorithm's hypothesis h when the concept to be learned is f is denoted $\mathrm{Err}_{\mathcal{D}}(f,h)$. Formally, $\mathrm{Err}_{\mathcal{D}}(f,h) = \mathbf{Pr}_{x \in \mathcal{D}}[f(x) \neq h(x)]$.

If A is a prediction algorithm and $f \in \mathcal{F}$ then $\mathbf{M}_{A,f}(\boldsymbol{x})$ is the probability that A makes a mistake on the last instance of $\boldsymbol{x}$ when learning f. More precisely, when $\boldsymbol{x}$ is a sequence of m instances, $\mathbf{M}_{A,f}(\boldsymbol{x}) = \mathbf{Pr}_{r \in U_{[0,1]}}[A(\mathrm{sam}_f(\boldsymbol{x}^{m-1}), x_m, r) \neq f(x_m)]$, where $U_{[0,1]}$ is the uniform distribution on $[0,1]$. Thus the expected total number of mistakes made by A on a

[4] If $\boldsymbol{x}$ is the empty sequence of instances, then $\mathrm{sam}_f(\boldsymbol{x})$ is the empty sequence of examples.

[5] Here, and in the definition of "learning algorithm" we use the term "algorithm" loosely, without the requirement that the mapping be computable. However, all the algorithms we present here are computable when the volumes of samples can be computed.

[6] For simplicity we let r be a real number drawn from the uniform distribution on $[0,1]$. More precisely the random input r given to an algorithm should be a finite number of random bits and for computationally efficient algorithms the number of random bits required must be polynomially bounded.

sequence of m instances, $\boldsymbol{x}^m$, is $\sum_{t=1}^{m} \mathbf{M}_{A,f}(\boldsymbol{x}^t)$.

In some sense learning algorithms and prediction algorithms are interchangeable. Given any prediction algorithm, A, one can create a trivial learning algorithm, A', which uses A as its hypothesis evaluator, i.e.

$$A'(\mathrm{sam}_f(\boldsymbol{x}^{m-1}), r)(x_m) = A(\mathrm{sam}_f(\boldsymbol{x}^{m-1}), x_m, r).$$

Furthermore, any learning algorithm and its associated hypothesis evaluator can be used to produce predictions.

Our performance measures for learning and prediction algorithms can be related as follows. Suppose prediction algorithm A when given $\mathrm{sam}_f(\boldsymbol{x}^{m-1})$, x_m, and r first uses learning algorithm A' to produce a hypothesis $h_r = A'(\mathrm{sam}_f(\boldsymbol{x}^{m-1}), r)$ and then predicts with the value $h_r(x_m)$. In this case, with $\boldsymbol{x}^{m-1}$ and f fixed, $\mathbf{E}_{x_m \in \mathcal{D}}[\mathbf{M}_{A,f}(\boldsymbol{x}^m)] = \mathbf{E}_{r \in [0,1]}[\mathrm{Err}_{\mathcal{D}}(f, h_r)]$. The same relationship holds when learning algorithm A' uses the prediction algorithm A as its hypothesis evaluator.

Usually we are not just interested in learning a fixed concept class $\mathcal{F}$ over a fixed domain X but instead we would like to learn a parameterized concept class $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \cdots$ over a parameterized domain $X = X_1 \cup X_2 \cup \cdots$. Informally, the parameter s in $\mathcal{F}_s$ measures the "size" of the concepts and $\mathcal{F}_s$ contains all concepts of size at most s. Similarly, the paramenter n in X_n measures the "length" of the instances and X_n contains all instances of length at most n. For example, X_n might consist of all bitstrings of length at most n and $\mathcal{F}_s$ of all concepts accepted by DFAs of at most s states. The prediction (or learning) algorithm is given both parameters as inputs, and the algorithm is *polynomial* if its resource requirements grow polynomially in n, s, and the size of the input sample.

Algorithm A is a *weak learning algorithm* [KV89] for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ if there exist three polynomials p_1, p_2 and p_3 s.t. if A is given the parameters n and s then for all $f \in \mathcal{F}_s$ and probability distributions $\mathcal{D}$ on X_n the following holds: upon receiving a random number $r \in [0,1]$ drawn according to $U_{[0,1]}$ plus a sample $\mathrm{sam}_f(\boldsymbol{x}^m)$, where $\boldsymbol{x}^m$ is drawn according to D^m and $m = p_1(n, s)$, the algorithm outputs a hypothesis $h = A[\mathrm{sam}_f(\boldsymbol{x}), r]$ on X_n for which

$$\mathbf{Pr}_{\boldsymbol{x} \in \mathcal{D}^m, r \in U_{[0,1]}}\left[\mathrm{Err}_{\mathcal{D}}(f,h) > \frac{1}{2} - \frac{1}{p_2(m)}\right] \leq 1 - \frac{1}{p_3(m)} \qquad (1)$$

For a *strong learning algorithm* [Val84, BEHW89] condition (1) is replaced by the condition

$$\mathbf{Pr}_{\boldsymbol{x} \in \mathcal{D}^m, r \in U_{[0,1]}}[\mathrm{Err}_{\mathcal{D}}(f,h) > \epsilon] \leq \delta,$$

where ϵ and δ are additional paramenters in $[0,1]$. These parameters are given to the algorithm and the sample size m is allowed to be polynomial in n and s as well as $1/\epsilon$ and $1/\delta$.

The hypotheses output by a learning algorithm for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ are *polynomially evaluatable* if the evaluation algorithm's running time on any hypothesis representation output by the learning algorithm and any instance $x \in X_n$ is polynomial in the parameters n and s of the learning algorithm. A *computationally efficient* weak (strong) learning algorithm must output polynomially evaluatable hypotheses and the total running time of the weak learning algorithm must be polynomial in n and s (respectively in n, s, $1/\epsilon$ and $1/\delta$ for strong learning algorithms).

It has been shown that any weak learning algorithm A for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ can be used iteratively to build a strong learning algorithm for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ [Sch90, Fre90]. Moreover if the weak learning algorithm is computationally efficient then the strong learning is also computationally efficient.

In this paper we also use the following sufficient condition for weak learning (proved using Markov's Lemma) in place of condition (1). It bounds the probability that prediction algorithm A makes a mistake in the last trial away from $\frac{1}{2}$: for $m = p_1(n, s)$

$$\mathbf{E}_{\boldsymbol{x} \in \mathcal{D}^m}[\mathbf{M}_{A,f}(\boldsymbol{x})] \leq \frac{1}{2} - \frac{1}{p_2(m)}, \qquad (2)$$

where p_1 and p_2 are two polynomials replacing the three polynomials used in the previous condition.

The sufficiency of condition (2) for weak learning follows from the following lemma applied with $\alpha = 1/p_2(m)$. The lemma says that if the expected error is $1 - \alpha$ then the error is at most is $1 - \frac{\alpha}{2}$ with probability at most $1 - \frac{\alpha}{1-\alpha}$.

Lemma 1 *For any distribution $\mathcal{D}$ on X, any learning algorithm A for $\mathcal{F}$ on X, and any concept f on X, if*

$$\mathbf{E}_{\boldsymbol{x} \in \mathcal{D}^m}[\mathbf{M}_{A,f}(\boldsymbol{x})] \leq \frac{1}{2} - \alpha,$$

then

$$\mathbf{Pr}_{D^{m-1} \times U_{[0,1]}}\left[Err_{\mathcal{D}}(f, A[\mathrm{sam}_f(\boldsymbol{x}^{m-1}), r]) \geq \frac{1}{2} - \frac{\alpha}{2}\right] \leq 1 - \frac{\alpha}{1-\alpha}.$$

Proof: Recall that when learning algorithm A' uses prediction algorithm A to evaluate its hypotheses:

$$\begin{aligned}
&\mathbf{E}_{\boldsymbol{x} \in \mathcal{D}^m}[\mathbf{M}_{A,f}(\boldsymbol{x})] \\
&= \mathbf{E}_{\boldsymbol{x} \in \mathcal{D}^m, r \in U_{[0,1]}}[A(\mathrm{sam}_f(\boldsymbol{x}^{m-1}, x_m, r) \neq f(x_m))] \\
&= \mathbf{E}_{\boldsymbol{x} \in \mathcal{D}^{m-1}, r \in U_{[0,1]}} \\
&\qquad [\mathbf{E}_{x_m \in \mathcal{D}}[A(\mathrm{sam}_f(\boldsymbol{x}^{m-1}, x_m, r) \neq f(x_m))]] \\
&= \mathbf{E}_{\boldsymbol{x} \in \mathcal{D}^{m-1}, r \in U_{[0,1]}}[\mathrm{Err}_{\mathcal{D}}(f, A'(\mathrm{sam}_f(\boldsymbol{x}^{m-1}), r))].
\end{aligned}$$

Markov's Lemma says that for any non-negative random variable R, any distribution D and $z > 0$, $\mathbf{Pr}_{a \in D}[R(a) \geq z\mathbf{E}_{b \in D}[R(b)]] \leq 1/z$. The lemma follows by using $\mathrm{Err}_{\mathcal{D}}(f, A[\mathrm{sam}_f(\cdot), \cdot])$ as the random variable mapping $X^{m-1} \times [0,1]$ to $[0,1]$, $\mathcal{D}^{m-1} \times U_{[0,1]}$ as the distribution and $z = (\frac{1}{2} - \frac{\alpha}{2})/(\frac{1}{2} - \alpha)$. □

Although inequality (2) implies inequality (1), the converse is not true. Inequality (2) is a stronger constraint on the learner than inequality (1).

3 The Consistency Trick

One way to construct a learning algorithm is to use an "Occam algorithm" which outputs consistent hypothesis from a "small" hypotheses class [BEHW87]. A *strong Occam* learning Algorithm A for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ is specified by a polynomial p and a constant $\kappa < 1$ such that the following holds for all $n, s \geq 1$, $f \in F_s$ and $\boldsymbol{x}^m \in X_n^m$:

> when given n, s, and the sample $\mathrm{sam}_f(\boldsymbol{x}^m)$, learning Algorithm A outputs a hypothesis on X_n that is consistent with the sample and is from a class $\mathcal{H}_{n,s,m}$ of cardinality at most $p(n,s)m^\kappa$.

In [BEHW87] it is shown that for each *strong Occam algorithm* for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ there is a sample size polynomial in n, s, $1/\epsilon$ and $1/\delta$ for which this algorithm is a strong learning algorithm.

Here we define "weak Occam algorithms" whose hypothesis classes grow exponentially in m and show using the methods of [BEHW87] that weak Occam algorithms lead to weak learning algorithms. Thus they can be used iteratively to build strong learning algorithms [Sch90, Fre90].

A *weak Occam Algorithm* A for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ is specified by two polynomials p_1 and p_2 such that the following holds for all $n, s \geq 1$, $f \in F_s$, $m = p_1(n,s)$, and $\boldsymbol{x}^m \in X_n^m$:

> when given n, s, and the sample $\mathrm{sam}_f(\boldsymbol{x}^m)$, Algorithm A outputs a hypothesis on X_n that is consistent with the sample and is from a class $\mathcal{H}_{n,s}$ of cardinality at most $2^{m-1/p_2(m)}$.

Lemma 2 *Let $\mathcal{D}$ be any distribution on X, $m \in N$, f be any concept on X, and $\mathcal{H}$ be any hypotheses class on X of cardinality at most $2^{m-1/p(m)}$, for some polynomial p. Let*

$$BAD = \{\boldsymbol{x}^m \in X^m : \exists h \in \mathcal{H} \text{ consistent with } f \text{ on } \boldsymbol{x}^m \text{ and } Err_{\mathcal{D}}(f,h) \geq \frac{1}{2} - \frac{\ln(2)}{4mp(m)}\}.$$

Then

$$\mathbf{Pr}_{\boldsymbol{x}^m \in \mathcal{D}^m}[BAD] \leq 1 - \frac{1}{1 + \frac{2p(m)}{\ln(2)}}.$$

Proof: We repeatedly use the following for proving that some inequality $a \leq b$ holds. We find an overestimate $\tilde{a}$ of a (i.e. $a \leq \tilde{a}$) and an underestimate of $\tilde{b}$ of b (i.e. $\tilde{b} \leq b$). Then for $a \leq b$ to hold it suffices to show that $\tilde{a} \leq \tilde{b}$.

Let $p'(m) = 4mp(m)/\ln(2)$ and $p''(m) = 1 + 2p(m)/\ln(2)$. For each $h \in \mathcal{H}$, let

$$\mathrm{BAD}_h = \{\boldsymbol{x}^m \in X^m : h \text{ is consistent with } f \text{ on } \boldsymbol{x}^m \text{ and } \mathrm{Err}_{\mathcal{D}}(f,h) \geq \frac{1}{2} - 1/p'(m)\}.$$

Note $\mathrm{BAD} = \cup_{h \in \mathcal{H}} \mathrm{BAD}_h$. Clearly $\mathbf{Pr}_{\boldsymbol{x}^m \in \mathcal{D}^m}[\mathrm{BAD}_h] \leq (1 - \mathrm{Err}_{\mathcal{D}}(f,h))^m \leq (\frac{1}{2} + 1/p'(m))^m$ and thus

$$\begin{aligned}\mathbf{Pr}_{\boldsymbol{x}^m \in \mathcal{D}^m}[\mathrm{BAD}] &\leq 2^{m-1/p(m)}\left(\frac{1}{2} + \frac{1}{p'(m)}\right)^m \\ &= 2^{-1/p_2(m)}\left(1 + \frac{2}{p'(m)}\right)^m \\ &\leq 2^{-1/p(m)} e^{2m/p'(m)}.\end{aligned}$$

To show $\mathbf{Pr}_{\boldsymbol{x}^m \in \mathcal{D}^m}[\mathrm{BAD}] \leq 1 - 1/p''(m)$, it suffices to show that $2^{-1/p(m)} e^{2m/p'(m)} \leq 1 - 1/p''(m)$. Taking logarithms on both sides we get

$$\frac{2m}{p'(m)} - \frac{\ln(2)}{p(m)} \leq \ln(1 - \frac{1}{p''(m)})$$

$$\Leftrightarrow \frac{2m}{p'(m)} - \ln(1 - \frac{1}{p''(m)}) \leq \frac{\ln(2)}{p(m)}.$$

Since $-\ln(1 - 1/p''(m)) \leq \frac{1/p''(m)}{1 - 1/p''(m)} = \frac{1}{p''(m)-1}$, it suffices to show that

$$\frac{2m}{p'(m)} + \frac{1}{p''(m) - 1} \leq \frac{\ln(2)}{p(m)}.$$

The above is implied by $2m/p'(m) \leq \frac{1}{2}\ln(2)/p(m)$ and $1/(p''(m) - 1) \leq \frac{1}{2}\ln(2)/p(m)$. These last two inequalities are follow from the choice of $p'(m) = 4mp(m)/\ln(2)$ and $p''(m) = 1 + 2p(m)/\ln(2)$. Although these choices suffice, there are other choices for the polynomials p' and p''. □

Theorem 3 *If there is a weak Occam algorithm for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ then this algorithm is a weak learning algorithm for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$.*

Proof: Let p_1 and p_2 be the polynomials specifying the weak Occam Algorithm A and $\mathcal{H}_{n,s,m}$ be its hypotheses class when the parameters are n and s. The proof applies the above lemma as follows: for any n and s, let $\mathcal{D}$ be any distribution on $X = X_n$, f be any concept in $\mathcal{F}_s$, $m = p_1(n,s)$ and $\mathcal{H} = \mathcal{H}_{n,s,m}$. When given a sample $\mathrm{sam}_f(\boldsymbol{x}^m)$, where $\boldsymbol{x}^m \in X^m$, A outputs a hypothesis of $\mathcal{H}$ which is of cardinality at most $2^{m-1/p_2(m)}$. Denote the output hypothesis by $A[\mathrm{sam}_f(\boldsymbol{x}^m)]$.

Define BAD as in the Lemma 2 with polynomial $p(m)$ set to $p_2(m)$. Then

$$\mathbf{Pr}_{\boldsymbol{x}^m \in \mathcal{D}^m}\left[\mathrm{Err}_{\mathcal{D}}(f, A[\mathrm{sam}_f(\boldsymbol{x}^m)]) \geq \frac{1}{2} - \frac{\ln(2)}{4mp_2(m)}\right] \leq \mathbf{Pr}_{\boldsymbol{x}^m \in \mathcal{D}^m}[\mathrm{BAD}].$$

From Lemma 2, the latter is at most $1 - 1/(1 + 2p_2(m)/\ln(2))$ and A is a weak learning algorithm. □

If the total running time of the weak Occam algorithm is polynomial in n and s and the algorithm outputs polynomially evaluatable hypotheses, then the Occam algorithm is a computationally efficient weak learning

algorithm. Note that a strong Occam algorithm produces hypotheses with smaller error when the sample size m is increased [BEHW87] and for some polynomial choice of m the strong Occam algorithm becomes a strong learning algorithm. This is not necessarily true for a weak Occam algorithm as the error $(1/2 - 1/p_2(m))$ approaches $1/2$ as m increases. Instead, the conversion algorithms of [Sch90, Fre90] would use the weak Occam algorithm repeatedly for a number of different samples of size $p_1(n,s)$, where $p_1(n,s)$ is the size of the sample expected by the Occam algorithm when the parameters are n and s. The samples are drawn according to various filtered distributions and the resulting hypotheses are combined using the majority function.

Lemma 2 and Theorem 3 are nearly tight in view of the following Occam algorithm which is not a weak learning algorithm because it outputs consistent hypotheses with error exactly $1/2$. Let X_m consist of all bitvectors of length m and $\mathcal{F}_m = \{f_{v,b} : v \in X_m, b \in \{0,1\}\}$, where $f_{v,b}(v') \equiv v \cdot v' - b \bmod 2$ for any $v \in X_m$. Let $\bar{0}$ denote the all zero vector. The concepts $f_{\bar{0},0}$ and $f_{\bar{0},1}$ label the whole domain with 0 and 1, respectively. Call those concepts the *trivial concepts* and the remaining concepts of F_m the non-trivial concepts.

Consider the Occam algorithm that when given m for both parameters n and s, always requests m examples and forms its hypothesis as follows. If any of the m examples are labeled with 1 then it outputs any nontrivial concept in F_m that is consistent with the sample. If all m examples are labeled with 0 then the Occam algorithm forms a matrix M from the examples (each example becomes a row of M). If M is singular then it outputs a hypothesis $f_{v,0}$ such that $Mv \equiv \bar{0}$ and $v \neq \bar{0}$. If M is non-singular it outputs the unique hypothesis $f_{v,1}$ s.t. $Av \equiv \bar{1}$. Again $v \neq \bar{0}$.

Note that in all cases the hypothesis output by this algorithm is consistent with the sample and is a nontrivial concept of F_m. Thus the hypotheses class used by the algorithm has size $2^{m+1} - 2$. When the target concept is the trivial concept $f_{\bar{0},0} \in F_m$ then all m examples are labeled 0. Furthermore all non-trivial concepts (including the hypothesis output by the algorithm) have error exactly $\frac{1}{2}$ with respect to $f_{\bar{0},0}$ and the uniform distribution on X_m. We conclude that the above Occam algorithm which uses a hypothesis class of size $2^{m+1} - 2$ when given samples of size m is not a weak learning algorithm.

4 Prediction by Information Gain

In this section we describe a prediction algorithm which is a special case of an algorithm introduced by Vovk [Vov90]. The algorithm is presented for the case when there is a probability distribution $\mathcal{P}$ on $\mathcal{F}$ available representing the prior beliefs about the functions of $\mathcal{F}$ being the target. We call this algorithm the Information Gain Prediction Algorithm since it predicts $b \in \{0,1\}$ with probability proportional to the information gain obtained when the correct value is $1-b$. This algorithm is the basis for all our later algorithms. We bound its expected total number of mistakes over a sequence of m trials. The proven bound is a special case of the bound given in [Vov90].

Definition 4 The Information Gain Prediction Algorithm $G^{\mathcal{P}}$.

$G^{\mathcal{P}}$ is given a sample S, the instance x, and a random $r \in [0,1]$. Algorithm $G^{\mathcal{P}}$ also makes use of the prior $\mathcal{P}$ on the concept class $\mathcal{F}$.

If $V^{\mathcal{P}}(S) = 0$ then Algorithm $G^{\mathcal{P}}$ predicts arbitrarily. Otherwise, Algorithm $G^{\mathcal{P}}$ computes both $I^{\mathcal{P}}(S,(x,1))$ and $I^{\mathcal{P}}(S,(x,0))$ using the prior on the concept class $\mathcal{F}$, and outputs the prediction 0 if

$$r \leq \frac{I^{\mathcal{P}}(S,(x,1))}{I^{\mathcal{P}}(S,(x,1)) + I^{\mathcal{P}}(S,(x,0))} = \frac{-\lg \frac{V^{\mathcal{P}}(S,(x,1))}{V^{\mathcal{P}}(S)}}{-\lg \frac{V^{\mathcal{P}}(S,(x,1))}{V^{\mathcal{P}}(S)} - \lg \frac{V^{\mathcal{P}}(S,(x,0))}{V^{\mathcal{P}}(S)}}$$

and 1 otherwise.

Thus, for $b \in \{0,1\}$, the Information Gain Prediction Algorithm's prediction is b with probability

$$\frac{I^{\mathcal{P}}(S,(x,1-b))}{I^{\mathcal{P}}(S,(x,1)) + I^{\mathcal{P}}(S,(x,0))}$$

when $V^{\mathcal{P}}(S) > 0$.

Even when $V^{\mathcal{P}}(S) > 0$, it is possible that either $V^{\mathcal{P}}(S,(x,1)) = 0$ or $V^{\mathcal{P}}(S,(x,0)) = 0$. In that case if $V^{\mathcal{P}}(S,(x,1)) = 0$ then $I^{\mathcal{P}}(S,(x,1)) = \infty$ and $G^{\mathcal{P}}$ predicts 0 with probability 1. Similarly, $G^{\mathcal{P}}$ always predicts 1 when $V^{\mathcal{P}}(S,(x,0)) = 0$.

Lemma 5 *For all $\mathcal{P}$ on $\mathcal{F}$, $S \in (X \times \{0,1\})^*$, and $x \in X$: if $V^{\mathcal{P}}(S) > 0$ then $I^{\mathcal{P}}(S,(x,0)) + I^{\mathcal{P}}(S,(x,1)) \geq 2$.*

Proof: Using the definition of $I^{\mathcal{P}}$, it suffices to show that

$$-\lg \frac{V^{\mathcal{P}}(S,(x,0))}{V^{\mathcal{P}}(S)} - \lg \frac{V^{\mathcal{P}}(S,(x,1))}{V^{\mathcal{P}}(S)} \geq 2.$$

Since $V^{\mathcal{P}}(S) = V^{\mathcal{P}}(S,(x,0)) + V^{\mathcal{P}}(S,(x,1)) > 0$, this is equivalent to showing, $\forall p \in [0,1]$, that $-\lg p - \lg(1-p) \geq 2$. Clearly the left hand side is minimized when $p = \frac{1}{2}$ and in that case the inequality is tight. □

Note that the lemma holds when either $V^{\mathcal{P}}(S,(x,0)) = 0$ or $V^{\mathcal{P}}(S,(x,1)) = 0$, as then $I^{\mathcal{P}}(S,(x,0)) + I^{\mathcal{P}}(S,(x,1)) = \infty$.

We now state the well known fact that information is additive [HKS91].

Lemma 6 *If $V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x})) > 0$ then*

$$\sum_{t=1}^{|\boldsymbol{x}|} I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^t)) = -\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x})).$$

Proof:

$$\begin{aligned}\sum_{t=1}^{|\boldsymbol{x}|} I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^t)) &= \sum_{t=1}^{|\boldsymbol{x}|} -\lg \frac{V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^t))}{V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^{t-1}))} \\ &= -\lg \frac{V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}))}{V^{\mathcal{P}}(\Lambda)} \\ &= -\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}))\end{aligned}$$

We are now ready to bound the probability that Algorithm $G^{\mathcal{P}}$ predicts incorrectly.

Theorem 7 *For all $\mathcal{P}$ on $\mathcal{F}$, $f \in \mathcal{F}$, and $\boldsymbol{x} \in X^*$:*

$$\sum_{t=1}^{|\boldsymbol{x}|} \mathbf{M}_{G^{\mathcal{P}},f}(\boldsymbol{x}^t) \leq -\frac{1}{2} \lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x})).$$

Proof: If $V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x})) = 0$ then the theorem holds trivially. Otherwise, for each $1 \leq t \leq |\boldsymbol{x}|$,

$$\begin{aligned}&\mathbf{M}_{G^{\mathcal{P}},f}(\boldsymbol{x}^t) \\ &= \frac{I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^t))}{I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^{t-1}),(x_t,1)) + I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^{t-1}),(x_t,1))} \\ &\leq \frac{1}{2} I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^t))\end{aligned}$$

using Lemma 5. By the additivity of information (Lemma 6),

$$\sum_{t=1}^{|\boldsymbol{x}|} \mathbf{M}_{G^{\mathcal{P}},f}(\boldsymbol{x}^t) \leq -\frac{1}{2} \lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}))$$

as desired. □

As done in [HKS91] we consider the case where the "perceived prior" $\mathcal{P}$ that the algorithm is using to compute information gains is different from the "true" prior $\mathcal{Q}$ according to which the target function f is drawn. Intuitively the performance of a prediction algorithm which calculates volumes with respect to $\mathcal{P}$ degrades as the "distance" between the two priors increases.

Since Theorem 7 bounds the performance of Prediction Algorithm $G^{\mathcal{P}}$ for all target functions, we can get bounds that hold for any pair of priors $\mathcal{P}$ and $\mathcal{Q}$.

$$\mathbf{E}_{f\in\mathcal{Q}}\left[\sum_{t=1}^{m} \mathbf{M}_{G^{\mathcal{P}},f}(\boldsymbol{x}^t)\right] \leq -\frac{1}{2}\mathbf{E}_{f\in\mathcal{Q}}\left[\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}))\right].$$

Similar, but weaker, results have been shown [HKS91] for the Bayes[7] Algorithm and the Gibbs Algorithm. Both of these algorithm use volumes with respect to the perceived prior $\mathcal{P}$ to determine their prediction. Specifically, they show that for any pair of priors $\mathcal{P}$ and $\mathcal{Q}$:

$$\mathbf{E}_{f\in\mathcal{Q}}\left[\sum_{t=1}^{m} \mathbf{M}_{Bayes^{\mathcal{P}},f}(\boldsymbol{x}^t)\right] \leq -\mathbf{E}_{f\in\mathcal{Q}}\left[\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}))\right],$$

and

$$\mathbf{E}_{f\in\mathcal{Q}}\left[\sum_{t=1}^{m} \mathbf{M}_{Gibbs^{\mathcal{P}},f}(\boldsymbol{x}^t)\right] \leq -\ln(2)\mathbf{E}_{f\in\mathcal{Q}}\left[\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}))\right].$$

We are interested in the extreme case where the true prior $\mathcal{Q}$ is concentrated on one function $f \in \mathcal{F}$. In this worst case setting, the bounds on the expected total number of mistakes for Algorithm $G^{\mathcal{P}}$, Bayes, and Gibbs all grow as $-\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}))$, but with different constant factors. By Theorem 7, the constant factor for Algorithm $G^{\mathcal{P}}$ is at most 1/2. The above bounds from [HKS91] give constant factors 1 and ln(2) for the Bayes and Gibbs prediction algorithms, respectively ($\ln 2 \approx 0.7$).

It is easy to show that these constants for the Bayes and Gibbs Algorithms cannot be improved. Consider a single instance x (i.e. $m = 1$) and a prior $\mathcal{P}$ that is concentrated on only 2 functions, f and g, where $f(x) \neq g(x)$.

For the Bayes example, set

$$\mathcal{P}(f) = \frac{1}{2} + \epsilon, \text{ and } \mathcal{P}(g) = \frac{1}{2} - \epsilon$$

so that

$$\mathbf{M}_{Bayes^{\mathcal{P}},g}(x) = 1 \text{ and } -\lg V^{\mathcal{P}}(\mathrm{sam}_g(x)) = -\lg(\tfrac{1}{2}-\epsilon).$$

For Gibbs, set

$$\mathcal{P}(f) = \epsilon, \text{ and } \mathcal{P}(g) = 1 - \epsilon$$

so that

$$\mathbf{M}_{Gibbs^{\mathcal{P}},g}(x) = \epsilon \text{ and } -\lg V^{\mathcal{P}}(\mathrm{sam}_g(x)) = -\lg(1-\epsilon).$$

By letting ϵ go to zero in both examples it can be seen that the constant factors of 1 for Bayes and ln(2) for Gibbs cannot be improved.

All weak learning algorithms described in this paper are based on the Information Gain Prediction Algorithm and the factor of 1/2 seems to be necessary in the derivation of our bounds for these algorithms.

Recall that the dichotomies of $\mathcal{F}$ induced by $\boldsymbol{x}$ is the set of samples $\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x}) = \{\mathrm{sam}_f(\boldsymbol{x}) : f \in \mathcal{F}\}$. This set has one member for each possible way that functions in $\mathcal{F}$ label the instances of $\boldsymbol{x}$.

Definition 8 *The* partition entropy *of $\mathcal{P}$ with respect to $\mathcal{F}$ and $\boldsymbol{x}$ is*

$$H^{\mathcal{P}}_{\mathcal{F}}(\boldsymbol{x}) = \sum_{S\in\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x})} -V^{\mathcal{P}}(S)\lg V^{\mathcal{P}}(S).$$

[7] The Bayes Algorithm is defined for any loss function. Our case corresponds to the 0,1 loss. Thus the expected loss is the probability of predicting wrong.

This allows us to state an important corollary of Theorem 7.

Corollary 9 *For all $\boldsymbol{x} \in X^*$,*

$$\mathbf{E}_{f\in\mathcal{P}}\left[\sum_{t=1}^{|\boldsymbol{x}|}\mathbf{M}_{G^{\mathcal{P}},f}(\boldsymbol{x}^t)\right] \leq \frac{1}{2}H^{\mathcal{P}}_{\mathcal{F}}(\boldsymbol{x}) \leq \frac{1}{2}\lg|\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x})| \leq \frac{1}{2}|\boldsymbol{x}|.$$

The same upper bounds on the performance of the Bayes Algorithm and Gibbs Algorithm is given in [HKS91]. However to get the factor of 1/2 for Bayes and Gibbs it was necessary to average over $\mathcal{F}$, whereas in the case of Algorithm $G^{\mathcal{P}}$ the corollary follows directly from the worst case bound proven in Theorem 7.

5 The Random Position Prediction Algorithm

In this section we change our focus from minimizing the total number of expected mistakes to minimizing the probability of a mistake on the last (mth) instance. The desired bound is to hold uniformly for all $f \in \mathcal{F}$. We define a modification of the $G^{\mathcal{P}}$ whose probability of mistake in the m-th trial is an m-th fraction of the total number of mistakes in all m trials. As in the previous section, we assume that a probability distribution $\mathcal{P}$ on $\mathcal{F}$ representing prior belief is available.

Definition 10 Randomized Position Information Gain Prediction Algorithm $R^{\mathcal{P}}$.

$R^{\mathcal{P}}$ is given a sample $S = \mathrm{sam}_f(\boldsymbol{x})$ where $\boldsymbol{x} \in X^*$, an instance $x \in X$, and a random $r \in [0,1]$. Algorithm $R^{\mathcal{P}}$ also makes use of the prior $\mathcal{P}$ on the concept class $\mathcal{F}$.

If $V^{\mathcal{P}}(sam_f(\boldsymbol{x}^{t-1})) = 0$ then the information gains are undefined and Algorithm $R^{\mathcal{P}}$ predicts arbitrarily. Otherwise Algorithm $R^{\mathcal{P}}$ starts by splitting r into two random numbers, t and r' such that t is uniformly distributed over $\{1,2,\ldots,|\boldsymbol{x}|\}$ and r' is uniformly distributed over $[0,1]$. Assuming $V^{\mathcal{P}}(sam_f(\boldsymbol{x}^{t-1})) > 0$, the prediction of Algorithm $R^{\mathcal{P}}$ is 0 when

$$r' \leq \frac{I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^{t-1}),(x,1))}{I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^{t-1}),(x,1)) + I^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^{t-1}),(x,0))}$$

and 1 otherwise. Note that the $\boldsymbol{x}^{t-1}$ is always a prefix of $\boldsymbol{x}$, and thus the sequence $\mathrm{sam}_f(\boldsymbol{x}^{t-1})$ is always available to the algorithm.

We can obtain a good bound on $\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{x})$ by averaging over all permutations of $\boldsymbol{x}$. This method has been used in [HLW]. Recall that $U(\boldsymbol{x})$ denotes the uniform distribution on all permutations of $\boldsymbol{x}$.

Theorem 11 *For all $f \in \mathcal{F}$, $m \in \mathbf{N}$, and $\boldsymbol{x}^m \in X^m$:*

$$\mathbf{E}_{\boldsymbol{y}^m\in U(\boldsymbol{x}^m)}\left[\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{y}^m)\right] \leq -\frac{1}{2m}\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^m)).$$

Proof: If $V^{\mathcal{P}}(f(\boldsymbol{x}^m)) = 0$ then the right hand side is infinite, and the theorem is trivial. We now assume that $V^{\mathcal{P}}(f(\boldsymbol{x}^m)) > 0$. Recall that $\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{y}^m)$ is the probability that, after seeing $\mathrm{sam}_f(\boldsymbol{y}^{m-1})$, the prediction of Algorithm $R^{\mathcal{P}}$ on y_m is different from $f(y_m)$. When

$$R(\boldsymbol{y},f,t) = \frac{I^{\mathcal{P}}(f(\boldsymbol{y}^{t-1}),(y_m,f(y_m)))}{I^{\mathcal{P}}(f(\boldsymbol{y}^{t-1}),(y_m,0)) + I^{\mathcal{P}}(f(\boldsymbol{y}^{t}),(y_m,0))}$$

then

$$\begin{aligned}
&\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{y})\right] \\
&= \frac{1}{m}\sum_{t=1}^{m}\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[R(\boldsymbol{y},f,t)\right] \\
&\leq \frac{1}{2m}\sum_{t=1}^{m}\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[I^{\mathcal{P}}(f(\boldsymbol{y}^{t-1}),(y_m,f(y_m)))\right] \\
&= \frac{1}{2m}\sum_{t=1}^{m}\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[I^{\mathcal{P}}(f(\boldsymbol{y}^{t-1}),(y_t,f(y_t)))\right] \\
&= \frac{1}{2m}\sum_{t=1}^{m}\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[I^{\mathcal{P}}(f(\boldsymbol{y}^{t}))\right] \\
&= \frac{1}{2m}\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\sum_{t=1}^{m}I^{\mathcal{P}}(f(\boldsymbol{y}^{t}))\right] \\
&= \frac{1}{2m}\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[-\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{y}^m))\right] \quad \text{(Lemma 6)} \\
&= -\frac{1}{2m}\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{y}^m)) \\
&= -\frac{1}{2m}\lg V^{\mathcal{P}}(\mathrm{sam}_f(\boldsymbol{x}^m))
\end{aligned}$$

as desired. □

There are cases where $\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{\mathrm{Bayes}^{\mathcal{P}},f}(\boldsymbol{y})\right]$, $\mathbf{E}_{\boldsymbol{y}\in\mathcal{D}^m}\left[\mathbf{M}_{\mathrm{Gibbs}^{\mathcal{P}},f}(\boldsymbol{y})\right]$, and $\mathbf{E}_{\boldsymbol{y}\in\mathcal{D}^m}\left[\mathbf{M}_{G^{\mathcal{P}},f}(\boldsymbol{y})\right]$ all equal 1/2. For example, consider an $\boldsymbol{x}^m$ and concept class $\mathcal{F}$ of $m+1$ functions: f_0 which maps all x_t to 0 and for each $1 \leq i \leq |\boldsymbol{x}|$, the function f_i mapping x_i to 1 and all other x_j to 0. Now if the target function is f_0 and $\mathcal{P}$ is uniform on $\mathcal{F}$, then all three prediction algorithms have a mistake probability of $\frac{1}{2}$ on the last trial. However, for the above example we have the following bound for $R^{\mathcal{P}}$ (Theorem 11):

$$\mathbf{E}_{\boldsymbol{y}\in\mathcal{D}^m}\left[\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{y})\right] \leq \frac{\lg(m+1)}{2m}.$$

This shows that that Algorithm $R^{\mathcal{P}}$ has a lower probability of predicting wrong in the last trial than either of the other three algorithms. The crucial feature responsible for the improved performance of Algorithm $R^{\mathcal{P}}$ over the Algorithm $G^{\mathcal{P}}$ (and Vovk's Aggregating Strategy [Vov90]) is the fact that $R^{\mathcal{P}}$ predicts from a random position.

It is not necessary to use random bits to select a random position. The sample can be sorted on the instances and only that part of the sorted sample whose instances are

less then x_m is used to predict on x_m. This gives the same bound (averaged over permutations) as given in Theorem 11 for Algorithm $R^{\mathcal{P}}$.

Let us return to our method of predicting from a random position as done in $R^{\mathcal{P}}$. One can also modify Bayes and Gibbs to predict from a random position. The bounds are again of the form "constant times $-\frac{1}{m}\lg V^{\mathcal{P}}(\text{sam}_f(\boldsymbol{x}^m))$" for all functions $f \in \mathcal{F}$. However the constant obtainable for the modified Bayes and Gibbs are 1 and ln(2), respectively, instead of the constant 1/2 for $R^{\mathcal{P}}$ given in Theorem 11.

There is another prediction algorithm that achieves the factor of 1/2 for certain "uniform" priors (the uniform prior on $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$ discussed in the next section). This is the prediction algorithm of [HLW] based on the 1-inclusion graph [Bon72, AHW87]. The disadvantage of this algorithm is that it is not computationally efficient whereas we will show that $R^{\mathcal{P}}$ leads to an efficient weak learning algorithm.

We conclude this section by stating two simple corollaries of Theorem 11.

Corollary 12 *For all $\mathcal{D}$ on X, $m \geq 1$, and $f \in \mathcal{F}$:*

$$\mathbf{E}_{\boldsymbol{x}\in\mathcal{D}^m}\left[\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{x})\right] \leq -\frac{1}{2m}\mathbf{E}_{\boldsymbol{x}\in D^m}\left[\lg V^{\mathcal{P}}(\text{sam}_f(\boldsymbol{x}))\right].$$

Proof:

$$\begin{aligned}
&\mathbf{E}_{\boldsymbol{x}\in\mathcal{D}^m}\left[\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{x})\right] \\
&\quad = \mathbf{E}_{\boldsymbol{x}\in\mathcal{D}^m}\left[\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{y})\right]\right] \\
&\quad \leq \mathbf{E}_{\boldsymbol{x}\in\mathcal{D}^m}\left[-\frac{1}{2m}\lg V^{\mathcal{P}}(\text{sam}_f(\boldsymbol{x}^m))\right] \\
&\quad = -\frac{1}{2m}\mathbf{E}_{\boldsymbol{x}\in\mathcal{D}^m}\left[\lg V^{\mathcal{P}}(\text{sam}_f(\boldsymbol{x}^m))\right]
\end{aligned}$$

□

Corollary 13 *For all $\mathcal{D}$ on X, $\mathcal{P}$ on $\mathcal{F}$, and $m \geq 1$:*

$$\mathbf{E}_{f\in\mathcal{F},\boldsymbol{x}\in\mathcal{D}^m}\left[\mathbf{M}_{R^{\mathcal{P}},f}(\boldsymbol{x})\right] \leq -\frac{1}{2m}\mathbf{E}_{\boldsymbol{x}\in\mathcal{D}^m}\left[H^{\mathcal{P}}_{\mathcal{F}}(\boldsymbol{x})\right].$$

The same bound as in the above Corollary can be shown for the Bayes Algorithm and the Gibbs Algorithm [HKS91]. However to obtain these bounds averaging over the prior $\mathcal{P}$ is needed, whereas for Algorithm $R^{\mathcal{P}}$ the above bound directly follows from Theorem 11, whose bound holds for all functions.

6 Dichotomies

In this section we shift our focus from concept classes to the set of dichotomies, $\text{sam}_{\mathcal{F}}(\boldsymbol{x})$. Recall that $\text{sam}_{\mathcal{F}}(\boldsymbol{x}) = \{\text{sam}_f(\boldsymbol{x}) : f \in \mathcal{F}\}$.

The previous section described Algorithm $R^{\mathcal{P}}$ which receives three inputs: a sample in $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^{m-1})$, an additional instance $x_m \in X$, and a random number r drawn uniformly from $[0,1]$. Algorithm $R^{\mathcal{P}}$ also uses the prior $\mathcal{P}$ on the concept class $\mathcal{F}$. Actually, Algorithm $R^{\mathcal{P}}$ uses the prior only to evaluate the volumes $V^{\mathcal{P}}(\text{sam}_f(\boldsymbol{x}^{t-1}),(x_m,1))$ and $V^{\mathcal{P}}(\text{sam}_f(\boldsymbol{x}^{t-1}),(x_m,0))$ for some $1 \leq t \leq m$. The following shows how these volumes can be computed from the volumes of samples in $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$.

For any $m \in \mathbf{N}$, $\boldsymbol{x} \in X^m$, $1 \leq t \leq m$, $f \in F$ and $b \in \{0,1\}$.

$$V^{\mathcal{P}}(\text{sam}_f(\boldsymbol{x}^{t-1}),(x_m,b)) \;=\; \sum_{S\in T} V^{\mathcal{P}}(S)$$

where $T = \{S : S \in \text{sam}_F(\boldsymbol{x})$ and S contains all examples of $\text{sam}_f(\boldsymbol{x}^{t-1}),(x_m,b)\}$.

This motivates using a new prior on $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$, rather than the old prior on $\mathcal{F}$, when the algorithm is to predict on x_m from a sample of $\boldsymbol{x}^{m-1}$. Clearly this new prior depends on $\boldsymbol{x}^m$. Essentially, $\boldsymbol{x}^m$ defines a set of equivalence classes on $\mathcal{F}$. Each dichotomy in $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$ represents one of these equivalence classes. The probability of a sample under the new prior is the sum (or integral) of the probabilities given to the functions in the equivalence class by the old prior.

One interesting prior is the uniform prior on $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$. Using this prior, computing the volume of a sample S amounts to counting the number of ways S can be "extended" to samples in $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$.

Recall that $\text{sam}_*(\boldsymbol{x})$ is the set of all $2^{|\boldsymbol{x}|}$ possible samples whose examples from the sequence $\boldsymbol{x}$.

Definition 14 *For any subsequence $\boldsymbol{y}$ of $\boldsymbol{x}$ and sample $S \in \text{sam}_*(\boldsymbol{y})$, $N^{\boldsymbol{x}}_{\mathcal{F}}(S)$ is the number of samples in $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$ that contain all examples of S.*

Note that $N^{\boldsymbol{x}}_{\mathcal{F}}(\Lambda) = |\text{sam}_{\mathcal{F}}(\boldsymbol{x})|$ and for any $f \in \mathcal{F}$, $N^{\boldsymbol{x}}_{\mathcal{F}}(\text{sam}_f(\boldsymbol{x})) = 1$. When $\mathcal{P}$ is the uniform prior on $\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$, $V^{\mathcal{P}}(S) = N^{\boldsymbol{x}}_{\mathcal{F}}(S)/\,|\text{sam}_{\mathcal{F}}(\boldsymbol{x})|$.

We now generalize our notation of prior so that any of the 2^m samples in $\text{sam}_*(\boldsymbol{x}^m)$ can have positive probability, rather than just those of $\text{sam}_F(\boldsymbol{x}^m)$. We are particularly interested in weighted combinations of the uniform prior on $\text{sam}_{\mathcal{F}}(\boldsymbol{x})$ and the uniform prior on $\text{sam}_*(\boldsymbol{x})$. These weighted combinations ensure that each sample in $\text{sam}_*(\boldsymbol{x})$ has a significant probability, a fact that will prove useful in Section 7. The following definition provides notation for these combination priors.

Definition 15 *For any $m \in \mathbf{N}$, $\boldsymbol{x} \in X^m$ and $c \in [0,1]$, let $U^c_{\mathcal{F}}(\boldsymbol{x}^m)$ denote the following prior on $\text{sam}_*(\boldsymbol{x}^m)$. For any $S \in \text{sam}_*(\boldsymbol{x}^m)$:*

if $S \in \text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$ then

$$U^c_{\mathcal{F}}(\boldsymbol{x}^m)(S) \;=\; \frac{1}{(1-c)\,|\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)| + c2^m}$$

and if $S \in (\text{sam}_*(\boldsymbol{x}^m) - \text{sam}_{\mathcal{F}}(\boldsymbol{x}^m))$ then

$$U^c_{\mathcal{F}}(\boldsymbol{x}^m)(S) \;=\; \frac{c}{(1-c)\,|\text{sam}_{\mathcal{F}}(\boldsymbol{x}^m)| + c2^m}.$$

Note that $U^0_{\mathcal{F}}(\boldsymbol{x}^m)$ is the uniform distribution on $\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x}^m)$ and $U^1_{\mathcal{F}}(\boldsymbol{x}^m)$ is the uniform distribution on $\mathrm{sam}_*(\boldsymbol{x}^m)$.

The above definition leads to the following expression for the volume of a sample with respect to the prior $U^c_{\mathcal{F}}(\boldsymbol{x})$.

Lemma 16 *For any $m \in \mathbf{N}$, $\boldsymbol{x} \in X^m$, any subsequence $\boldsymbol{y}$ of $\boldsymbol{x}$, and any $S \in \mathrm{sam}_*(\boldsymbol{y})$:*

$$V^{U^c_{\mathcal{F}}(\boldsymbol{x})}(S) = \frac{(1-c)N^{\boldsymbol{x}}_{\mathcal{F}}(S) + c2^{m-|\boldsymbol{y}|}}{(1-c)\,|\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x}^m)| + c2^m}$$

Proof: Follows immediately from the definitions of volume and $U^c_{\mathcal{F}}(\boldsymbol{x})$. □

Algorithm R^c is the prediction Algorithm $R^{\mathcal{P}}$ using the prior $\mathcal{P} = U^c_{\mathcal{F}}(\boldsymbol{x}^m)$, where $\boldsymbol{x}^{m-1}$ are the instances in the sample and x_m is the additional unlabeled example.

Definition 17 Algorithm R^c
Algorithm R^c is given the parameter $c \in [0,1]$, a sample $S = \mathrm{sam}_f(\boldsymbol{x})$ where $\boldsymbol{x} \in X^{m-1}$, an additional instance $x_m \in X$, and a random number $r \in [0,1]$. Algorithm R^c first splits r into a random position $t \in \{1,\ldots,m\}$, and an independent random number $r' \in [0,1]$. It then determines $u_0 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1}),(x_m,0))/2^{m-t}$ and $u_1 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1}),(x_m,1))/2^{m-t}$. Since S is a sample of some $f \in \mathcal{F}$, either u_0 or u_1 will be positive. If $c = 0$ and $u_0 = 0$ then $R^{\mathcal{P}}$ predicts 1, and if $c = 0$ and $u_1 = 0$ then $R^{\mathcal{P}}$ predicts 0. Otherwise, at least two of c, u_0, and u_1 are positive, and Algorithm R^c predicts 0 when

$$r' \le \frac{\lg\left(1+\frac{(1-c)u_0+c}{(1-c)u_1+c}\right)}{\lg\left(1+\frac{(1-c)u_0+c}{(1-c)u_1+c}\right)+\lg\left(1+\frac{(1-c)u_1+c}{(1-c)u_0+c}\right)}$$

and 1 when r' is greater than the ratio.

Note that $u_0 + u_1 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^t))/2^{m-t-1}$ which is likely to be less than one. Algorithm R^c predicts 1 with probability

$$\frac{\lg\left(1+\frac{(1-c)u_1+c}{(1-c)u_0+c}\right)}{\lg\left(1+\frac{(1-c)u_0+c}{(1-c)u_1+c}\right)+\lg\left(1+\frac{(1-c)u_1+c}{(1-c)u_0+c}\right)}.$$

Furthermore the above equals

$$\frac{\lg\frac{(n_0+n_1)(1-c)+c2^{m-t+1}}{n_0(1-c)+c2^{m-t}}}{\lg\frac{(n_0+n_1)(1-c)+c2^{m-t+1}}{n_0(1-c)+c2^{m-t}}+\lg\frac{(n_0+n_1)(1-c)+c2^{m-t+1}}{n_1(1-c)+c2^{m-t}}}.$$

where $n_0 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1}),(x_m,0))$ and $n_1 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1}),(x_m,1))$. Thus R^c is the prediction Algorithm $R^{\mathcal{P}}$ where $\mathcal{P} = U^c_{\mathcal{F}}(\boldsymbol{x}^m)$.

From Theorem 11 and Definition 15 we get the following for Algorithm R^c.

Corollary 18 *For all $f \in \mathcal{F}$, $m \in \mathbf{N}$, $\boldsymbol{x}^m \in X^m$, and $0 \le c \le 1$:*

$$\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x}^m)}\left[\mathbf{M}_{R^c,f}(\boldsymbol{y})\right] \le \frac{1}{2m}\lg\left(|\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x})|\,(1-c)+c2^m\right)$$

$$\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x}^m)}\left[\mathbf{M}_{R^0,f}(\boldsymbol{y})\right] \le \frac{1}{2m}\lg\left(|\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x})|\right)$$

$$\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x}^m)}\left[\mathbf{M}_{R^1,f}(\boldsymbol{y})\right] = \frac{1}{2}.$$

As in Corollary 12, the expectations over permutations of $\boldsymbol{x}^m$ can be replaced by expectations over $\boldsymbol{x}^m$ drawn at random from $\mathcal{D}^m$.

For the case of the uniform prior on $\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x})$ the 1-inclusion graph prediction algorithm of [HLW] (here denoted by 1-inc) is a good predictor. For all $f \in \mathcal{F}$, $m \in \mathbf{N}$, and $\boldsymbol{x}^m \in X^m$ its performance is bounded as follows:

$$\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x}^m)}\left[\mathbf{M}_{1\text{-inc},f}(\boldsymbol{y})\right] \le \frac{\mathrm{maxdens}_{\mathcal{F}}(\boldsymbol{x})}{m},$$

where $\mathrm{maxdens}_{\mathcal{F}}(\boldsymbol{x})$ is the maximum density (number of edges over number of vertices) of any subgraph of the 1-inclusion graph with respect to $\mathcal{F}$ and $\boldsymbol{x}$. In [HLW] it is shown that this density is upper bounded by the Vapnik-Chervonenkis dimension of the class $\mathcal{F}$ [VC71, BEHW89]. It is easy to show by induction on m that $\frac{1}{2}\lg(|\mathrm{sam}_{\mathcal{F}}(\boldsymbol{x})|)$ is a second upper bound on the density. Thus the probability that the 1-inclusion graph prediction strategy predicts wrong on the last trial is bounded by the same bound given for R^0 in the above corollary.

The parameter c has an interesting effect on the predictions of Algorithm R^c. As c increases from 0 to 1, the probability that R^c predicts 0 (or 1) gets biased more and more heavily towards $\frac{1}{2}$. When $c = 1$, the algorithm ignores the sample and predicts with a random coin flip. Increasing c also increases the bound on $\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x}^m)}[\mathbf{M}_{R^c,f}(\boldsymbol{y})]$, as would be expected since the algorithm's predictions become more and more random.

To gain more insight into how the predictions of R^c are effected by the parameter c we now compare the probabilities that R^c and R^0 predict 1. Recall that $u_0 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1}),(x_m,0))/2^{m-t}$ and $u_1 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1}),(x_m,1))/2^{m-t}$.

R^0	R^c
$\frac{\lg\left(1+\frac{u_1}{u_0}\right)}{\lg\left(1+\frac{u_0}{u_1}\right)+\lg\left(1+\frac{u_0}{u_1}\right)}$	$\frac{\lg\left(1+\frac{(1-c)u_1+c}{(1-c)u_0+c}\right)}{\lg\left(1+\frac{(1-c)u_0+c}{(1-c)u_1+c}\right)+\lg\left(1+\frac{(1-c)u_0+c}{(1-c)u_1+c}\right)}$

If $u_0 = u_1 = \frac{1}{2}$ then both R^c and R^0 predict 1 with probability $\frac{1}{2}$. If $u_0 < u_1$ then both ratios are more than $\frac{1}{2}$, but the probability that R^c predicts 1 is smaller then the probability that R^0 predicts 1. Conversely, if $u_0 > u_1$ then both ratios are less than $\frac{1}{2}$, but the probability that R^c predicts 1 is larger then the probability that R^0 predicts 1. Both of these differences are due to the bias towards $\frac{1}{2}$ induced by c.

7 Randomized Counting

Since $\text{sam}_{\mathcal{F}}(\boldsymbol{x})$ is often exponentially large for interesting concept classes $\mathcal{F}$, an efficient implementation of R^c can not simply compute $u_0 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1},(x_m,0))/2^{m-t}$ and $u_0 = N^{\boldsymbol{x}}_{\mathcal{F}}(f(\boldsymbol{x}^{t-1},(x_m,1))/2^{m-t}$. We use random sampling and a weak consistency for $\mathcal{F}$ to estimate these quantities.

For arbitrary $\mathcal{F}$ and $\boldsymbol{x}^m$, determining if an element of $\text{sam}_*(\boldsymbol{x})$ is in $\text{sam}_{\mathcal{F}}(\boldsymbol{x})$ can be difficult. To avoid this difficulty, we require a weak consistency oracle for $\mathcal{F}$.

Definition 19 *A* weak consistency oracle *for* $\mathcal{F} = \bigcup_s \mathcal{F}_s$ *on* $X = \bigcup_n X_n$ *is given* n, s, m, *and a sample* $S \in \text{sam}_*(\boldsymbol{x}^m)$ *where* $\boldsymbol{x}^m \in (X_n)^m$. *The oracle answers "yes" if* $S = \text{sam}_f(\boldsymbol{x}^m)$ *for some* $f \in \mathcal{F}_s$ *and "no" otherwise. A* polynomial time weak consistency oracle *is a weak consistency whose answers are computed in time polynomial in* n, s, *and* m.

We call these oracles *weak* to emphasize the fact that they do not return functions in $\mathcal{F}_s$, only whether or not there exists a function in $\mathcal{F}_s$ consistent with the sample.

We form an estimate $\hat{u}_0$ for u_0 by creating q new samples of length m. Each new m-sample is constructed by appending the $(t-1)$-sample $\text{sam}_f(\boldsymbol{x}^{t-1})$ with a randomly drawn sample from $\text{sam}_*(x_t, x_{t+1}, \ldots, x_{m-1})$ and finally adding $(x_m, 0)$ at the end. Our estimate of u_0 is the number of these new m-samples in $\text{sam}_{\mathcal{F}}(\boldsymbol{x})$ divided by q, the number of the m-samples created for estimating u_0.

We obtain the estimate $\hat{u}_1$ for u_1 in similarly, except the final example in each of the q samples is set to $(x_m, 1)$ rather $(x_m, 0)$. If q is large enough, then both $|\hat{u}_0 - u_0|$ and $|\hat{u}_1 - u_1|$ are likely to be small. We will use Hoeffding's inequality to measure the accuracy of our estimates.

Note that $(1-c)\hat{u}_0 + c$ is less sensitive to the error $|\hat{u}_0 - u_0|$ the closer c is to 1. The following corollary gives a reasonable choice for c.

Corollary 20 *For all* $f \in \mathcal{F}$, $m \in \mathbf{N}$, *and* $\boldsymbol{x}^m \in X^m$ *for which* $|\text{sam}_{\mathcal{F}}(\boldsymbol{x})| \leq 2^{m-\alpha}$, *if* $c = \frac{\alpha}{2^\alpha - 1}\frac{\ln 2}{4}$, *which is always at most* $\frac{1}{4}$, *then*

$$\mathbf{E}_{\boldsymbol{y} \in U(\boldsymbol{x})}\left[\mathbf{M}_{R^c,f}(\boldsymbol{x})\right] \leq \frac{1}{2} - \frac{3\alpha}{8m}.$$

Proof: The inequality follows from Corollary 18 and our choice for c.

$$\begin{aligned}
&\mathbf{E}_{\boldsymbol{y} \in U(\boldsymbol{x})}\left[\mathbf{M}_{R^c,f}(\boldsymbol{x})\right] \\
&\leq \frac{1}{2m}\lg\left(|\text{sam}_{\mathcal{F}}(\boldsymbol{x})|(1-c) + c2^m\right) \\
&\leq \frac{1}{2m}\lg\left((1-c)2^{m-\alpha} + c2^m\right) \\
&= \frac{1}{2m}(m-\alpha) + \frac{1}{2m}\lg(1-c+c2^\alpha) \\
&\leq \frac{1}{2} - \frac{\alpha}{2m} + \frac{1}{2m}\frac{c(2^\alpha - 1)}{\ln 2} \\
&= \frac{1}{2} - \frac{3\alpha}{8m}.
\end{aligned}$$

The value of c is at most $\frac{1}{4}$ since $1 + \alpha \ln 2 < 2^\alpha$. □

Note that if $c = 0$ then by Corollary 18 the edge of the algorithm is $\alpha/(2m)$. Thus increasing c from 0 to $\frac{\alpha}{2^\alpha - 1}\frac{\ln 2}{4}$ reduced the edge of the algorithm from $\alpha/(2m)$ to $3\alpha/(8m)$.

We now define our final algorithm, Algorithm $\hat{R}^c_q$.

Definition 21 Algorithm $\hat{R}^c_q$

Algorithm $\hat{R}^c_q$ *is given a sample* $S = \text{sam}_f(\boldsymbol{x})$, *the additional instance* x, *and a random* $r \in [0,1]$, *as well as its two parameters* $c \in (0,1]$ *and* $q \in \mathbf{N}$. *Algorithm* $\hat{R}^c_q$ *also makes use of a weak consistency oracle for* $\mathcal{F}$.

Let $m = |S| + 1$. *Algorithm* $\hat{R}^c_q$ *splits* r *into a random position* t *in* $\{1, \ldots, m\}$, *an independent sequence of* $2q(m-t)$ *random bits, and an independent* $r' \in [0,1]$.

Algorithm $\hat{R}^c_q$ *then uses the* $2q(m-t)$ *random bits to generate* $2q$ *independent samples uniformly at random from* $\text{sam}_*(x_{t+1}, x_{t+2}, \ldots, x_{m-1})$. *Half of these samples are concatenated with the sample "*$\text{sam}_f(\boldsymbol{x}^{t-1}), (x,0)$*" to form* q *samples of length* m. *Each of these* m*-samples is fed to the weak consistency oracle. Let* $\hat{u}_0$ *be the fraction of them that are consistent with any function in* $\mathcal{F}$. *Similarly, the other half of the randomly generated samples are concatenated with* $\text{sam}_f(\boldsymbol{x}^{t-1}), (x,1)$ *and then fed to the weak consistency oracle. Let* $\hat{u}_1$ *be fraction of these samples that are consistent with any function in* $\mathcal{F}$.

Prediction Algorithm $\hat{R}^c_q$ *outputs the prediction* 0 *when*

$$r' \leq \frac{\lg\left(1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c}\right)}{\lg\left(1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c}\right) + \lg\left(1 + \frac{(1-c)\hat{u}_1 + c}{(1-c)\hat{u}_0 + c}\right)}$$

and 1 *otherwise.*

Theorem 22 *For all* $\boldsymbol{x} \in X^m$ *for which* $|\text{sam}_{\mathcal{F}}(\boldsymbol{x})| \leq 2^{m-\alpha}$, *if* $c = \frac{\alpha}{2^\alpha - 1}\frac{\ln 2}{4}$ *and* $q \geq 1066m^2\frac{(2^\alpha - 1)^2}{\alpha^4}\ln\frac{32m}{\alpha}$ *then the following holds for any* $f \in \mathcal{F}$:

$$\mathbf{E}_{\boldsymbol{y} \in U(\boldsymbol{x})}\left[\mathbf{M}_{\hat{R}^c_q,f}(\boldsymbol{y})\right] \leq \frac{1}{2} - \frac{\alpha}{8m}.$$

We first present some implications of Theorem 22 and then give its proof.

Corollary 23 *If for all* $\boldsymbol{x} \in X^m$, $|\text{sam}_{\mathcal{F}}(\boldsymbol{x})| \leq 2^{m-\alpha}$, *then for any* $f \in \mathcal{F}$

$$\mathbf{E}_{\boldsymbol{x} \in \mathcal{D}^m}\left[\mathbf{M}_{\hat{R}^c_q,f}(\boldsymbol{x})\right] \leq \frac{1}{2} - \frac{\alpha}{8m}$$

where c *and* q *are chosen as in Theorem 22.*

Corollary 24 *Let $\mathcal{F} = \bigcup_s \mathcal{F}_s$ and $X = \bigcup_n X_n$ be a prediction problem. If there is a sample size $m = p_1(n,s)$ such that for all n, s, $\boldsymbol{x} \in (X_n)^m$, $|\mathrm{sam}_{\mathcal{F}_s}(\boldsymbol{x})| \leq 2^{m-\alpha}$ where $\alpha = 1/p_2(m)$ and if there is a polynomial time weak consistency oracle for $\mathcal{F}$ then $\hat{R}_q^c$ leads to a weak learning algorithm.*

Proof: If $m = p_1(n,s)$ and $\alpha = 1/p_2(m)$ then $\hat{R}_q^c$ with c and q set as in Theorem 22 uses time (including calls to the weak consistency oracle) bounded by a polynomial in n and s. □

If $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ is learnable, then the VC-dimension of $\mathcal{F}_s$ on X_n is bounded by some $p(n,s)$ (where p is a polynomial) [BEHW89]. By Sauer's Lemma [Sau72], for any m and $\boldsymbol{x} \in (X_n)^m$ $|\mathrm{sam}_{\mathcal{F}_s}(\boldsymbol{x})| \leq \sum_{i=0}^{p(n,s)} \binom{m}{i}$. When $m = 2p(n,s)$ this sum[8] is 2^{m-1}. Thus $|\mathrm{sam}_{\mathcal{F}_s}(\boldsymbol{x})| \leq 2^{m-1}$ and we can apply Corollary 24.

Corollary 25 *If $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ is learnable and there is a polynomial consistency oracle for $\mathcal{F}$ then there is a computationally efficient weak learning algorithm for $\mathcal{F}$.*

Proof: (of Theorem 22) We begin by stating the obvious relationship between the mistake probabilities of Prediction Algorithm $\hat{R}_q^c$ and Prediction Algorithm R^c.

$$\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{\hat{R}_q^c,f}(\boldsymbol{y})\right] =$$
$$\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{R^c,f}(\boldsymbol{y})\right] + \mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{\hat{R}_q^c,f}(\boldsymbol{y}) - \mathbf{M}_{R^c,f}(\boldsymbol{y})\right]$$

Since $c = \frac{\alpha}{2^\alpha - 1}\frac{\ln 2}{4}$ it follows from Corollary 20 that

$$\mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{\hat{R}_q^c,f}(\boldsymbol{y})\right] \leq$$
$$\frac{1}{2} - \frac{3\alpha}{8m} + \mathbf{E}_{\boldsymbol{y}\in U(\boldsymbol{x})}\left[\mathbf{M}_{\hat{R}_q^c,f}(\boldsymbol{y}) - \mathbf{M}_{R^c,f}(\boldsymbol{y})\right]$$

Lemma 26 (below) completes the proof of the theorem by showing that

$$\mathbf{M}_{\hat{R}_q^c,f}(\boldsymbol{y}) - \mathbf{M}_{R^c,f}(\boldsymbol{y}) \leq \frac{\alpha}{4m},$$

for $c = \frac{\alpha}{2^\alpha - 1}\frac{\ln 2}{4}$ and $q \geq 1066m^2\frac{(2^\alpha-1)^2}{\alpha^4}\ln\frac{32m}{\alpha}$. □

Lemma 26 *For all $\boldsymbol{y} \in X^m$, $f \in \mathcal{F}$, $\alpha \in [0,m]$, and $q \geq 1066m^2\frac{(2^\alpha-1)^2}{\alpha^4}\ln\frac{32m}{\alpha}$,*

$$\mathbf{M}_{R^c,f}(\boldsymbol{y}) - \mathbf{M}_{R^c,f}(\boldsymbol{y}) \leq \frac{\alpha}{4m}$$

when $c = \frac{\alpha}{2^\alpha - 1}\frac{\ln 2}{4}$.

[8] In general, $\sum_{i=0}^{d}\binom{m}{i}$ is bounded by $m^d + 1$. When $m = 2p(n,s)$ we are summing half the binomial coefficients, so the sum is exactly 2^{m-1}.

Proof: Let $b = f(\boldsymbol{x}_m)$, then from the definition of $\hat{R}_q^c$ and R^c we can write:

$$\widehat{\mathrm{term}}_t = \frac{\lg\left(1 + \frac{(1-c)\hat{u}_{1-b}+c}{(1-c)\hat{u}_b+c}\right)}{\lg\left(1 + \frac{(1-c)\hat{u}_{1-b}+c}{(1-c)\hat{u}_b+c}\right) + \lg\left(1 + \frac{(1-c)\hat{u}_b+c}{(1-c)\hat{u}_{1-b}+c}\right)}$$

$$\mathrm{term}_t = \frac{\lg\left(1 + \frac{(1-c)u_{1-b}+c}{(1-c)u_b+c}\right)}{\lg\left(1 + \frac{(1-c)u_{1-b}+c}{(1-c)u_b+c}\right) + \lg\left(1 + \frac{(1-c)u_b+c}{(1-c)u_{1-b}+c}\right)}$$

and

$$\mathbf{M}_{\hat{R}_q^c,f}(\boldsymbol{y}) - \mathbf{M}_{R^c,f}(\boldsymbol{y}) = \frac{1}{m}\sum_{t=1}^{m}\left(\widehat{\mathrm{term}}_t - \mathrm{term}_t\right)$$

where $b = f(\boldsymbol{x}_m)$. This is a slight misuse of notation as the u_0, u_1, $\hat{u}_0$, and $\hat{u}_1$ all depend on f, $\boldsymbol{y}$, and t. In addition, $\hat{u}_0$ and $\hat{u}_1$ each depend on the q samples of length m selected by the algorithm's randomization. What we will show is that for any f, $\boldsymbol{y}$, and t the expectation of the difference inside the sum with respect to the randomization of $\hat{R}_q^c$ is at most $\frac{\alpha}{4m}$. We consider two cases.

Case 1: either $|\hat{u}_0 - u_0| > c\alpha/8m$ or $|\hat{u}_1 - u_1| > c\alpha/8m$. By Lemma 28 (applied with $\gamma = c\alpha/8m$) of the Appendix, this case happens with probability at most $\frac{\alpha}{8m}$. Note that the bound $q \geq 1066m^2\frac{(2^\alpha-1)^2}{\alpha^4}\ln\frac{32m}{\alpha}$ implies that

$$\begin{aligned} q &\geq \frac{\ln(32m/\alpha)}{2\gamma^2} \\ &= \frac{\ln(32m/\alpha)}{2}\frac{(8m)^2}{(c\alpha)^2} \\ &= \frac{\ln(32m/\alpha)}{2}\frac{64m^2(16)(2^\alpha-1)^2}{\alpha^4\ln^2 2} \end{aligned}$$

as required by Lemma 28.

Case 2: both $|\hat{u}_0 - u_0| \leq c\alpha/8m$ and $|\hat{u}_1 - u_1| \leq c\alpha/8m$. Applying Lemma 29 in the Appendix, for $b \in \{0,1\}$,

$$\lg\left(1 + \frac{(1-c)\hat{u}_{1-b}+c}{(1-c)\hat{u}_b+c}\right) - \lg\left(1 + \frac{(1-c)u_{1-b}+c}{(1-c)u_b+c}\right)$$

is at most $\alpha/(4m+\alpha)$. Thus by Lemma 30 (also in the Appendix), $\widehat{\mathrm{term}}_t - \mathrm{term}$ is at most $\alpha/8m$.

We now combine the cases by noting that $\widehat{\mathrm{term}}_t$ and term are both mistake probabilities, and so can never be greater than one. Thus if $a \leq \alpha/8m$ is the probability of Case 1, then the difference between the terms is at most

$$a \cdot 1 + (1-a)\frac{\alpha}{8m} \leq \frac{\alpha}{4m}.$$

□

The bound on the number of queries q in Theorem 22 is $O((2^\alpha - 1)^2/\alpha^4)$. This is exponential in m as α goes to m and is unbounded as α goes to zero.

Note that if $|\text{sam}_{\mathcal{F}_s}(\boldsymbol{x})| \leq 2^{m-\alpha}$ then $|\text{sam}_{\mathcal{F}_s}(\boldsymbol{x})| < 2^{m-\alpha'}$ for all $\alpha' < \alpha$. Thus we can use a smaller α as long as q remains polynomial in n and s. Clearly if m is polynomial in n and s, and $\alpha = 1/p_2(m)$ for some polynomial p_2, then q is polynomial in n and s as well. An alternative to simply using a smaller α is to embed the concept class $\mathcal{F}$ into some larger class $\mathcal{H}$ with $|\text{sam}_{\mathcal{H}_s}(\boldsymbol{x})| \leq 2^{m-(1/p_2(m))}$. There is an additional benefit to the embedding in that it may be easier to find a weak consistency oracle for $\mathcal{H}$ than for $\mathcal{F}$.

This leads us to the following definition of a "one-sided" consistency oracle:

Definition 27 *A* one-sided consistency oracle *for* $\mathcal{F} = \bigcup_s \mathcal{F}_s$ *on* $\bigcup_n X_n$ *is given* n, s, m, *and a sample* $S \in \text{sam}_*(\boldsymbol{x}^m)$ *where* $\boldsymbol{x}^m \in (X_n)^m$. *The oracle must answers "yes" if* $S = \text{sam}_f(\boldsymbol{x}^m)$ *for some* $f \in \mathcal{F}_s$ *and may answer either "yes" or "no" otherwise. However the total number of "yes" answers on the* 2^m *examples of* $S \in \text{sam}_*(\boldsymbol{x}^m)$ *can not exceed* $2^{m-1/p_2(m)}$, *for some polynomial* p_2. *Such an oracle is called* polynomial *if its answers are computed in time polynomial in* n, s, *and* m.

Clearly the existence of a polynomial one-sided consistency oracle for $\mathcal{F}$ implies polynomial weak learnability. Such an oracle might be much easier to find than a weak consistency oracle for $\mathcal{F}$. Even weaker oracles will be explored in future research.

Acknowledgements

We are grateful to Lenny Pitt, Peter Frankle, David Haussler, and Phil Long for valuable discussions and Hikoe Enomoto for improving the bound on size of the hypotheses class in the example following Theorem 3 by one.

David P. Helmbold was support by NSF grant CCR-9102635. Manfred K. Warmuth was support by ONR grants N00014-86-K-0454 and N00014-91-J-1162 and part of this work was done while he was visiting the IIAS-SIS Institute of Fujitsu Laboratories, Numazu, Japan.

References

[AHW87] N. Alon, D. Haussler, and E. Welzl. Partitioning and geometric embedding of range spaces of finite Vapnik-Chervonenkis dimension. In *Proceedings of Third Symposium on Computational Geometry*, pages 331–340, June 1987.

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

[BEHW89] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, 1989.

[Bon72] J. A. Bondy. Induced subsets. *Journal of Combinatorial Theory*, 12(B):201–202, 1972.

[Fre90] Y. Freund. Boosting a weak learning algorithm by majority. In *Proceedings of the 1990 Workshop on Computational Learning Theory*, pages 202–231, Morgan Kaufmann, San Mateo, CA, August 1990.

[HKLW91] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. *Information and Computation*, 95(2):129–161, December 1991.

[HKS91] D. Haussler, M. Kearns, and R. Schapire. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. In *Proceedings of the 1991 Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, CA, August 1991. To appear in *Machine Learning*.

[HLW] D. Haussler, N. Littlestone, and M. K. Warmuth. Predicting {0,1} functions on randomly drawn points. To appear in *Information and Computation*. An extended abstract appeared in COLT 88.

[HW92] D. Helmbold and M. K. Warmuth. On weak learning. In *Proceedings of the Third NEC Research Symposium on Computational Learning and Cognition*, SIAM, 3600 University City Science Center, Philadelphia, PA 19104-2688, May 1992.

[KV89] M. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 433–444, ACM, New York, May 1989. To appear in *Journal of the ACM*.

[Sau72] N. Sauer. On the density of families of sets. *Journal of Combinatorial Theory (Series A)*, 13:145–147, 1972.

[Sch90] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[VC71] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Application*, 16(2):264–280, 1971.

[Vov90] V. Vovk. Aggregating strategies. In *Proceedings of the 1990 Workshop on Computational Learning Theory*, pages 371–383, Morgan Kaufmann, San Mateo, CA, August 1990.

A Appendix

Lemma 28 *If* $q \geq \frac{\ln \frac{32m}{\alpha}}{2\gamma^2}$ *then for Prediction Algorithm* $\hat{R}_q^c$

$$\mathbf{Pr}_{r\in U_{[0,1]}}\left[|\hat{u}_0 - u_0| > \gamma \text{ or } |\hat{u}_1 - u_1| > \gamma\right] \leq \frac{\alpha}{8m}.$$

Proof: By symmetry, it suffices to show $\mathbf{Pr}_{r\in U_{[0,1]}}\left[|\hat{u}_0 - u_0| > \gamma\right] \leq \frac{\alpha}{16m}$. Since q queries are used to estimate n_0, Hoeffding's inequality shows

$$\mathbf{Pr}_{r\in U_{[0,1]}}\left[|\hat{u}_0 - u_0| > \gamma\right] \leq 2e^{-2q\gamma^2}.$$

Using the lower bound on q we see that this probability is at most $\frac{\alpha}{16m}$ as desired. □

Lemma 29 *If* $0 \leq u_0, \hat{u}_0, u_1, \hat{u}_1 \leq 1$, $0 < \gamma \leq \frac{1}{8}\frac{c\alpha}{m}$, $|\hat{u}_0 - u_0| \leq \gamma$, $|\hat{u}_1 - u_1| \leq \gamma$, $0 < c \leq 1$, *and* $\alpha \in (0, m]$ *then both*

$$\left|\lg(1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c}) - \lg(1 + \frac{(1-c)u_0 + c}{(1-c)u_1 + c})\right|$$

and

$$\left|\lg(1 + \frac{(1-c)\hat{u}_1 + c}{(1-c)\hat{u}_0 + c}) - \lg(1 + \frac{(1-c)u_1 + c}{(1-c)u_0 + c})\right|$$

are at most $\frac{\alpha}{4m+\alpha}$.

Proof: By symmetry it suffices to show the first inequality. First observe that

$$\begin{aligned}
&1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c} \\
&\leq 1 + \frac{(1-c)u_0 + c + (1-c)\gamma}{(1-c)u_1 + c - (1-c)\gamma} \\
&= \frac{(1-c)(u_0+u_1) + 2c}{(1-c)u_1 + c - (1-c)\gamma} \\
&\leq \frac{(1-c)(u_0+u_1) + 2c}{(1-c)u_1 + c} \frac{1}{1 - \gamma(1-c)/c} \\
&= \left(1 + \frac{(1-c)u_0 + c}{(1-c)u_1 + c}\right) \frac{1}{1 - \gamma(1-c)/c} \qquad (3)
\end{aligned}$$

Similarly,

$$\begin{aligned}
&1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c} \\
&\geq 1 + \frac{(1-c)u_0 + c - (1-c)\gamma}{(1-c)u_1 + c + (1-c)\gamma} \\
&= \frac{(1-c)(u_0+u_1) + 2c}{(1-c)u_1 + c + (1-c)\gamma} \\
&\geq \frac{(1-c)(u_0+u_1) + 2c}{(1-c)u_1 + c} \frac{1}{1 + \gamma(1-c)/c} \\
&= \left(1 + \frac{(1-c)u_0 + c}{(1-c)u_1 + c}\right) \frac{1}{1 + \gamma(1-c)/c} \qquad (4)
\end{aligned}$$

Combining (3) and (4), dividing by $1 + \frac{(1-c)u_0+c}{(1-c)u_1+c}$ and taking the lg of all sides gives us:

$$\begin{aligned}
&-\lg\left(1 + \gamma\frac{1-c}{c}\right) \\
&\leq \lg(1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c}) - \lg(1 + \frac{(1-c)u_0 + c}{(1-c)u_1 + c}) \\
&\leq -\lg\left(1 - \gamma\frac{1-c}{c}\right).
\end{aligned}$$

Since $0 \leq \lg\left(1 + \gamma\frac{1-c}{c}\right) < -\lg\left(1 - \gamma\frac{1-c}{c}\right)$, we have

$$\begin{aligned}
&\left|\lg(1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c}) - \lg(1 + \frac{(1-c)u_0 + c}{(1-c)u_1 + c})\right| \\
&\qquad \leq -\lg\left(1 - \gamma\frac{1-c}{c}\right) \\
&\qquad = -\lg\left(1 - \frac{\gamma}{c} + \gamma\right).
\end{aligned}$$

Using the facts that $\gamma \leq \frac{c\alpha}{8m}$ and $-\lg(1 - \frac{1}{8}x) \leq -x\lg\frac{7}{8}$ for $x \in [0,1]$ we obtain

$$\begin{aligned}
&\left|\lg(1 + \frac{(1-c)\hat{u}_0 + c}{(1-c)\hat{u}_1 + c}) - \lg(1 + \frac{(1-c)u_0 + c}{(1-c)u_1 + c})\right| \\
&\qquad \leq -\lg\left(1 - \frac{1}{8}\frac{\alpha}{m}\right) \\
&\qquad \leq -\frac{\alpha}{m}\lg\frac{7}{8} \\
&\qquad \leq -5\frac{\alpha}{5m}\lg\frac{7}{8} \\
&\qquad \leq \frac{\alpha}{4m+\alpha} 5\lg\frac{8}{7} \\
&\qquad \leq \frac{\alpha}{4m+\alpha}.
\end{aligned}$$

□

Lemma 30 *For any non-negative numbers* a *and* b *such that* $a + b \geq 2$, *and any* δ_0 *and* δ_1 *whose absolute values are both at most* $\delta = \frac{\alpha}{4m+\alpha} < \frac{1}{2}$,

$$\left|\frac{a+\delta_0}{a+b+\delta_0+\delta_1} - \frac{a}{a+b}\right| \leq \frac{\alpha}{8m}.$$

Proof:

$$\begin{aligned}
&\left|\frac{a+\delta_0}{a+b+\delta_0+\delta_1} - \frac{a}{a+b}\right| \\
&= \left|\frac{a(a+b) + \delta_0(a+b) - a(a+b) - a(\delta_0+\delta_1)}{(a+b+\delta_0+\delta_1)(a+b)}\right| \\
&= \left|\frac{\delta_0 b - \delta_1 a}{(a+b+\delta_0+\delta_1)(a+b)}\right| \\
&\leq \left|\frac{(a+b)\delta}{(a+b-2\delta)(a+b)}\right| \\
&\leq \frac{\delta}{2-2\delta} \\
&= \frac{\alpha}{8m}.
\end{aligned}$$

□