On Weak Learning

David P. Helmbold*and Manfred K. Warmuth†

UCSC-CRL-92-54
December 16, 1992

Board of Studies in Computer and Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064

ABSTRACT

An algorithm is a weak learning algorithm if with some small probability it outputs a hypothesis with error slightly below 50%. This paper presents relationships between weak learning, weak prediction (where the probability of being correct is slightly larger than 50%), and consistency oracles (which decide whether or not a given set of examples is consistent with a concept in the class). Our main result is a simple polynomial prediction algorithm which makes only a single query to a consistency oracle and whose predictions have a polynomial edge over random guessing. We compare this prediction algorithm with several of the standard prediction techniques, deriving an improved worst case bound on Gibbs Algorithm in the process. We use our algorithm to show that a concept class is polynomially learnable if and only if there is a polynomial probabilistic consistency oracle for the class. Since strong learning algorithms can be built from weak learning algorithms, our results also characterizes strong learnability.

---

*David P. Helmbold was supported by NSF grant CCR-9102635, email: dph@cis.ucsc.edu.

†Manfred K. Warmuth was supported by ONR grant N00014-91-J-1162 and part of this work was done while he was visiting the IIAS-SIS Institute of Fujitsu Laboratories, Numazu, Japan, email: manfred@cis.ucsc.edu.

1 Introduction

This paper presents a several learning results, including necessary and sufficient conditions for weak learning. To help introduce the learning terminology 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 target DFA. The labeled bitstrings are called examples and each possible target DFA defines a concept consisting of all bitstrings which it accepts (or equivalently, the associated indicator function on bitstrings). 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 bound on the size of (number of states in) the unknown target DFA. After seeing a reasonable number of instances (bitstrings of length at most n) labeled by the hidden target DFA of size at most s, the learning algorithm outputs a hypothesis which is intended to approximate the set of bitstrings of length at most n that 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 n, s, $\epsilon > 0$ 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 *polynomial* 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.[1]

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

Surprisingly, it has been shown that any polynomial weak learning algorithm can be used to build a polynomial strong learning algorithm [Sch90, Fre90]. These constructions create many copies of the weak learning algorithm and each copy generates a hypothesis based on a filtered sequence of examples. These hypotheses are then combined to form a master hypothesis. Thus to determine whether a concept class is polynomially learnable it suffices to construct polynomial weak learning algorithms. In this paper we give necessary and sufficient conditions for polynomial weak learning.

An Occam algorithm returns any hypothesis from some hypothesis class $\mathcal{H}$ that is consistent with the examples seen. Thus the hypothesis class must be large enough to represent the way that each possible target labels the examples. The hypothesis class $\mathcal{H}$ is allowed to vary based on the parameters n and s, and the number of examples, m. When the hypothesis class is the same as the concept class, the Occam algorithm can be

[1] More precisely, a polynomial learning algorithm is allowed to take time polynomial in the total bit length of all received instances as well as the four parameters. If n is an upper bound on the bit length of instances then the two definitions are equivalent. They differ only when n measures some aspect of the instances other than their bit length.

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

viewed as an oracle which returns a concept consistent with the set of examples. Previously it has been shown that if the cardinality of $\mathcal{H}$ is small (bounded by $p(n,s)\,m^{1-\kappa}$, where κ is a constant less than one), then the Occam algorithm is a strong learning algorithm [BEHW87]. We show that even if the size of the hypotheses class grows exponentially, Occam algorithms may already be weak learners. More precisely, our first result shows that when the cardinality of $\mathcal{H}$ is moderately sized, bounded[3] by $2^{m-1/p(m)}$, then the Occam algorithm is a weak learning algorithm. In contrast, we show that one Occam algorithm for a particular concept class that uses a slightly larger hypothesis class of size $2^{m+1}-2$ is not a weak learning algorithm.

A consistency oracle for a concept class $\mathcal{F}$ is given parameters n, s, and a sequence of labeled examples whose instances are words of length at most n. The consistency oracle determines whether or not there is a concept of size at most 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 a concept of size at most s consistent with the examples.

A probabilistic consistency oracle must answer "yes" with probability at least 50% (over its internal randomization) when given a set of examples consistent with a concept in the class, and must always answer "no" on some sets of examples which are not consistent with any concept in the class. The asymmetry in this definition seems to be necessary as shown by the counterexample and discussion in Section 10. We show that any polynomial time weak learning algorithm for $\mathcal{F}$ can be converted into a polynomial probabilistic consistency oracle of $\mathcal{F}$.

We also show that if a polynomial time probabilistic consistency oracle is available then it can be used to construct a polynomial weak learning algorithm for $\mathcal{F}$ whenever $\mathcal{F}$ is learnable at all with respect to an arbitrary distribution (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 consistency oracles was given by Haussler, Littlestone, and Warmuth [HLW]. However that algorithm is only polynomial if the VC dimension of $\mathcal{F}$ is a constant independent of n and s. Thus a class is polynomially learnable if and only if it has a polynomial probabilistic consistency oracle.

The following section contains an introduction to our notation. We define weak prediction algorithms in Section 3 and relate them to weak learning algorithms. Section 4 shows that one kind of weak learning algorithm is a "weak Occam algorithm" (similar to the "strong" Occam algorithms studied by Blumer et al. [BEHW87], Board and Pitt [BP92]) and Haussler[4] et al. [HKLW91]. The next part of the paper concentrates on prediction algorithms. In Section 5 we define lookahead prediction algorithms which get the entire set of instances where predictions will be required before making any predictions. There we show how to transform any lookahead prediction algorithm with a good total mistake bound into a (normal) prediction algorithm which has a small probability of making a mistake on the last trial. Section 6 presents and analyzes the Query Lookahead Algorithm, a generic lookahead prediction algorithm which uses a single query to a consistency oracle. Section 7 shows that a polynomial weak prediction algorithm is created when the Query Lookahead Algorithm uses a polynomial time consistency oracle and is transformed as described in Section 5. Furthermore, the resulting polynomial weak prediction algorithm makes only a

[3] An equivalent condition is to bound the cardinality of $\mathcal{H}$ by $2^m(1-1/p'(m))$, for some polynomial p'.

[4] The kind of Occam algorithms studied there are called random polynomial time hypothesis finders.

single query to the consistency oracle. In contrast, our earlier prediction algorithm [HW92b] requires a large number of consistency oracle queries to make each prediction. In Section 8 we show using our algorithm that sample size $2d - \Omega(\sqrt{d log d})$ suffices for weak learning (not necessarily polynomial weak learning) of concept classes of VC dimension d. In Goldman et al. [GKS90] it was shown that no algorithm can weakly learn some concept classes of VC dimension d from $d - O(log(d))$ examples. Section 9 compares our prediction algorithm with several of the standard prediction algorithms, as well as the weak prediction algorithm of [HW92b]. This comparison includes an improved bound on the expected total number of mistakes made by the Gibbs prediction algorithm [HKS91] when learning a worst-case concept. In Section 10 we define one-sided and probabilistic consistency oracles, and prove that a concept class is polynomially weakly learnable if and only if there is a polynomial probabilistic consistency oracle for the class. In Section 11 we introduce polynomial "data interpolators" and discuss how they generalize Weak Occam algorithms. We conclude in Section 12 by discussing a number of open problems raised by this research.

Preliminary versions of several results presented here have appeared in conference papers [HW92b, HW92a].

2 Notation

Throughout, lg and ln denote the binary and natural logarithms, respectively. When logarithms appear in asymptotic notation we use log, as the base is not relevant. We use $\mathbf{N}$ to denote the positive numbers, and adopt the convention that $\overline{0} = 1$ and $\overline{1} = 0$. Furthermore, if X is a set then X^* is the set of all finite sequences of elements from X (including the empty sequence) and X^+ is the set of all finite non-empty sequences of elements from X.

Let X be an arbitrary 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 $\mathbf{x}$ and $\mathbf{y}$ denote (finite) sequences of instances and $|\mathbf{x}|$ is the length of the sequence $\mathbf{x}$. For $1 \leq t \leq |\mathbf{x}|$, we use:

- x_t to denote the tth component of $\mathbf{x}$,
- $\mathbf{x}^{\leq t}$ to denote the t-vector $(x_1, x_2, \ldots, x_t)$,
- $\mathbf{x}^{<t}$ to denote the $t-1$ vector $(x_1, x_2, \ldots, x_{t-1})$, and
- $\mathbf{x}^{>t}$ to denote the $|\mathbf{x}| - t$ vector $(x_{t+1}, x_{t+2}, \ldots, x_{|\mathbf{x}|})$.

All of $\mathbf{x}^{\leq 0}$, $\mathbf{x}^{<1}$, and $\mathbf{x}^{>|\mathbf{x}|}$ denote the empty sequence Λ. We will often superscript sequences solely to emphasize their length.

Examples are instances labeled by either 0 or 1, i.e. elements of $X \times \{0, 1\}$. *Samples* are sequences of examples. The *sample of f on $\mathbf{x}$* is denoted by $\mathrm{sam}_f(\mathbf{x})$ and is the sequence of examples[5] $(x_1, f(x_1)), \ldots, (x_{|\mathbf{x}|}, f(x_{|\mathbf{x}|}))$.

We define $\mathrm{sam}_{\mathcal{F}}(\mathbf{x}) = \{\mathrm{sam}_f(\mathbf{x}) : f \in \mathcal{F}\}$. We also use $\mathrm{sam}_*(\mathbf{x})$ to denote the set of $2^{|\mathbf{x}|}$ samples where the first example contains x_1, and second example contains x_2, and so on. If $\mathrm{sam}_{\mathcal{F}}(\mathbf{x}) = \mathrm{sam}_*(\mathbf{x})$ then $\mathbf{x}$ is *shattered* by $\mathcal{F}$.

The *Vapnik-Chervonenkis dimension*, or *VC dimension*, of a concept class $\mathcal{F}$ on X is the largest k such that there exists an $\mathbf{x} \in X^k$ that is shattered by $\mathcal{F}$ [VC71, BEHW89].

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

If e is an example and S is a sample of length m then $\langle S, e\rangle$ 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 $\mathbf{x}$ (the sequence of instances in the sample) such that $S = \mathrm{sam}_f(\mathbf{x})$. Every $f \in \mathcal{F}$ is consistent with the empty sample.

We use $\mathbf{E}_{a\in\mathcal{P}}[z(a)]$ to denote the expectation of the random variable z under distribution $\mathcal{P}$, and $\mathbf{Pr}_{a\in\mathcal{P}}[\mathrm{condition}(a)]$ to denote the probability under the distribution $\mathcal{P}$ of the set containing all a satisfying the condition. We adopt the usual assumption that any probability used in this paper is measurable.

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 $U(\mathbf{x})$ is the uniform distribution on the $|\mathbf{x}|!$ permutations of sequence $\mathbf{x}$.

We will also make frequent use of the following bounds on the function 2^x.

Fact 2.1: *For any* $\alpha \in \mathbf{R}$, $1 - \alpha\ln 2 \le 2^{-\alpha}$ *and if* $\alpha \in [0,1]$ *then* $2^{-\alpha} \le 1 - \frac{\alpha}{2}$.

3 Weak Learning Models

A (randomized) *prediction algorithm*[6] A takes a sample, an instance, and a random number[7] 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 target 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 is intended to approximate f. Class $\mathcal{H}$ is called the *hypothesis class*. Let $A(\mathrm{sam}_f(\mathbf{x}^{<m}), r)$ denote (the representation of) the hypothesis output by algorithm A when run on the example sequence $\mathrm{sam}_f(\mathbf{x}^{<m})$ with randomization r. Each learning algorithm has an associated (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 the pair $(\mathrm{sam}_f(\mathbf{x}), r)$ 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 between a learning algorithm's hypothesis h and target concept f with respect to distribution $\mathcal{D}$ on X is denoted $\mathrm{Err}_{\mathcal{D}}(f,h)$. Formally, $\mathrm{Err}_{\mathcal{D}}(f,h) = \mathbf{Pr}_{x\in\mathcal{D}}[f(x) \neq h(x)]$.

---

[6]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 (see Definition 9.1).

[7]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 polynomial algorithms the number of random bits required must be polynomially bounded.

For any prediction algorithm A and concept $f \in \mathcal{F}$, we define $\mathbf{M}(A, f, \mathbf{x})$ as the probability that A makes a mistake on the last instance of $\mathbf{x}$ when learning f. More precisely, when $\mathbf{x}$ is a sequence of m instances,

$$\mathbf{M}(A, f, \mathbf{x}) = \mathbf{E}_{r \in U_{[0,1]}} \left[A(\mathrm{sam}_f(\mathbf{x}^{<m}), x_m, r) \neq f(x_m) \right],$$

where $U_{[0,1]}$ is the uniform distribution on $[0, 1]$. Thus the expected total number of mistakes made by A on a sequence $\mathbf{x}$ of m instances labeled by target f is $\sum_{t=1}^{m} \mathbf{M}(A, f, \mathbf{x}^{\leq t})$.

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.

$$A'(\mathrm{sam}_f(\mathbf{x}^{<m}), r)(x_m) \stackrel{\mathrm{def}}{=} A(\mathrm{sam}_f(\mathbf{x}^{<m}), x_m, r).$$

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 $\mathrm{sam}_f(\mathbf{x}^{<m})$, x_m, and r first uses learning algorithm A' to produce a hypothesis $h = A'(\mathrm{sam}_f(\mathbf{x}^{<m}), r)$ and then predicts with the value $h(x_m)$. In this case (see [HKLW91]), with $\mathbf{x}^{<m}$ and f fixed, $\mathbf{E}_{x_m \in \mathcal{D}} \left[\mathbf{M}(A, f, \mathbf{x}^{\leq m}) \right] = \mathbf{E}_{r \in [0,1]} \left[\mathrm{Err}_{\mathcal{D}}(f, h) \right]$. The same relationship holds when learning algorithm A' uses the prediction algorithm A as its hypothesis evaluator.

Note that probabilistic learning and prediction algorithms can be easily converted into deterministic learning and prediction algorithms by extracting random bits from additional examples ([HKLW91], Lemma 3.5). We use randomized learning and prediction algorithms for our basic models as our algorithms are naturally randomized.

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[8] and X_n contains all instances of length at most n. For the example in the introduction, X_n consists of all bitstrings of length at most n and $\mathcal{F}_s$ contains 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.

We extend the $\mathbf{M}(A, f, \mathbf{x})$ notation to handle these parameterized learning problems by defining

$$\mathbf{M}_{n,s}(A, f, \mathbf{x}) \stackrel{\mathrm{def}}{=} \mathbf{E}_{r \in U_{[0,1]}} \left[A(\mathrm{sam}_f(\mathbf{x}^{<m}), x_m, r, n, s) \neq f(x_m) \right].$$

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 such that 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 $U_{[0,1]}$ together with a sample $\mathrm{sam}_f(\mathbf{x})$, where $\mathbf{x}$ is drawn according to D^m and $m = p_1(n, s)$, the algorithm outputs a hypothesis $h = A[\mathrm{sam}_f(\mathbf{x}), r]$ on X_n for which

[8] The parameter n need not be the bit length of the instances. Other measures of instance complexity are allowed.

$$\mathbf{Pr}_{\mathbf{x}\in\mathcal{D}^m, r\in U_{[0,1]}}\left[\mathrm{Err}_{\mathcal{D}}(f,h) > \frac{1}{2} - \frac{1}{p_2(m)}\right] \leq 1 - \frac{1}{p_3(m)} \tag{3.1}$$

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

For a *strong learning algorithm* [Val84, BEHW89], Inequality (3.1) is replaced by the inequality

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

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 $1/\epsilon$ and $1/\delta$, as well as n and s.

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 bounded by a polynomial in the parameters n and s of the learning algorithm and the bit length of x. A *polynomial* weak (strong) learning algorithm must output polynomially evaluatable hypotheses and the total running time of the weak learning algorithm must be polynomial in the total length of its input,[9] n, and s (or in the total length of its input, 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 polynomial then the resulting strong learning algorithm is also polynomial.

Note that one can not convert weak learning algorithms into strong learning algorithms by simply increasing the sample size m. In fact, the error bound of $\frac{1}{2} - \frac{1}{p_2(m)}$ on the hypotheses produced by a weak learning algorithm can approach $\frac{1}{2}$ as m increases. The conversion algorithms of [Sch90] and [Fre90] repeatedly use the weak learning algorithm on different "small" samples of size $m = p_1(n,s)$ (where $p_1(n,s)$ is the first polynomial in the weak learning algorithm definition). These "small" samples are created by cleverly filtering the distribution and the resulting hypotheses are combined using the majority function.

As discussed above, prediction algorithms are closely related to learning algorithms. Intuitively, a weak prediction algorithm must make predictions that are slightly better than random guessing when given a polynomially sized sample. This is made precise in the following definition.

Definition 3.1 (Weak Prediction Algorithm): *Prediction algorithm A is a* weak prediction algorithm *for a concept class $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ if there are polynomials p_1 and p_2 such that when $m = p_1(n,s)$ then for all concepts $f \in \mathcal{F}_s$ and all distributions $\mathcal{D}$ on X_n,*

$$\mathbf{E}_{\mathbf{x}\in\mathcal{D}^m}\left[\mathbf{M}_{n,s}(A,f,\mathbf{x})\right] \leq \frac{1}{2} - \frac{1}{p_2(m)}. \tag{3.2}$$

Furthermore, if there is a polynomial p_3 such that the predictions made by A are computed in time bounded by $p_3(n,s,l)$ where l is the total bit length of the input, then A is a polynomial *weak prediction algorithm.*

[9] We allow the algorithm to use the total bit length of its input in its running time bound since the parameterization of the domain need not be based on the bit lengths of the instances.

Weak prediction algorithms perform well enough to be used as the hypothesis evaluators for trivial weak learning algorithms. This follows from the following lemma (which is proven using Markov's Lemma) applied with $\alpha = 1/p_2(m)$. The lemma states that if the prediction algorithm's expected error is $\frac{1}{2} - \alpha$, then the error of the trivial learning algorithm's hypothesis is at most $\frac{1}{2} - \frac{\alpha}{2}$ with probability at least $1 - \frac{\alpha}{1-\alpha}$.

Lemma 3.2: *For any distribution $\mathcal{D}$ on X, any prediction algorithm A and any concept f on X, if*

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

and A' is the trivial learning algorithm which uses A as its hypothesis evaluator then

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

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

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

Markov's Lemma says that for any non-negative random variable R, any distribution D and $z > 0$, $\mathbf{Pr}_{a\in D}\left[R(a) \geq z\mathbf{E}_{b\in D}\left[R(b)\right]\right] \leq 1/z$. 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 a weak prediction algorithm can trivially be used to create a weak learning algorithm, the converse is not true. Inequality (3.2) is a stronger constraint on the prediction/learning algorithm than Inequality (3.1).

4 Weak Occam Algorithms

In this section we define a kind of learning algorithm called "weak Occam algorithms" and show that any weak Occam algorithm is also a weak learning algorithm. An "Occam algorithm" is a learning algorithm that outputs consistent hypothesis from a "small" hypothesis class [BEHW87]. Algorithm A is a *strong Occam algorithm* for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ if there exists a polynomial p and a constant $\kappa < 1$ such that the following holds for all $n, s \geq 1$, targets $f \in \mathcal{F}_s$, and $\mathbf{x} \in X_n^m$:

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

It has been shown [BEHW87] 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. The above definition of (strong) Occam algorithm is less restrictive than previous definitions as they require that the hypotheses produced by the Occam algorithm be in the concept class [BP92], or in a specified hypotheses class [HKLW91]. We require only that the hypothesis class be polynomially evaluatable.

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

Algorithm A is a *weak Occam algorithm* for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ if there exist polynomials p_1 and p_2 such that the following holds for all $n, s \geq 1$, targets $f \in \mathcal{F}_s$, and $\mathbf{x} \in X_n^m$ for $m = p_1(n, s)$:

> when given n, s, and the sample $\mathrm{sam}_f(\mathbf{x})$, 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 $2^m(1 - 1/p_2(m))$.

Recall that the hypotheses output by a weak Occam algorithm using sample size $m = p_1(n, s)$ are called polynomially evaluatable if there is an algorithm that when given n, s, the representation of a hypothesis $h \in \mathcal{H}_{n,s}$ and $x \in X_n$, the algorithm can decide in time polynomial in n and s and the total bitlength of its input whether $x \in h$.

A weak Occam algorithm is called *polynomial* if running time is polynomial in n and s and the total bitlength of its input and if its hypotheses are polynomially evaluatable.

Clearly weak Occam algorithms can use much larger hypothesis classes than strong Occam algorithms (exponential as opposed to sub-linear in m). Using Fact 2.1, an equivalent definition of weak Occam algorithm is obtained by requiring $|\mathcal{H}_{n,s}| \leq 2^{m-1/p_2(m)}$ instead of $|\mathcal{H}_{n,s}| \leq 2^m(1 - 1/p_2(m))$.

Lemma 4.1: *Let p be any polynomial, $\mathcal{D}$ be any distribution on X, sample size m be in N, target f be any concept on X, and $\mathcal{H}$ be any hypothesis class on X of cardinality at most $2^{m-1/p(m)}$. If*

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

then

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

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

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

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

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

To show $\mathbf{Pr}_{\mathbf{x}\in\mathcal{D}^m}[\mathrm{BAD}] \leq 1 - 1/p''(m)$, 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 $-\ln(1 - \frac{1}{p''(m)}) \leq \frac{1/p''(m)}{1-1/p''(m)} = \frac{1}{p''(m)-1}$, it suffices to show that

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

Recall that $p'(m) = 4mp(m)/\ln(2)$ and $p''(m) = 1 + 2p(m)/\ln(2)$. Therefore $2m/p'(m) = \frac{1}{2}\ln(2)/p(m)$ and $1/(p''(m) - 1) = \frac{1}{2}\ln(2)/p(m)$, verifying the last inequality. Although these choices suffice, there are other choices for the polynomials p' and p''. □

Theorem 4.2: *If Algorithm A is a weak Occam algorithm for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ then A is a weak learning algorithm for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$. If A is a polynomial weak Occam algorithm then A is a polynomial weak learning algorithm.*

Proof: The second part follows from the definitions and from the first part. Let p_1 and p_2 be the polynomials for the weak Occam algorithm A and $\mathcal{H}_{n,s}$ be its hypothesis class when the parameters are n and s. The proof applies Lemma 4.1 as follows: for any n and s, let $\mathcal{D}$ be any distribution on $X = X_n$, target f be any concept in $\mathcal{F}_s$, sample size $m = p_1(n, s)$, and $\mathcal{H} = \mathcal{H}_{n,s}$. When given a sample $\mathrm{sam}_f(\mathbf{x})$, where $\mathbf{x} \in X^m$, Algorithm A outputs a hypothesis from class $\mathcal{H}$, a class which has cardinality at most $2^m(1 - 1/p_2(m))$. By Fact 2.1 the latter is equivalent to $|\mathcal{H}| \leq 2^{m-1/\tilde{p}_2(m)}$, for some polynomial $\tilde{p}_2$. Denote the output hypothesis by $A[\mathrm{sam}_f(\mathbf{x})]$.

Define BAD as in the Lemma 4.1 with polynomial $p(m)$ set to $\tilde{p}_2(m)$. Then

$$\mathbf{Pr}_{\mathbf{x}\in\mathcal{D}^m}\left[\mathrm{Err}_{\mathcal{D}}(f, A[\mathrm{sam}_f(\mathbf{x})]) \geq \frac{1}{2} - \frac{\ln(2)}{4m\tilde{p}_2(m)}\right] \leq \mathbf{Pr}_{\mathbf{x}\in\mathcal{D}^m}[\mathrm{BAD}].$$

From Lemma 4.1, $\mathbf{Pr}_{\mathbf{x}\in\mathcal{D}^m}[\mathrm{BAD}]$ is at most $1 - 1/(1 + 2\tilde{p}_2(m)/\ln(2))$ and A is a 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] 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.

It is interesting to investigate when an Occam style algorithm must be a weak learning algorithm simply because of its sample size m (which is a function of n and s) and the size of its hypothesis class (which is a function of n, s and m). By our definition of weak

Occam algorithm and the proof of Theorem 4.2, sample size $p_1(n,s)$ and hypothesis class size $2^{m-1/p_2(n,s)}$ (where p_1 and p_2 are polynomials) always assure weak learning. Note that in this case the hypotheses can be encoded using $m - 1/p_2(n,s)$ bits, which is less than m, the number of binary labels in the examples. Thus, for each n and s, a weak Occam algorithm can be viewed as compressing samples of size $m = p_1(n,s)$ down to $m - 1/p_2(n,s)$ bits.

There are degenerate cases where sample size one and hypothesis class size two (i.e. "compressing" one label to one bit) does not lead to weak learning. Let the domain consist of two points and the concept class contain all four concepts on the two points (i.e. the VC dimension of the concept class is two). One Occam-style algorithm uses the hypothesis class consisting of the all-zero and the all-one concept. After seeing a single example, the algorithm returns whichever hypothesis is consistent with that example. If the target concept is one of the concepts not in the hypothesis class and the distribution on the domain is the uniform distribution, then the error of the produced hypothesis is always exactly half, and this Occam-style algorithm is not a weak learning algorithm.

We now present a second Occam-style algorithm which is not a weak learning algorithm. This algorithm outputs, from samples of size $m = s$, consistent hypotheses from a class of size $2^{m+1} - 2$ which have error exactly 50%. Let $X_n = \{0,1\}^n$ and $\mathcal{F}_s = \{f_{\mathbf{v},b} : \mathbf{v} \in \{0,1\}^s, b \in \{0,1\}\}$, where $f_{\mathbf{v},b}(\mathbf{x}) \equiv \mathbf{v} \cdot \mathbf{x} - b \bmod 2$ for any $\mathbf{x} \in \{0,1\}^s$ and 0 otherwise. Thus the learning problem is only interesting when $n = s$. Let $\mathbf{0}$ denote the all zero vector, and $\mathbf{1}$ the all one vector. The concepts $f_{\mathbf{0},0}$ and $f_{\mathbf{0},1}$ of $\mathcal{F}_s$ label all of $\{0,1\}^s$ with 0 and 1, respectively. Call those two concepts of $\mathcal{F}_s$ the *trivial concepts* and the remaining concepts of $\mathcal{F}_s$ the non-trivial concepts.

Consider the Occam-style algorithm that when $n \neq s$ outputs the trivial concept $f_{\mathbf{0},0}$ and if $n = s$, it sees $m = n = s$ examples and forms its hypothesis as follows. If any of the m examples are labeled with 1 then it outputs any non-trivial concept in $\mathcal{F}_m$ that is consistent with the sample. If all m examples are labeled with 0 then the Occam-style algorithm forms a matrix M from the examples (each example becomes a row of M). If M is singular then it outputs a hypothesis $f_{\mathbf{v},0}$ such that $M\mathbf{v} \equiv \mathbf{0}$ and $\mathbf{v} \neq \mathbf{0}$. If M is non-singular it outputs the unique hypothesis $f_{\mathbf{v},1}$ s.t. $M\mathbf{v} \equiv \mathbf{1}$. Again $\mathbf{v} \neq \mathbf{0}$.

Note that in all cases the hypothesis output by this algorithm is consistent with the sample. If $n = s = m$, then the hypothesis is a nontrivial concept of $\mathcal{F}_m$. Thus the hypothesis class used by the algorithm has size $2^{m+1} - 2$. When the target concept is the trivial concept $f_{\mathbf{0},0} \in \mathcal{F}_m$ 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 target $f_{\mathbf{0},0}$ and the uniform distribution on $\{0,1\}^m \subset X_n$. We conclude that the above Occam-style algorithm which uses a hypothesis class of size $2^{m+1} - 2$ when given samples of size m is not a weak learning algorithm. (Note that $|\mathcal{F}_m| = 2^{m+1}$ and the VC dimension[10] of $\mathcal{F}_m$ is $m + 1$, one larger than the number of examples.)

Intuitively, "compressing" samples of size m to m bits should not be sufficient to show weak learning. A more specific conjecture is given in the final section of this paper.

Note these results have certain negative implications. Since DFAs over a binary alphabet are not polynomially learnable under certain cryptographic assumptions [KV89, Kha92], there can't exist a polynomial weak Occam algorithm for this class. Thus, given the same cryptographic assumptions, if there exists a polynomial algorithm that, on inputs

[10] The $m + 1$ bitvectors of $\{0,1\}^m$ with at most one 1 are shattered by $\mathcal{F}_m$.

of $m = p(n,s)$ many bitstrings of length at most n labelled by a DFA of at most s states, outputs a consistent hypothesis from a polynomially evaluatable class $\mathcal{H}_{n,s}$, then the fraction $1/(m - lg|\mathcal{H}_{n,s}|)$ is not polynomial.

5 Lookahead Prediction

The last section has analyzed weak Occam learning algorithms. In the next several sections we develop and analyze a weak prediction algorithm. The presentation of this and the next section is simplified by omitting the parameters s and n on the concept class and instance space. We will return to the parameterized case when considering the running time of our algorithm in Section 7.

Recall that a prediction algorithm A receives three inputs: a sample, the instance whose label is to be predicted, and a random number. Thus algorithm A can be viewed as a function, $A : (X \times \{0,1\})^* \times X \times [0,1] \to \{0,1\}$. One would expect that a prediction algorithm would be able to perform better if it knew ahead of time which instances it will be asked to predict on.

A *lookahead prediction algorithm*, L, receives a sequence of (unlabeled) instance as an additional input. Formally, $L : (X \times \{0,1\})^* \times X \times X^* \times [0,1] \to \{0,1\}$. The sequence of additional instances contains those instances where the algorithm will be asked for predictions in the future.

We now extend our $\mathbf{M}()$ notation for the probability of a mistake to handle lookahead prediction algorithms. Recall that for a prediction algorithm A, $\mathbf{M}(A, f, \mathbf{x})$ denotes the probability that algorithm A incorrectly predicts the label of $x_{|\mathbf{x}|}$ when given the sample $\mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|})$.

For a lookahead prediction algorithm L, we define $\mathbf{M}(L, f, \mathbf{x}, t)$ (for $1 \leq t \leq |\mathbf{x}|$) as the probability that L incorrectly predicts the label of x_t when the target concept is f and L is given the labels of each x_i for $1 < i < t$, instance x_t, and the additional instances x_j for $t < j \leq |\mathbf{x}|$. Formally,

$$\mathbf{M}(L, f, \mathbf{x}, t) \stackrel{\text{def}}{=} \mathbf{E}_{r \in U_{[0,1]}} \left[L(\mathrm{sam}_f(\mathbf{x}^{<t}), x_t, \mathbf{x}^{>t}, r) \neq f(x_t) \right].$$

The natural use of lookahead algorithms is to predict on each of the instances of $\mathbf{x}$ in turn. For a given $\mathbf{x} \in X^m$ and hidden target $f \in \mathcal{F}$, the lookahead algorithm is used as follows.

for $t := 1$ to m do
  pick $r \in U_{[0,1]}$
  use $L(\mathrm{sam}_f(\mathbf{x}^{<t}), x_t, \mathbf{x}^{>t}, r)$ as the prediction of $f(x_t)$
  receive feedback $f(x_t)$

Note that when predicting the label of x_t, the lookahead algorithm is given only the labels of the previous instances.

Any lookahead prediction algorithm can be trivially used as a prediction algorithm by simply supplying it with the empty sequence of additional instances. Since our goal is a weak prediction algorithm, we want to minimize the mistake probability on the last instance of the sequence $\mathbf{x}$. If each instance in $\mathbf{x}$ is independently drawn from the same distribution (as in the definitions of weak learning and weak prediction), then all permutations of a set of instances are equally likely. Thus it suffices to bound the probability of a mistake on the last instance of a random permutation of $\mathbf{x}$ (this was used extensively in [HLW]).

Lemma 5.1: *Let X be any domain, $\mathcal{D}$ be any distribution on X, $m \in \mathbf{N}$, and R be a random variable on X^m. Then*

$$\mathbf{E}_{\mathbf{x}\in\mathcal{D}^m}\left[R(\mathbf{x})\right] = \mathbf{E}_{\mathbf{x}\in\mathcal{D}^m}\left[\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[R(\mathbf{x})\right]\right].$$

Lemma 9.2 shows that several good lookahead prediction algorithms have a mistake probability of $\frac{1}{2}$ on the last instance of a random permutation of some $\mathbf{x}$. Therefore the trivial use of lookahead prediction algorithms as predictors does not appear to yield weak prediction algorithms. We now present a more sophisticated way to construct prediction algorithms from lookahead algorithms.

Definition 5.2 (Lookahead Conversion, $\widetilde{L}$): *For each lookahead prediction algorithm L, the Lookahead Conversion of L is the prediction algorithm $\widetilde{L}$ described as follows:*

Input: *A sample* $\mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|})$ *for some* $\mathbf{x}^{<|\mathbf{x}|} \in X^*$ *and unknown* $f \in \mathcal{F}$, *instance* $x_{|\mathbf{x}|} \in X$, *and a random number* $r \in U_{[0,1]}$.

Computation: *Split r into a random t chosen uniformly from $\{1, \cdots, |\mathbf{x}|\}$ and an independent r' chosen from $U_{[0,1]}$. Call*

$$L(\mathrm{sam}_f(\mathbf{x}^{<t}), x_{|\mathbf{x}|}, \langle x_{t+1}, x_{t+2}, \cdots, x_{|\mathbf{x}|-1}, x_t\rangle, r').$$

Output: *The prediction returned by lookahead algorithm L.*

In other words, the Lookahead Conversion of L, Algorithm $\widetilde{L}$, creates a new $\mathbf{x}'$ by swapping $x_{|\mathbf{x}|}$ with a randomly chosen x_t and predicts as L does on the tth instance of this new $\mathbf{x}'$ (because of the swap, $x'_t = x_{|\mathbf{x}|}$).

The following theorem bounds the probability that the Lookahead Conversion incorrectly predicts the label of the last instance of a random permutation of $\mathbf{x}$ by $(1/|\mathbf{x}|)$ times the expected total number of mistakes made by the lookahead algorithm when it predicts on each instance of $\mathbf{x}$ in turn.

Theorem 5.3: *Let L be any lookahead prediction algorithm for concept class $\mathcal{F}$ on domain X, and $\widetilde{L}$ be the Lookahead Conversion of L. For any sequence of instances $\mathbf{x} \in X^+$ and target $f \in \mathcal{F}$,*

$$\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{L}, f, \mathbf{y})\right] = \frac{1}{|\mathbf{x}|}\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\sum_{t=1}^{|\mathbf{x}|}\mathbf{M}(L, f, \mathbf{y}, t)\right].$$

Proof: Recall that $U(\mathbf{x})$ is the uniform distribution over the permutations of $\mathbf{x}$. For any permutation σ of $(1, \cdots, |\mathbf{x}|)$ and sequence $\mathbf{y}$ of $|\mathbf{x}|$ instances, let $\sigma(\mathbf{y}) = \langle y_{\sigma(1)}, \cdots, y_{\sigma(|\mathbf{x}|)}\rangle$. Note that

$$\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(L, f, \mathbf{y}, t)\right] = \mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(L, f, \sigma(\mathbf{y}), t)\right].$$

Let σ_t be the permutation $\langle 1, \cdots, t-1, |\mathbf{x}|, t+1, \cdots, |\mathbf{x}|-1, t\rangle$. That is σ_t simply swaps $|\mathbf{x}|$ and t.

It follows from the definition of $\widetilde{L}$, that

$$\begin{aligned}
\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{L}, f, \mathbf{y})\right] &= \frac{1}{|\mathbf{x}|}\sum_{t=1}^{|\mathbf{x}|}\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(L, f, \sigma_t(\mathbf{y}), t)\right] \\
&= \frac{1}{|\mathbf{x}|}\sum_{t=1}^{|\mathbf{x}|}\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(L, f, \mathbf{y}, t)\right] \\
&= \frac{1}{|\mathbf{x}|}\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\sum_{t=1}^{|\mathbf{x}|}\mathbf{M}(L, f, \mathbf{y}, t)\right].
\end{aligned}$$

□

Theorem 5.3 shows that any good lookahead algorithm can be converted into a good prediction algorithm (when each permutation of the example sequence is equally likely). Lemma 5.1 can be combined with Theorem 5.3, giving us the following corollary.

Corollary 5.4: *Let L be any lookahead prediction algorithm for concept class $\mathcal{F}$ on domain X, $\mathcal{D}$ be a distribution on X, $m \in \mathbf{N}$, and $\widetilde{L}$ be the Lookahead Conversion of L. For any $f \in \mathcal{F}$:*

$$\mathbf{E}_{\mathbf{x}\in\mathcal{D}^m}\left[\mathbf{M}(\widetilde{L}, f, \mathbf{x})\right] = \frac{1}{m}\mathbf{E}_{\mathbf{x}\in\mathcal{D}^m}\left[\sum_{t=1}^{m}\mathbf{M}(L, f, \mathbf{x}, t)\right].$$

In the next section we present and analyze a surprisingly simple general purpose lookahead prediction algorithm.

6 The Query Lookahead Prediction Algorithm

This section presents a lookahead algorithm which makes a single query to a consistency oracle (defined below). In many situations the performance of this lookahead algorithm is good enough so that the transformation of the preceding section leads to a weak learning algorithm.

Definition 6.1 (Consistency Oracle): *A* consistency oracle *for $\mathcal{F}$ on X is given a sample*[11] *$S \in \mathrm{sam}_*(\mathbf{x})$ where $\mathbf{x} \in X^*$ and answers "yes" if $S \in \mathrm{sam}_{\mathcal{F}}(\mathbf{x})$ and "no" otherwise.*

Note that the consistency oracle gives a yes/no answer rather than returning an $f \in \mathcal{F}$.

Definition 6.2 (Query Lookahead Prediction Algorithm Q): *Let $\mathcal{F}$ be the concept class and X be the domain. The (one) Query Lookahead Prediction Algorithm Q works as follows:*

Input: *A sample $\mathrm{sam}_f(\mathbf{x}^{<t})$ for some $\mathbf{x}^{<t} \in X^{t-1}$ and unknown $f \in \mathcal{F}$, an instance $x_t \in X$, a sequence of future instances $\mathbf{x}^{>t}$, and a random number $r \in U_{[0,1]}$. Algorithm Q also uses a consistency oracle for $\mathcal{F}$ on X.*

Computation: *Extract $|\mathbf{x}| - t + 1$ independent random bits, $b_t, b_t + 1, \cdots, b_{|\mathbf{x}|}$, from r and query the oracle on the sample:*

$$\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, b_t), (x_{t+1}, b_{t+1}), \cdots, (x_{|\mathbf{x}|}, b_{|\mathbf{x}|})\rangle.$$

Output: *If the oracle answers "yes" then predict that the label of x_t is b_t. If the oracle answers "no" then predict that the label of x_t is $\overline{b_t}$.*

The Query Lookahead Prediction Algorithm is very simple. It randomly extends the sample to include the other instances. If the extended sample is consistent with some $f \in \mathcal{F}$ it predicts with the same label attached to the query instance in the extended sample. If no $f \in \mathcal{F}$ is consistent with the extended sample then the algorithm predicts with the opposite label.

Definition 6.3 (Quantity $N_{\mathcal{F}}(S, \mathbf{x})$): *For concept class $\mathcal{F}$ over X, $\mathbf{x} \in X^*$, $0 \le t \le |\mathbf{x}|$ and a sample $S \in \mathrm{sam}_*(\mathbf{x}^{\le t})$, let $\mathcal{F}' = \{f \in \mathcal{F} : f \text{ is consistent with } S\}$. We define "$N_{\mathcal{F}}(S, \mathbf{x})$" to be the number of samples in $\mathrm{sam}_{\mathcal{F}'}(\mathbf{x})$.*

[11] Recall that $\mathrm{sam}_{\mathcal{F}}(\mathbf{x})$ is the set of all $\mathrm{sam}_f(\mathbf{x})$ where $f \in \mathcal{F}$ and that $\mathrm{sam}_*(\mathbf{x})$ is the set of all $\mathrm{sam}_g(\mathbf{x})$ where g is one of the $2^{|\mathbf{x}|}$ ways of labeling $\mathbf{x}$.

In other words, $N_{\mathcal{F}}(S, \mathbf{x})$ is the number of ways functions in $\mathcal{F}$ can label $\mathbf{x}^{>t}$ while remaining consistent with the (sub-) sample S. Since every function is consistent with the empty sample Λ, we have $N_{\mathcal{F}}(\Lambda, \mathbf{x}) = |\mathrm{sam}_{\mathcal{F}}(\mathbf{x})|$. If S is a sample in $\mathrm{sam}_*(\mathbf{x}^{\leq t}) - \mathrm{sam}_{\mathcal{F}}(\mathbf{x}^{\leq t})$, then no function in $\mathcal{F}$ is consistent with S and $N_{\mathcal{F}}(S, \mathbf{x}) = 0$.

The single query done by the Query Lookahead Prediction Algorithm is attempting to determine which of $N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 0)\rangle, \mathbf{x})$ and $N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 1)\rangle, \mathbf{x})$ is larger. When bit $b_t = 1$, the probability of $Q(\mathrm{sam}_f(\mathbf{x}^{<t}), x_t, \mathbf{x}^{>t}, r)$ predicting 1 equals $N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 1)\rangle, \mathbf{x})/2^{|\mathbf{x}|-t}$. When $b_t = 0$, the probability that $Q(\mathrm{sam}_f(\mathbf{x}^{<t}), x_t, \mathbf{x}^{>t}, r)$ predicts 1 is $1 - (N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 0)\rangle, \mathbf{x})/2^{|\mathbf{x}|-t})$. Since b_t is equally likely to be either 0 or 1, the probability that $Q(\mathrm{sam}_f(\mathbf{x}^{<t}), x_t, \mathbf{x}^{>t}, r)$ predicts 1 is

$$\frac{1}{2} + \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 1)\rangle, \mathbf{x}) - N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 0)\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t+1}}. \tag{6.1}$$

Note that this probability lies in $[0, 1]$ as both values $N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 1)\rangle, \mathbf{x})$ and $N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, 0)\rangle, \mathbf{x})$ are between 0 and $2^{|\mathbf{x}|-t}$.

The following lemma bounds the probability that Algorithm Q predicts incorrectly on the tth instance.

Lemma 6.4: *For any class $\mathcal{F}$ on X, target $f \in \mathcal{F}$, instance sequence $\mathbf{x} \in X^+$, and $1 \leq t \leq |\mathbf{x}|$, if the Query Lookahead Prediction Algorithm Q uses a consistency oracle for $\mathcal{F}$ then*

$$\begin{aligned} \mathbf{M}(Q, f, \mathbf{x}, t) &= \frac{1}{2} + \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, \overline{f(x_t)})\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t+1}} \\ &\quad - \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, f(x_t))\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t+1}}. \end{aligned}$$

Proof: Using Equation 6.1, the probability that $Q(\mathrm{sam}_f(\mathbf{x}^{<t}), x_t, \mathbf{x}^{>t}, r)$ predicts $f(x_t)$ on x_t is

$$\frac{1}{2} + \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, f(x_t))\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t+1}} - \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, \overline{f(x_t)})\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t+1}}.$$

Thus the probability (and expectation) that algorithm Q makes a mistake by predicting $\overline{f(x_t)}$ is

$$\frac{1}{2} - \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, f(x_t))\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t+1}} + \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, \overline{f(x_t)})\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t+1}}$$

as claimed. □

We are now ready to present the main theorem of this section.

Theorem 6.5: *For class $\mathcal{F}$ on X, $f \in \mathcal{F}$, and $\mathbf{x} \in X^*$, if the Query Lookahead Prediction Algorithm Q uses a consistency oracle for $\mathcal{F}$, then*

$$\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(Q, f, \mathbf{x}, t) = \frac{|\mathbf{x}|}{2} + \frac{|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})|}{2^{|\mathbf{x}|}} - 1.$$

In particular, if $|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| \leq 2^{|\mathbf{x}|-\alpha}$, then $\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(Q, f, \mathbf{x}, t) \leq \frac{|\mathbf{x}|}{2} + 2^{-\alpha} - 1$.

Proof: The Theorem trivially holds when $\mathbf{x} = \Lambda$. Otherwise, for $1 \leq t \leq |\mathbf{x}|$ we define

$$u_t = \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, f(x_t))\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t}}$$

and

$$\overline{u_t} = \frac{N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, \overline{f(x_t)})\rangle, \mathbf{x})}{2^{|\mathbf{x}|-t}}.$$

Clearly $u_t = \frac{1}{2}(\overline{u_{t+1}} + u_{t+1})$ for $1 \leq t < |\mathbf{x}|$. Using this notation and Lemma 6.4,

$$\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(Q, f, \mathbf{x}, t) = \frac{|\mathbf{x}|}{2} + \frac{1}{2}\sum_{t=1}^{|\mathbf{x}|}(\overline{u_t} - u_t). \tag{6.2}$$

We first show by induction on m that

$$\sum_{t=1}^{m}(\overline{u_t} - u_t) = \sum_{t=1}^{m} \frac{\overline{u_t}}{2^{t-1}} + (\frac{1}{2^{m-1}} - 2)u_m \tag{6.3}$$

for each $1 \leq m \leq |\mathbf{x}|$. The base case $m = 1$ is trivial and if $1 < m \leq |\mathbf{x}|$ then by induction

$$\begin{aligned}
\sum_{t=1}^{m}(\overline{u_t} - u_t) &= \sum_{t=1}^{m-1} \frac{\overline{u_t}}{2^{t-1}} + (\frac{1}{2^{m-2}} - 2)u_{m-1} + \overline{u_m} - u_m \\
&= \sum_{t=1}^{m-1} \frac{\overline{u_t}}{2^{t-1}} + (\frac{1}{2^{m-2}} - 2)\frac{1}{2}(u_m + \overline{u_m}) + \overline{u_m} - u_m \\
&= \sum_{t=1}^{m} \frac{\overline{u_t}}{2^{t-1}} + (\frac{1}{2^{m-1}} - 2)u_m.
\end{aligned}$$

Using the definition of $\overline{u_t}$ we see that

$$\begin{aligned}
\sum_{t=1}^{|\mathbf{x}|} \frac{\overline{u_t}}{2^{t-1}} &= \frac{2}{2^{|\mathbf{x}|}}\sum_{t=1}^{|\mathbf{x}|} N_{\mathcal{F}}(\langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_t, \overline{f(x_t)})\rangle, \mathbf{x}) \\
&= \frac{2}{2^{|\mathbf{x}|}}(|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| - 1)
\end{aligned}$$

as the sum counts every sample in $\mathrm{sam}_{\mathcal{F}}(\mathbf{x})$ except for $\mathrm{sam}_f(\mathbf{x})$. Plugging this and the fact that $u_{|\mathbf{x}|} = 1$ into Equation 6.3 shows that

$$\begin{aligned}
\sum_{t=1}^{|\mathbf{x}|}(\overline{u_t} - u_t) &= \frac{2(|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| - 1)}{2^{|\mathbf{x}|}} + (\frac{1}{2^{|\mathbf{x}|-1}} - 2) \\
&= \frac{2\,|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})|}{2^{|\mathbf{x}|}} - 2.
\end{aligned}$$

Combining this with Equation 6.2 gives the first equality of the theorem. □

Corollary 6.6: *Let $\mathcal{F}$ be a concept class on X, target f be in $\mathcal{F}$, and $\mathbf{x} \in X^+$. If $|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| \leq 2^{|\mathbf{x}|-\alpha}$, the Query Lookahead Prediction Algorithm Q uses a consistency oracle for $\mathcal{F}$ and Prediction Algorithm $\widetilde{Q}$ is the Lookahead Conversion of Q then*

$$\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{Q}, f, \mathbf{y})\right] = \frac{1}{2} - \frac{1 - 2^{-\alpha}}{|\mathbf{x}|}.$$

For $\alpha > 0$ the above expectation is at least $\frac{1}{2} - \frac{\alpha \ln 2}{|\mathbf{x}|}$ and for $\alpha \in [0, 1]$ the expectation is at most $\frac{1}{2} - \frac{\alpha}{2|\mathbf{x}|}$.

Proof: The corollary follows from Theorem 6.5, Theorem 5.3, and Fact 2.1. □

Note that by Fact 2.1 $|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| \leq 2^{|\mathbf{x}|}(1 - ln(2)\alpha)$ implies $|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| \leq 2^{|\mathbf{x}|-\alpha}$.

7 Polynomially Efficient Weak Learning

We now return to the setting where $\mathcal{F} = \bigcup_s \mathcal{F}_s$ is a parameterized concept class on the parameterized domain $X = \bigcup_n X_n$. In this setting, algorithms and oracles are given n and s as additional parameters. We call an algorithm (or oracle) *polynomial* if its resource requirements[12] are bounded by a polynomial in n, s, and the total bit length of the instances in $\mathbf{x}$.

Definition 7.1 (Polynomial Consistency Oracle): *Algorithm $\mathcal{O}$ is a* polynomial consistency oracle *for $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ if $\mathcal{O}$ maps $\mathbf{N} \times \mathbf{N} \times (X \times \{0,1\})^*$ to $\{yes, no\}$, and there is a polynomial p such that for each $n, s \in \mathbf{N}$ and $S \in (X_n \times \{0,1\})^*$:*

1. *Algorithm $\mathcal{O}(n, s, S)$ answers "yes" if and only if S is consistent with some $f \in \mathcal{F}_s$, and*
2. *The computation time of $\mathcal{O}(n, s, S)$ is bounded by $p(n, s, l)$ where l is the total bit length of the instances in S.*

This definition requires that the polynomial consistency oracle be correct on samples of all lengths. In fact, our algorithm can get by with a weaker oracle. Calls to $\mathcal{O}(n, s, S)$ need answer correctly only when $|S| = 2p(n, s)$ for some polynomial $p(n, s)$ which is always at least the VC dimension of $\mathcal{F}_s$ on X_n. Even weaker oracles with "one-sided error" were considered in [HW92b] and are discussed in Section 10.

We now consider the Lookahead Transform, $\widetilde{Q}$, of the parameterized Query Lookahead Prediction Algorithm. When $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ is learnable at all and has a polynomial consistency oracle then Prediction Algorithm $\widetilde{Q}$ is a polynomial weak learning algorithm. The following is description of the parameterized Prediction Algorithm $\widetilde{Q}$.

Definition 7.2 (Prediction Algorithm $\widetilde{Q}$): *Prediction Algorithm $\widetilde{Q}$ for $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ works as follows:*

Input: *Parameters n and s; a sample $\mathrm{sam}_f(\mathbf{x}^{<m})$ for some $m \in \mathbf{N}$, unknown $f \in \mathcal{F}_s$, and $\mathbf{x}^{<m} \in (X_n)^{m-1}$; the instance $x_m \in X_n$; and a random number $r \in U_{[0,1]}$. The algorithm also uses a polynomial consistency oracle $\mathcal{O}$ for $\mathcal{F}$ on X.*

Computation: *Extract from r a random t chosen uniformly from $\{1, \ldots, m\}$ and an additional $m - t + 1$ random bits, $b_t, b_{t+1}, \cdots, b_m$, each chosen uniformly from $\{0, 1\}$. Call $\mathcal{O}(n, s, \langle \mathrm{sam}_f(\mathbf{x}^{<t}), (x_m, b_m), (x_{t+1}, b_{t+1}), \cdots, (x_t, b_t)\rangle)$.*

Output: *If the oracle answers "yes" then predict that the label of x_m is b_m. If the oracle answers "no" then predict that the label of x_m is $\overline{b_m}$.*

Theorem 7.3: *If $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ is learnable and the oracle $\mathcal{O}$ for $\mathcal{F}$ on X is a polynomial consistency oracle then $\widetilde{Q}$ is a polynomial weak prediction algorithm.*

[12] The main resource we are interested in is running time in some standard computational model such as the RAM [AHU74]. All of our algorithms can be implemented so that the space used and number of random bits required is bounded by the running time.

Proof: 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 upper-bounded by some polynomial $p(n,s)$ [BEHW89]. By Sauer's Lemma [Sau72], for any m and $\mathbf{x} \in (X_n)^m$: $|\mathrm{sam}_{\mathcal{F}_s}(\mathbf{x})| \leq \sum_{i=0}^{p(n,s)} \binom{m}{i} \leq m^{p(n,s)} + 1$. Let $m = 2p(n,s)$ so that $\sum_{i=0}^{p(n,s)} \binom{m}{i} = 2^{m-1}$, and $|\mathrm{sam}_{\mathcal{F}_s}(\mathbf{x})| \leq 2^{m-1}$.

We can now apply Theorem 6.5 with concept class $\mathcal{F}_s$ on domain X_n to get for all $f \in \mathcal{F}_s$ and $\mathbf{x} \in (X_n)^m$:

$$\begin{aligned} \sum_{t=1}^{m} \mathbf{M}(Q, f, \mathbf{x}, t) &= \frac{m}{2} + \frac{|\mathrm{sam}_{\mathcal{F}_s}(\mathbf{x})|}{2^m} - 1 \\ &\leq \frac{m}{2} - \frac{1}{2}. \end{aligned}$$

Using Corollary 5.4 we see that:

$$\begin{aligned} \mathbf{E}_{\mathbf{x} \in \mathcal{D}^m}\left[\mathbf{M}(\widetilde{Q}, f, \mathbf{x})\right] &= \frac{1}{m}\mathbf{E}_{\mathbf{x} \in \mathcal{D}^m}\left[\sum_{t=1}^{m} \mathbf{M}(Q, f, \mathbf{x}, t)\right] \\ &\leq \frac{1}{2} - \frac{1}{2m}. \end{aligned}$$

Thus for $p_1(n,s) = 2p(n,s)$ and $p_2(m) = 2m$, Prediction Algorithm $\widetilde{Q}$ is weak prediction algorithm for $\mathcal{F}$ on X.

The value t and the random bits can be extracted from r in $O(m)$ time using real arithmetic.[13] The running time of polynomial consistency oracle $\mathcal{O}$ is by definition bounded by some $p'(n,s,l)$ where l is the bit length of the instances in the sample. Thus there is a $p_3(n,s,l)$ bounding the running time of $\widetilde{Q}$, and Prediction Algorithm $\widetilde{Q}$ is a polynomial weak prediction algorithm. □

8 Sample Complexity of Weak Learning

In the proof of Theorem 7.3 we used a sample size $m = 2p(n,s)$, where $p(n,s)$ is an upper bound on the VC dimension of $\mathcal{F}_s$ on X_n. Smaller sample sizes also suffice. Algorithm $\widetilde{Q}$ is a weak prediction algorithm provided that the sample size m is large enough so that, for some polynomial q, $\sum_{i=0}^{p(n,s)} \binom{m}{i} \leq 2^{m-(1/q(m))}$. By Fact 2.1 this constraint on m is equivalent to: there exists a polynomial q' such that $\sum_{i=0}^{p(n,s)} \binom{m}{i} \leq 2^m(1 - \frac{1}{q'(m)})$.

The goal of this section is to determine using our methods the smallest sample size (as a function of the VC dimension of the concept class) that implies weak learning. Since here we are not interested in computational resources we omit the parameterization during this section. Even though the results in Goldman et al. [GKS90] suggest the sample complexity for weak learning is not well correlated with the VC dimension, the following theorem and corollary gives the lowest sample size known to us ($2d - O(\sqrt{d \log d})$) for which there is a general weak learning algorithm.

Theorem 8.1: *Let d be the VC dimension of some concept class $\mathcal{F}$ on X and c be a constant between $2/\sqrt{d \lg d}$ and $\sqrt{d/\lg d}$. If the sample size $m = 2d + 2 - c\sqrt{d \lg d}$ then for any $\mathbf{x} \in X^m$,*

$$\mathbf{E}_{\mathbf{y} \in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{Q}, f, \mathbf{y})\right] \leq \frac{1}{2} - \frac{1}{2m^{c^2+2}}.$$

[13] When integer arithmetic is used, the uniform distribution on t can be approximated to within $1/2^m$ using m random bits.

Proof: The constraint on c enforces the conditions that $d+2 \leq m \leq 2d$.

Let $q = m^{c^2+1}$. We will show that

$$|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| \leq 2^{m-(1/q)}, \tag{8.1}$$

from which we can apply Corollary 6.6 giving

$$\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{Q}, f, \mathbf{y})\right] \leq \frac{1}{2} - \frac{1}{2mq},$$

as desired.

We will show Inequality 8.1 by giving a sequence of inequalities, each of which implies the previous inequality in the sequence. The first inequality is Inequality 8.1, and the last inequality in the sequence will follow from the conditions of the theorem.

By Fact 2.1

$$|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| \leq 2^m(1 - \frac{\ln 2}{q})$$

implies Inequality 8.1. From Sauer's Lemma [Sau72], for any m and $\mathbf{x} \in (X_n)^m$:

$$\begin{aligned} |\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| &\leq \sum_{i=0}^{d} \binom{m}{i} \\ &= 2^m - \sum_{i=d+1}^{m} \binom{m}{i} \end{aligned}$$

Let $\lambda = \frac{d+1}{m}$, so that $\frac{1}{2} \leq \lambda \leq 1$. We use the following approximation to the binomial coefficient $\binom{m}{\lambda m}$ (see [MS77], Lemma 7, page 309):

$$\sum_{i=d+1}^{m} \binom{m}{i} \geq \binom{m}{\lambda m} \geq \frac{2^{mH_2(\lambda)}}{\sqrt{8m\lambda(1-\lambda)}}$$

where $H_2(\lambda) = \lambda \lg \frac{1}{\lambda} + (1-\lambda)\lg\frac{1}{1-\lambda}$. Thus

$$|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| \leq 2^m - \frac{2^{mH_2(\lambda)}}{\sqrt{8m\lambda(1-\lambda)}}$$

and it suffices to show that

$$2^m - \frac{2^{mH_2(\lambda)}}{\sqrt{8m\lambda(1-\lambda)}} \leq 2^m(1 - \frac{\ln 2}{q}) \tag{8.2}$$

$$\iff \quad \frac{q}{\ln 2}\frac{2^{mH_2(\lambda)}}{\sqrt{8m\lambda(1-\lambda)}} \geq 2^m. \tag{8.3}$$

Note that since $\frac{1}{2} \leq \lambda \leq 1$, we have $\lambda(1-\lambda) \leq 1/4$, and it suffices to show

$$\frac{q}{\ln 2}\frac{2^{mH_2(\lambda)}}{\sqrt{2m}} \geq 2^m,$$

and since $1/(\sqrt{2}\ln 2) \leq 1$,

$$\frac{q2^{mH_2(\lambda)}}{\sqrt{m}} \geq 2^m,$$

suffices.

Taking logs of both sides gives us that

$$\lg(q/\sqrt{m}) + mH_2(\lambda) \geq m$$

suffices. Since $H_2(\lambda) \geq 4\lambda(1-\lambda)$ for $\lambda \in [0,1]$, it suffices to show

$$\lg(q/\sqrt{m}) + 4m\lambda(1-\lambda) \geq m$$

$$\iff 4\lambda(1-\lambda) \geq 1 - \frac{1}{m}\lg(q/\sqrt{m})$$

$$\iff 4\lambda(1-\lambda) \geq 1 - \frac{1}{2m}\lg(q^2/m).$$

Note that $4\lambda(1-\lambda) \geq 1-x$ if and only if $\frac{1}{2} - \frac{1}{2}\sqrt{x} \leq \lambda \leq \frac{1}{2} + \frac{1}{2}\sqrt{x}$. Therefore (using $x = \frac{1}{2m}\lg(q^2/m)$), since $\lambda \geq \frac{1}{2}$, it suffices to show that

$$\lambda \leq \frac{1}{2} + \frac{1}{2}\sqrt{\frac{1}{2m}\lg(q^2/m)} = \frac{1}{2} + \sqrt{\frac{1}{8m}\lg(q^2/m)}.$$

Substituting $\frac{d+1}{m}$ for λ gives us that this is equivalent to

$$\frac{d+1}{m} \leq \frac{1}{2} + \sqrt{\frac{1}{8m}\lg(q^2/m)}$$

or

$$d+1 \leq \frac{m}{2} + \sqrt{\frac{m}{8}\lg(q^2/m)}.$$

Since $m \geq 2d + 2 - c\sqrt{d\lg d}$, and $q = m^{c^2+1}$,

$$\begin{aligned}
\frac{m}{2} + \sqrt{\frac{m}{8}\lg(q^2/m)} &\geq \frac{2d+2-c\sqrt{d\lg d}}{2} + \sqrt{\frac{m}{8}\lg m^{2c^2+1}} \\
&\geq d+1 - \frac{c}{2}\sqrt{d\lg d} + \sqrt{\frac{m}{8}\lg m^{2c^2+1}} \\
&\geq d+1 - \frac{c}{2}\sqrt{d\lg d} + \sqrt{\frac{m2c^2}{8}\lg m} \\
&\geq d+1 - \frac{c}{2}\sqrt{d\lg d} + \frac{c}{2}\sqrt{d\lg d} \\
&\geq d+1
\end{aligned}$$

Thus $d+1 \leq \frac{m}{2} + \sqrt{\frac{m}{8}\lg(q^2/2m)}$, completing the proof. □

Corollary 8.2: *Let d be the VC dimension of some concept class $\mathcal{F}$ on X. Then there is a weak learning algorithm for $\mathcal{F}$ using sample size $2d - O(\sqrt{d\log d})$.*

Proof: This follows from Theorem 8.1 and Lemma 5.1 □

9 Bounds on Gibbs and Bayesian prediction

In this section we describe and compare a number of prediction algorithms related to the query prediction algorithm presented in the previous sections. Some of these algorithms will have better performance than the Query Lookahead Prediction Algorithm for sparser concept classes, i.e. when $|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| = 2^{|\mathbf{x}|-\alpha}$ for $\alpha \gg 1$. Unfortunately, these algorithms are generally not polynomial.

In this section we take a Bayesian view point and assume that there is a prior probability distribution $\mathcal{P}$ on the concept class $\mathcal{F}$. For each $f \in \mathcal{F}$, $\mathcal{P}(f)$ represents the extent to which the learning algorithm initially (before seeing any examples) believes that f is the target function to be learned. After seeing a number of examples some concepts might be inconsistent with the past examples and the class of possible targets shrinks. 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 $x \in X$, we have $V^{\mathcal{P}}(S) = V^{\mathcal{P}}(S,(x,0)) + V^{\mathcal{P}}(S,(x,1))$.

Definition 9.1 (Volume Prediction Algorithm): *Algorithm A is a* volume prediction algorithm *for a prior $\mathcal{P}$ on $\mathcal{F}$ if there is a function g such that for all $\mathbf{x} \in X^+$ the probability that $A(\mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|}), x_{|\mathbf{x}|}, r)$ predicts 1 is*

$$g(V^{\mathcal{P}}(\langle \mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|}),(x_{|\mathbf{x}|},1)\rangle), V^{\mathcal{P}}(\langle \mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|}),(x_{|\mathbf{x}|},0)\rangle)))$$

and the probability that A predicts 0 is

$$g(V^{\mathcal{P}}(\langle \mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|}),(x_{|\mathbf{x}|},0)\rangle), V^{\mathcal{P}}(\langle \mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|}),(x_{|\mathbf{x}|},1)\rangle))).$$

Since every prediction algorithm predicts either 0 or 1, any g used in the above definition has the properties $g(\nu,\nu') = 1 - g(\nu',\nu)$ and $g(\nu,\nu) = \frac{1}{2}$. Although the function g used by a volume prediction algorithm may be simple, computing the volume of a sample may not be computationally feasible.

Here we consider three volume prediction algorithms. Algorithm Gibbs$^{\mathcal{P}}$ (Gibbs Algorithm) is well known [HKS91, HO91, GT90, HS90, STS90] and can be viewed as predicting with a randomly chosen consistent concept from the class where the consistent concepts are weighted according to the prior $\mathcal{P}$. Algorithm $G^{\mathcal{P}}$ is a special case of the aggregating strategy introduced by Vovk [Vov90], and was used as the basis for a polynomial weak prediction algorithm [HW92b]. The classical Bayes Prediction Algorithm, Bayes$^{\mathcal{P}}$, is known to be optimal when the target is drawn according to the prior. For any $\mathcal{F}$ on X, prior $\mathcal{P}$ on $\mathcal{F}$, prediction algorithm A, $m \in \mathbf{N}$, and $\mathbf{x} \in X^m$:

$$\mathbf{E}_{f\in\mathcal{P}}[\mathbf{M}(A,f,\mathbf{x})] \geq \mathbf{E}_{f\in\mathcal{P}}\left[\mathbf{M}(\mathrm{Bayes}^{\mathcal{P}},f,\mathbf{x})\right].$$

Figure 9.1 gives the prediction rules and Figure 9.2 contains a graphical comparison of these three prediction algorithms. In Figures 9.1 and 9.2, ρ_1 denotes

$$\frac{V^{\mathcal{P}}(\langle \mathrm{sam}_f(x^{<m}),(x_m,1)\rangle)}{V^{\mathcal{P}}(\mathrm{sam}_f(x^{<m}))}$$

and $\rho_0 = 1 - \rho_1$. Since

$$V^{\mathcal{P}}(\langle \mathrm{sam}_f(x^{<m}),(x_m,0)\rangle) + V^{\mathcal{P}}(\langle \mathrm{sam}_f(x^{<m}),(x_m,1)\rangle) = V^{\mathcal{P}}(\mathrm{sam}_f(x^{<m}))$$

Algorithm	probability prediction = 1	probability prediction = 0
Gibbs$^{\mathcal{P}}$	ρ_1	ρ_0
$G^{\mathcal{P}}$	$\frac{-\lg \rho_0}{-\lg \rho_0 - \lg \rho_1}$	$\frac{-\lg \rho_1}{-\lg \rho_0 - \lg \rho_1}$
Bayes$^{\mathcal{P}}$	1 if $\rho_1 > \frac{1}{2}$ $\frac{1}{2}$ if $\rho_1 = \frac{1}{2}$ 0 if $\rho_1 < \frac{1}{2}$	1 if $\rho_0 > \frac{1}{2}$ $\frac{1}{2}$ if $\rho_0 = \frac{1}{2}$ 0 if $\rho_0 < \frac{1}{2}$

Figure 9.1: Comparison of three prediction algorithms.

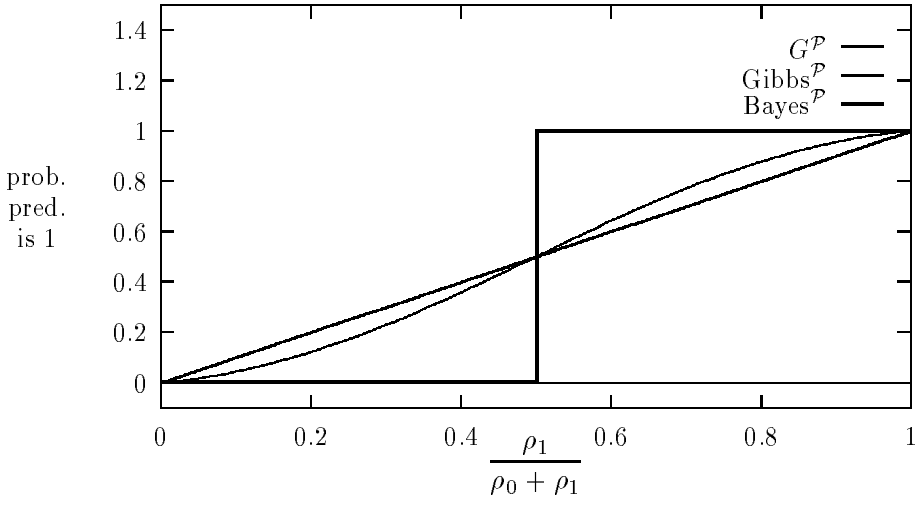

Figure 9.2: The probability that the Algorithms Gibbs$^{\mathcal{P}}$, $G^{\mathcal{P}}$, and Bayes$^{\mathcal{P}}$ predict 1 as a function of $\rho_1/(\rho_0 + \rho_1)$.

it is easy to see that $\rho_0 = \frac{V^{\mathcal{P}}(\langle \text{sam}_f(x^{<m}),(x_m,0)\rangle)}{V^{\mathcal{P}}(\text{sam}_f(x^{<m}))}$ and the definitions of ρ_1 and ρ_0 are symmetric.

The information gain (or amount of surprise) in the last example of a sample $\langle S, e\rangle$ is commonly defined as $-\lg(V^{\mathcal{P}}(\langle S,e\rangle)/V^{\mathcal{P}}(S))$. Using our ρ notation, the information gain on a trial is $-\lg \rho_1$ if the instance is labeled "1" and $-\lg \rho_0$ otherwise. Algorithm $G^{\mathcal{P}}$ predicts with the relative amount of information gained by the two possible outcomes, as shown in Figure 9.1 (See Appendix A for a more detailed treatment). Therefore we call Algorithm $G^{\mathcal{P}}$ the *Information Gain Prediction Algorithm.*

In this paper our goal is to construct weak prediction algorithms. This requires that we minimize the worst case (over all possible targets $f \in \mathcal{F}$) probability of an incorrect predication on the last instance. Bayes Algorithm minimizes the average case rather than the worst case. As the following lemma shows, all three algorithms can have a large (worst case) probability of a mistake on the last instance of $\mathbf{x}$, even when $|\mathcal{F}| \ll 2^{|\mathbf{x}|}$.

Lemma 9.2: *For every m there is a concept class $\mathcal{F}$ of $m+1$ concepts on domain X, $f \in \mathcal{F}$, and $\mathbf{x} \in X^m$ such that for any volume prediction algorithm $A^{\mathcal{P}}$ using the uniform prior on $\mathcal{F}$:*

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

Proof: Let $\{1, \cdots, m\}$ be the domain and $\mathcal{F} = \{f_0, \cdots, f_m\}$ where $f_i(j) = 1$ if and only if $i = j$ (for $0 \leq i \leq m$ and $1 \leq j \leq m$). Thus f_0 is the constant function 0 and each other f_i has the value 1 on exactly one instance. Let $\mathcal{P}$ be uniform on $\mathcal{F}$ and $\mathbf{x} = \langle 1, \cdots, m \rangle$. Then for any permutation $\mathbf{y} \in U(\mathbf{x})$,

$$\frac{V^{\mathcal{P}}(\mathrm{sam}_{f_0}(\mathbf{y}^{\leq m}))}{V^{\mathcal{P}}(\mathrm{sam}_{f_0}(\mathbf{y}^{<m}))} = \frac{1}{2}$$

and thus when f_0 is the target, any volume prediction algorithm incorrectly predicts that the label of y_m is 1 with probability $\frac{1}{2}$. □

Note that the Bayes Algorithm makes at most one prediction error on the whole sequence $\mathbf{y}$ from the proof of Lemma 9.2. For Algorithms Gibbs$^{\mathcal{P}}$ and $G^{\mathcal{P}}$, the errors are also likely to be concentrated at the end of the sequence. We use a simple trick to circumvent this potential for volume algorithms to predict incorrectly on the last instance.

We associate a special prior with the lookahead versions of volume prediction algorithms. Recall that a lookahead prediction algorithm is given not only some sequence of examples, $\mathrm{sam}_f(\mathbf{x}^{<t})$, and the instance to predict on, x_t, but also a sequence of unlabeled instances, $\mathbf{x}^{>t}$ (representing future instances on which the algorithm will be asked to predict). We let our prior depend on $\mathbf{x}$, the entire sequence of instances presented to the lookahead algorithm. Since our goal is good worst-case prediction, it is natural to weight each labeling of $\mathbf{x}$ consistent with a target in $\mathcal{F}$ equally. Thus we use the uniform prior $\mathcal{P}_{\mathbf{x}}$ on $\mathrm{sam}_{\mathcal{F}}(\mathbf{x})$ where each volume $V^{\mathcal{P}_{\mathbf{x}}}(S)$ for $S \in \mathrm{sam}_{\mathcal{F}}(\mathbf{x})$ is $\frac{1}{|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})|}$. We then apply the Lookahead Conversion to these lookahead algorithms with the special prior $\mathcal{P}_{\mathbf{x}}$ to obtain prediction algorithms.

We use $\widetilde{\text{Bayes}}$ to denote the algorithm that result from applying the Lookahead Conversion to the Bayes Algorithm. Thus Prediction Algorithm $\widetilde{\text{Bayes}}$ is given a sample $\mathrm{sam}_f(\mathbf{x}^{<|\mathbf{x}|})$, an instance $x_{|\mathbf{x}|}$ to predict on, and a random r from $U_{[0,1]}$. Algorithm $\widetilde{\text{Bayes}}$ first constructs the prior $\mathcal{P}_{\mathbf{x}}$ giving each sample in $\mathrm{sam}_{\mathcal{F}}(\mathbf{x})$ the same probability. Algorithm $\widetilde{\text{Bayes}}$ then splits r into a t chosen uniformly from $\{1, \cdots, |\mathbf{x}|\}$ and an r' from $U_{[0,1]}$. Algorithm $\widetilde{\text{Bayes}}$ obtains its prediction by calling $\text{Bayes}^{\mathcal{P}_{\mathbf{x}}}(\mathrm{sam}_f(\mathbf{x}^{<t}), x_{|\mathbf{x}|}, r')$.

Prediction Algorithms $\widetilde{\text{Gibbs}}$ and $\widetilde{G}$ are the Gibbs and Information Gain Algorithms transformed in the same way. Note that these transformed algorithms "manufacture" their own "priors" from the instances rather than obtaining a prior from the outside world.

We will apply Theorem 5.3 to bound the probability that the lookahead conversions predict incorrectly. This requires that we obtain bounds on the expected *total* number of mistakes made by the three lookahead algorithms. For any $\mathbf{x} \in X^*$, target $f \in \mathcal{F}$, and prior $\mathcal{P}$ on $\mathcal{F}$, the following bounds are known:

$$\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\text{Bayes}^{\mathcal{P}}, f, \mathbf{x}) \leq -\lg V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x})) \tag{9.1}$$

$$\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(G^{\mathcal{P}}, f, \mathbf{x}) \leq -\frac{1}{2}\lg V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x})) \tag{9.2}$$

$$\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\text{Gibbs}^{\mathcal{P}}, f, \mathbf{x}) \quad \leq \quad -(\ln 2) \lg V^{\mathcal{P}}(\text{sam}_f(\mathbf{x})) \tag{9.3}$$

The bounds on Bayes$^{\mathcal{P}}$ and Gibbs$^{\mathcal{P}}$ were shown in [HKS91] and the bound on $G^{\mathcal{P}}$ appears in [Vov90] and is presented in Appendix A. It is easy to show that the constants in these bounds cannot be improved unless the form of the bounds are changed. 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

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

For Gibbs, set

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

so that

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

By letting ϵ go to zero in both examples it can be seen that the constant factors of 1 for Bayes in Equation 9.1 and ln(2) for Gibbs in Equations 9.3 cannot be improved.

For the Information Gain Prediction algorithm, when $\mathcal{P}(f) = \mathcal{P}(g) = \frac{1}{2}$, the mistake probability $\mathbf{M}(G^{\mathcal{P}}, f, \mathbf{x})$ is $\frac{1}{2} = -\frac{1}{2} \lg V^{\mathcal{P}}(\text{sam}_f(\mathbf{x}))$.

These examples for Bayes$^{\mathcal{P}}$ and Gibbs$^{\mathcal{P}}$ rely on choosing a particular target. Better bounds can be shown in the average case setting, where the target is chosen at random using the same distribution as the prior. For this average case setting, the constant of $-\frac{1}{2}$ has also been obtained for the Gibbs and Bayes Algorithms [HKS91]. More precisely, they show that for any $\mathbf{x} \in X^*$:

$$\mathbf{E}_{f \in \mathcal{P}} \left[\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\text{Bayes}^{\mathcal{P}}, f, \mathbf{x}) \right] \leq \mathbf{E}_{f \in \mathcal{P}} \left[\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\text{Gibbs}^{\mathcal{P}}, f, \mathbf{x}) \right] \leq -\frac{1}{2} \mathbf{E}_{f \in \mathcal{P}} \left[\lg V^{\mathcal{P}}(\text{sam}_f(\mathbf{x})) \right].$$

The bounds in Equations 9.1, 9.2, and 9.3 are unsatisfactory when the volume is small. When $V^{\mathcal{P}}(\text{sam}_f(\mathbf{x})) < 1/4^{|\mathbf{x}|}$, all three mistake bounds are greater than $|\mathbf{x}|$, the number trials. Since the number of trials is a trivial bound on the number of mistakes made, the upper bounds are vacuous in this case.

We now present an improved worst case bound on $\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\text{Gibbs}^{\mathcal{P}}, f, \mathbf{x})$ that is at most $|\mathbf{x}|$ even when $V^{\mathcal{P}}(\text{sam}_f(\mathbf{x}))$ is arbitrarily small.

Theorem 9.3: *For any instance sequence $\mathbf{x} \in X^+$, target $f \in \mathcal{F}$, and prior $\mathcal{P}$ on $\mathcal{F}$,*

$$\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\mathit{Gibbs}^{\mathcal{P}}, f, \mathbf{x}^{<t}) \leq |\mathbf{x}| \, (1 - (V^{\mathcal{P}}(\text{sam}_f(\mathbf{x})))^{1/|\mathbf{x}|}).$$

Proof: Let $\nu_t = V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x}^{\leq t}))/V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x}^{<t}))$ for each $1 \leq t \leq |\mathbf{x}|$.

If follows from the definition of $\mathrm{Gibbs}^{\mathcal{P}}$ that $\mathbf{M}(\mathrm{Gibbs}^{\mathcal{P}}, f, \mathbf{x}^{\leq t}) = 1 - \nu_t$. Furthermore,

$$\prod_{t=1}^{|\mathbf{x}|} \nu_t = \prod_{t=1}^{|\mathbf{x}|} \frac{V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x}^{\leq t}))}{V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x}^{<t}))} = V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x})).$$

Thus we are bounding the sum

$$\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\mathrm{Gibbs}^{\mathcal{P}}, f, \mathbf{x}^{\leq t}) = \sum_{t=1}^{|\mathbf{x}|}(1 - \nu_t) = |\mathbf{x}| - \sum_{t=1} \nu_t$$

subject to the constraint that $\prod_{t=1}^{|\mathbf{x}|} \nu_t = V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x}))$. The theorem follows from the fact that the sum of the mistake probabilities is maximized when each $\nu_t = (V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x})))^{1/|\mathbf{x}|}$. □

Note that as $V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x})) \to 0$ the bound of Theorem 9.3 goes to $|\mathbf{x}|$, whereas the bound in Equation 9.3 goes to ∞. Both bounds show that zero mistakes are expected when $V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x})) = 1$. In fact, for any $z \in [0,1]$ and $m \geq 1$, $-\ln z \geq m(1 - z^{1/m})$. Thus the bound for $\mathrm{Gibbs}^{\mathcal{P}}$ in Theorem 9.3 is always smaller than the bound of Equation 9.3. Furthermore, the same argument used to show that the constant in Equation 9.3 was tight shows that the constant factor of one in the bound of Theorem 9.3 can not be improved.

We are now ready to state bounds on the performance of the converted Prediction Algorithms $\widetilde{\mathrm{Bayes}}$, $\widetilde{\mathrm{Gibbs}}$, and $\widetilde{G}$.

Theorem 9.4: *Let $\mathcal{F}$ be a concept class on X. For all instance sequences $\mathbf{x} \in X^+$ where $|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| = 2^{|\mathbf{x}|-\alpha}$, and targets $f \in \mathcal{F}$,*

$$\begin{aligned}
\mathbf{E}_{\mathbf{y} \in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{Bayes}, f, \mathbf{y})\right] &\leq 1 - \frac{\alpha}{|\mathbf{x}|} \\
\mathbf{E}_{\mathbf{y} \in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{G}, f, \mathbf{y})\right] &\leq \frac{1}{2} - \frac{\alpha}{2\,|\mathbf{x}|} \\
\mathbf{E}_{\mathbf{y} \in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{Gibbs}, f, \mathbf{y})\right] &\leq 1 - 2^{\frac{\alpha}{|\mathbf{x}|}-1} \leq \ln(2) - \frac{\alpha \ln(2)}{|\mathbf{x}|}
\end{aligned}$$

Proof: We first note that under the manufactured priors,

$$V^{\mathcal{P}_{\mathbf{x}}}(\mathrm{sam}_f(\mathbf{x})) = \frac{1}{|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})|} = \frac{1}{2^{|\mathbf{x}|-\alpha}}.$$

Plugging this into Equations 9.1, 9.2, and 9.3, and the bound of Theorem 9.3 gives us:

$$\begin{aligned}
\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\mathrm{Bayes}^{\mathcal{P}_{\mathbf{x}}}, f, \mathbf{x}) &\leq |\mathbf{x}| - \alpha \\
\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(G^{\mathcal{P}_{\mathbf{x}}}, f, \mathbf{x}) &\leq \frac{1}{2}(|\mathbf{x}| - \alpha) \\
\sum_{t=1}^{|\mathbf{x}|} \mathbf{M}(\mathrm{Gibbs}^{\mathcal{P}_{\mathbf{x}}}, f, \mathbf{x}) &\leq |\mathbf{x}|\,(1 - 2^{\frac{\alpha}{|\mathbf{x}|}-1}) \leq (\ln 2)(|\mathbf{x}| - \alpha), \text{ (Fact 2.1).}
\end{aligned}$$

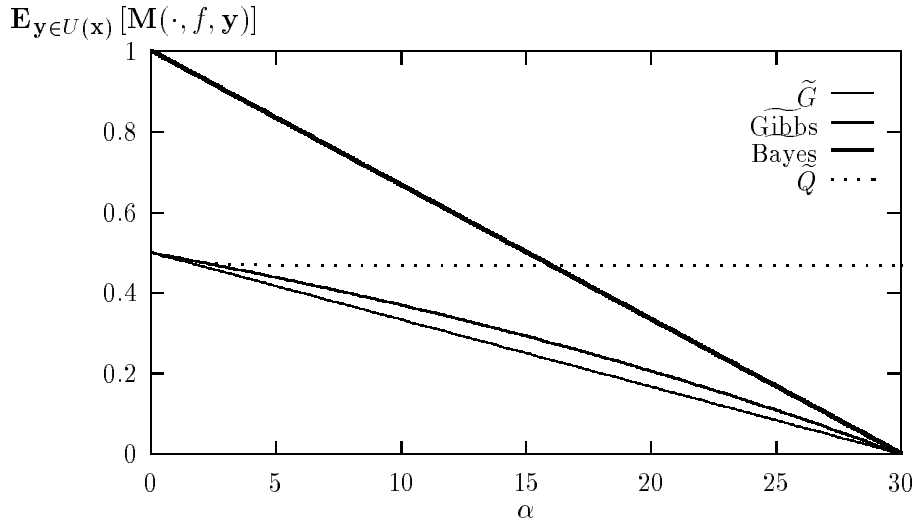

Figure 9.3: The bounds of Theorem 9.4 and Corollary 6.6 as a function of α when $|\mathbf{x}| = 30$.

Now we can apply Theorem 5.3 to each of the above inequalities to obtain:

$$\begin{aligned}
\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{\text{Bayes}}, f, \mathbf{y})\right] &\leq 1 - \frac{\alpha}{|\mathbf{x}|} \\
\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{G}, f, \mathbf{y})\right] &\leq \frac{1}{2} - \frac{\alpha}{2\,|\mathbf{x}|} \\
\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{\text{Gibbs}}, f, \mathbf{y})\right] &\leq 1 - 2^{\frac{\alpha}{|\mathbf{x}|}-1} \leq \ln(2) - \frac{\alpha \ln(2)}{|\mathbf{x}|}
\end{aligned}$$

as desired. □

In Figure 9.3 we graph the above bounds for Algorithms $\widetilde{\text{Bayes}}$, $\widetilde{G}$, and $\widetilde{\text{Gibbs}}$ together with the bound for $\widetilde{Q}$ given in Corollary 6.6:

$$\mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}\left[\mathbf{M}(\widetilde{Q}, f, \mathbf{y})\right] \leq \frac{1}{2} - \frac{1}{|\mathbf{x}|} + \frac{2^{-\alpha}}{|\mathbf{x}|}.$$

In contrast to Algorithm $\widetilde{Q}$, the other three algorithms ($\widetilde{\text{Bayes}}$, $\widetilde{G}$, and $\widetilde{\text{Gibbs}}$) are not polynomial since the uniform prior $\mathcal{P}_{\mathbf{x}}$ on $\text{sam}_{\mathcal{F}}(\mathbf{x})$ is not efficiently computable. However, in our previous paper [HW92b] a polynomial approximation to $\widetilde{G}$ was given. The number of consistency oracle queries used by this approximation to $\widetilde{G}$ is $\Omega(\frac{2^{\alpha'}-1}{\alpha'})$ where α' is an underestimate of α which must be supplied to the algorithm.

The simple Algorithm $\widetilde{Q}$ presented here which uses only one query works for all α, however its probability of predicting wrong on the last instance is never more than $\frac{1}{|\mathbf{x}|}$ below $\frac{1}{2}$.

The optimal algorithm for minimizing the worst case (over targets $f \in \mathcal{F}$) probability of predicting wrong on the last instance of a random permutation of $\mathbf{x}$ is the 1-Inclusion Graph Algorithm [HLW], here denoted by "1-Inc." For prediction algorithm A and any $\mathbf{x} \in X^+$,

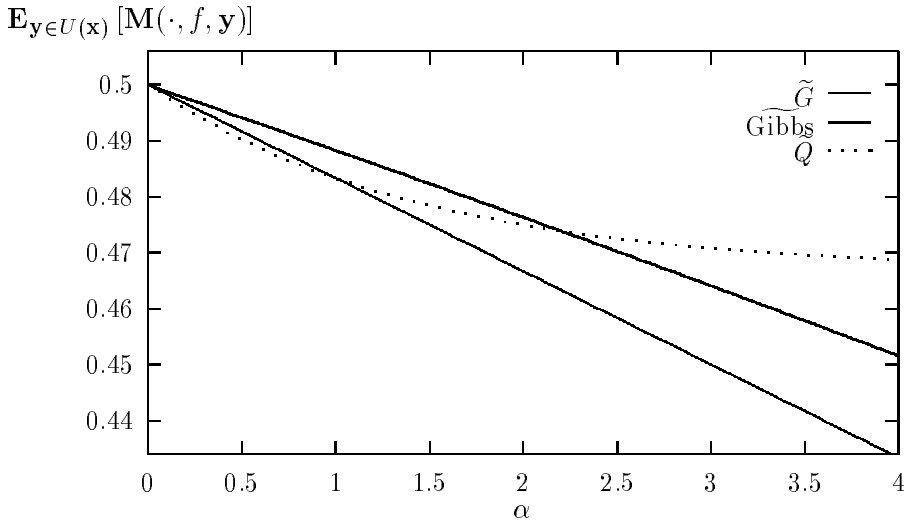

Figure 9.4: An expanded plot of part of Figure 9.3 showing the bounds of Theorem 9.4 and Corollary 6.6 as a function of α when $|\mathbf{x}| = 30$.

$$\sup_{f\in\mathcal{F}} \mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}[\mathbf{M}(A, f, \mathbf{y})] \geq \sup_{f\in\mathcal{F}} \mathbf{E}_{\mathbf{y}\in U(\mathbf{x})}[\mathbf{M}(\text{1-Inc}, f, \mathbf{y})] = \frac{\text{maxdens}_{\mathcal{F}}(\mathbf{x})}{|\mathbf{x}|} \tag{9.4}$$

where $\text{maxdens}_{\mathcal{F}}(\mathbf{x})$ 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 $\mathbf{x}$. In Haussler, Littlestone, and Warmuth [HLW] it is shown that $\text{maxdens}_{\mathcal{F}}(\mathbf{x})$ is upper bounded by the Vapnik-Chervonenkis dimension of the class $\mathcal{F}$ [VC71, BEHW89]. The main drawback of the 1-Inclusion Graph Prediction Algorithm is that it is not generally efficient as it solves flow problems on graphs containing $|\text{sam}_{\mathcal{F}}(\mathbf{x})|$ vertices.

Since for any prediction algorithm (and in particular Algorithms $\widetilde{Q}$ and $\widetilde{G}$) the probability of predicting wrong on the last instance of a random permutation of any $\mathbf{x} \in X^+$ is at least $\text{maxdens}_{\mathcal{F}}(\mathbf{x})/|\mathbf{x}|$ (Inequality 9.4) we get two additional upper bounds on $\text{maxdens}_{\mathcal{F}}(\mathbf{x})$ from Theorem 9.4 and Corollary 6.6.

Corollary 9.5: *For any concept class $\mathcal{F}$ on X of VC dimension d, and any $\mathbf{x} \in X^+$,*

$$\textit{maxdens}_{\mathcal{F}}(\mathbf{x}) \leq \min\{d, \frac{1}{2}\lg|\text{sam}_{\mathcal{F}}(\mathbf{x})|, \frac{1}{2} + \frac{|\text{sam}_{\mathcal{F}}(\mathbf{x})|}{2^{|\mathbf{x}|}} - \frac{1}{|\mathbf{x}|}\}.$$

In Appendix B we give the definition of the 1-inclusion graph, its density, and an inductive proof that $\text{maxdens}_{\mathcal{F}}(\mathbf{x}) \leq \frac{1}{2}\lg|\text{sam}_{\mathcal{F}}(\mathbf{x})|$. The inductive proof in the appendix is direct, and does not use the performance of Algorithm $\widetilde{G}$.

The bound of $\frac{1}{2}\lg|\text{sam}_{\mathcal{F}}(\mathbf{x})|$ from Algorithm $\widetilde{G}$ is better than the bound of d when $|\mathbf{x}| < 2d + 1$. Also, as hinted in Figure 9.4, Algorithm $\widetilde{Q}$ is better than the bounds proven for Algorithm $\widetilde{G}$ when $\alpha \in (0, 1)$. Thus the last bound of Corollary 9.5, $\frac{1}{2} + \frac{|\text{sam}_{\mathcal{F}}(\mathbf{x})|}{2^{|\mathbf{x}|}} - \frac{1}{|\mathbf{x}|}$, is the best of the three when $\text{sam}_{\mathcal{F}}(\mathbf{x}) \geq 2^{|\mathbf{x}|-1}$. This indicates that it may be possible to prove even better upper bounds on $\text{maxdens}_{\mathcal{F}}(\mathbf{x})$.

Alg.	Prob. of predicting 1		
	$n_1 = 0$	$n_1 = n_0$	$n_0 = 0$
$\text{Gibbs}^{\mathcal{P}_\mathbf{x}}$	0	$\frac{1}{2}$	1
Q	$\frac{1}{2} - \frac{n_0}{2^{\lvert\mathbf{x}\rvert-t+1}}$	$\frac{1}{2}$	$\frac{1}{2} + \frac{n_1}{2^{\lvert\mathbf{x}\rvert-t+1}}$

Figure 9.5: Comparison of Algorithm $\text{Gibbs}^{\mathcal{P}_\mathbf{x}}$ and Algorithm Q.

To get more intuition about our Algorithm $\widetilde{Q}$ we compare it with $\widetilde{\text{Gibbs}}$. Recall that $N_\mathcal{F}(S, \mathbf{x})$ is the number of samples in $\text{sam}_\mathcal{F}(\mathbf{x})$ that are consistent with S (Definition 6.3). For a fixed t in $\{1, 2, \cdots, |\mathbf{x}|\}$, let

$$n_0 = N_\mathcal{F}(\langle \text{sam}_f(\mathbf{x}^{<t}), (x_{|\mathbf{x}|}, 0)\rangle)$$

and

$$n_1 = N_\mathcal{F}(\langle \text{sam}_f(\mathbf{x}^{<t}), (x_{|\mathbf{x}|}, 1)\rangle).$$

Using this notation, Lookahead Algorithm Q on input $\text{sam}_f(\mathbf{x}^{<t})$, x_t, $\mathbf{x}^{>t}$, and $r \in U_{[0,1]}$ predicts one with probability

$$\frac{1}{2} + \frac{n_1 - n_0}{2^{|\mathbf{x}|-t+1}} = \frac{1}{2} - \frac{n_0 + n_1}{2^{|\mathbf{x}|-t+1}} + \frac{2n_1}{n_0 + n_1} \cdot \frac{n_0 + n_1}{2^{|\mathbf{x}|-t+1}}.$$

In contrast the probability that the lookahead version of $\text{Gibbs}^{\mathcal{P}_\mathbf{x}}$ predicts one on the same input is

$$\frac{1}{2} + \frac{n_1 - n_0}{2(n_0 + n_1)} = \frac{n_1}{n_0 + n_1}.$$

If $\frac{n_0+n_1}{2^{|\mathbf{x}|-t+1}}$ were a constant, then both algorithms would predict one with a probability that is linear in $\frac{n_1}{n_0+n_1}$. However, since $n_0 + n_1 \leq 2^{|\mathbf{x}|-t+1}$ the predictions of Q are more heavily biased towards $\frac{1}{2}$ than the predictions of $\text{Gibbs}^{\mathcal{P}_\mathbf{x}}$. This bias decreases as n_0+n_1 approaches $2^{|\mathbf{x}|-t+1}$.

Clearly Algorithm Q is not optimal since in the extreme case where $n_0 = 0$ Algorithm Q incorrectly predicts 0 with positive probability (unless $n_1 = 2^{|\mathbf{x}|-t}$). Algorithm Q can also make mistakes in the other extreme, when $n_1 = 0$. Algorithm $\text{Gibbs}^{\mathcal{P}_\mathbf{x}}$ predicts optimally for these cases, as indicated in Figure 9.5.

10 A Characterization of Weak Learning Using Consistency Oracles

In this section we show that Prediction Algorithm $\widetilde{Q}$ remains a weak learning algorithm even when it uses a less powerful "one-sided" or "probabilistic" consistency oracle. In addition, we will show that any polynomial weak learning algorithm can be used to construct a polynomial "probabilistic" consistency oracle and visa versa. Therefore a concept class is polynomially weakly learnable if and only if the concept class has a polynomial "probabilistic" consistency oracle.

A one-sided consistency oracle is a consistency oracle that need not always be correct. We call it "one-sided" because it can return false positives but not false negatives. Since an oracle that always answers "yes" is useless, we require that one-sided oracles answer "no" on a significant (1/polynomial) fraction of their inputs.

Definition 10.1 (One-Sided Consistency Oracle): *Oracle $\mathcal{O}$ is a* one-sided consistency oracle *for $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ if there exist polynomials p_1 and p_2 such that $\mathcal{O}$ has the following properties. When $\mathcal{O}$ is given the parameters n and s along with a sample $S \in \text{sam}_*(\mathbf{x})$, for $\mathbf{x} \in (X_n)^m$ where $m = p_1(n, s)$, Oracle $\mathcal{O}$ must answer "yes" if S is consistent with a concept in $\mathcal{F}_s$ and may answer either "yes" or "no" otherwise. However, the total number of "no" answers on the 2^m samples in $\text{sam}_*(\mathbf{x})$ must be at least $2^m/p_2(m)$. Such an oracle is called* polynomial *if its answers are computed in time polynomial in n, s, and the total bit length of S.*

Consistency oracles for large classes are often one-sided consistency oracles for smaller classes. Let $\mathcal{H}_s$ be a class where each $\mathbf{x}$ of $m = p(n, s)$ instances has at most $2^m(1 - \frac{1}{p_2(m)})$ labelings that are consistent with concepts in $\mathcal{H}_s$. Then a consistency oracle for $\mathcal{H}_s$ is also a one-sided consistency oracle for any $\mathcal{F}_s$ contained in $\mathcal{H}_s$.

If we have a polynomial weak Occam algorithm using hypothesis class $\mathcal{H}_{n,s}$ and a polynomial (in n, s, and the length of its input) decision algorithm which determines (when given n, s, and the representation of hypothesis h) whether or not hypothesis h is in $\mathcal{H}_{n,s}$, then we can construct a polynomial one-sided consistency oracle. Simply run the polynomial weak Occam algorithm on the sample and check that the returned hypothesis is in $\mathcal{H}_{n,s}$ and that it is consistent with the sample.

We now extend Theorem 7.3 to show if when Algorithm $\widetilde{Q}$ uses a polynomial one-sided consistency oracle then Algorithm $\widetilde{Q}$ is a polynomial weak prediction algorithm.

Theorem 10.2: *If Algorithm $\widetilde{Q}$ uses a polynomial one-sided consistency oracle for $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ then Algorithm $\widetilde{Q}$ is a polynomial weak prediction algorithm for $\mathcal{F}$ on X.*

Proof: By definition, when $m = p_1(n, s)$ and $\mathbf{x} \in (X_n)^m$, the number of "no" answers given by the oracle on queries in $\text{sam}_*(\mathbf{x})$ is is at least $2^m/p_2(m)$. Equivalently, the total number of "yes" answers is at most $2^m(1 - \frac{1}{p_2(m)})$. By Fact 2.1, there is a polynomial p_2' such that the total number of yes answers is at most $2^{m-(1/p_2'(m))}$. Now Corollary 6.6 and Theorem 7.3 imply that Algorithm $\widetilde{Q}$ is a polynomial weak prediction algorithm. □

Although Algorithm $\widetilde{Q}$ remains a weak learning algorithm when using a one-sided consistency oracle (which can give false positives), it is a different matter if the "consistency" oracle can return false negatives. The following shows that Algorithm $\widetilde{Q}$ can not weakly learn even a very simple concept class when the consistency oracle is incorrect on only a single labeling of each example sequence.

Consider the concept class $\mathcal{F}$ containing two functions: one labeling the entire domain one, and the other labeling the entire domain zero. Thus for every sequence $\mathbf{x}$ of m instances, exactly two samples in $\text{sam}_*(\mathbf{x})$ are consistent with $\mathcal{F}$. Assume the target concept is the all-one concept and the consistency oracle answers "yes" on only all-zero labelings (and thus incorrectly answers "no" only on the all-one labeling). Half the time the random label given to the instance to be predicted on by Algorithm $\widetilde{Q}$ is one. In this case the faulty oracle answers "no" and Algorithm $\widetilde{Q}$ incorrectly predicts zero. There is also some small chance that all of the instances will be given random labels and all of the random labels are set to zero. The oracle will answer yes in this case, and again Algorithm $\widetilde{Q}$ incorrectly predicts zero. Thus not only is Algorithm $\widetilde{Q}$ not a weak learning algorithm, its probability of error is greater than $\frac{1}{2}$.

It seems unlikely that an algorithm could exploit consistency oracles which give the same answer on both those samples consistent with the target concept and those samples inconsistent with any concept in the class.

Definition 10.3 (Probabilistic Consistency Oracle): *A randomized algorithm $\mathcal{O}$ is a* probabilistic consistency oracle *for $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ if there exists polynomials p_1 and p_2 such that when $\mathcal{O}$ is given the parameters n and s together with a sample $S \in \mathrm{sam}_*(\mathbf{x})$ for any $\mathbf{x} \in (X_n)^m$ where $m = p_1(n,s)$, Oracle $\mathcal{O}$ answers "yes" with probability at least half when S is consistent with a concept in $\mathcal{F}_s$, and answers no with probability one on at least $2^m/p_2(m)$ of the 2^m samples in $\mathrm{sam}_*(\mathbf{x})$. Such an oracle is called* polynomial *if its answers are computed in time polynomial in n, s, and the total bit length of S.*

We will exploit the "one-sidedness" of probabilistic consistency oracles in the same way that "random polynomial time hypothesis finders" were exploited by Haussler et al. [HKLW91].

Theorem 10.4: *A polynomial weak learning algorithm for $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ can be used to construct a polynomial probabilistic consistency oracle.*

Proof: Let A be a polynomial weak learning algorithm for $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$. We assume that Algorithm A is deterministic as polynomially many random bits can be obtained from additional examples (see [HKLW91], Lemma 3.5). There exist three polynomials p_1, p_2, and p_3 such that 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 sample $\mathrm{sam}_f(\mathbf{x})$, where $\mathbf{x}$ is drawn according to D^m and $m = p_1(n,s)$, Algorithm A outputs a hypothesis $h = A[\mathrm{sam}_f(\mathbf{x})]$ on X_n for which

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

Furthermore, the hypothesis output by A are polynomially evaluatable and the running time of A is polynomial in the total length of its input, n, and s.

The results of Freund [Fre90] imply that there is a randomized Algorithm B which repeatedly uses the weak learning algorithm A to compress a large sample. When given n, s, and any sample $S \in \mathrm{sam}_*(\mathbf{x})$, where $\mathbf{x} \in (X_n)^{\hat{m}}$, Algorithm B outputs a sequence S' of $\frac{1}{2}p_1(n,s)(p_2(n,s))^2\ln(\hat{m})$ examples from S. The output, S', should be viewed as $\frac{1}{2}(p_2(n,s))^2\ln(\hat{m})$ blocks of $p_1(n,s)$ examples each, and represents the hypothesis $h(S')$ defined as the majority function on the $\frac{1}{2}(p_2(n,s))^2\ln(\hat{m})$ hypotheses produced when A is run on each block in turn. The hypotheses $h(S')$ have the property that if S is labeled consistently with some concept in $\mathcal{F}_s$, then $h(S')$ is consistent with S with probability at least half (over the randomization of B). The running time of B is polynomial in n, s, and the total bit length of S. Also, since the hypothesis output by A are polynomially evaluatable, one can check in polynomial time whether $h(S')$ is consistent with S.

For each $\mathbf{x}$ of length $\hat{m}$, the total number of compressed sequences S' containing $\frac{1}{2}p_1(n,s)(p_2(n,s))^2\ln(\hat{m})$ examples chosen from some sample $S \in \mathrm{sam}_*(\mathbf{x})$ is at most

$$\hat{m}^{\frac{1}{2}p_1(n,s)(p_2(n,s))^2\ln(\hat{m})} = 2^{\frac{1}{2\ln 2}p_1(n,s)(p_2(n,s))^2\ln^2(\hat{m})}.$$

The number of labelings of $\mathbf{x}$ is $2^{\hat{m}}$. Thus there are at least

$$2^{\hat{m}} - 2^{\frac{1}{2\ln 2}p_1(n,s)(p_2(n,s))^2\ln^2(\hat{m})}$$

samples $S \in \mathrm{sam}_*(\mathbf{x})$ for which there is no sequence S' representing a hypothesis consistent with the sample S. We now choose $\hat{m}$ moderately large (polynomial in n and s) so that the number of nonrepresented samples of $\mathrm{sam}_*(\mathbf{x})$ is at least a polynomial fraction of all $2^{\hat{m}}$ labelings.

So we can use B to construct a probabilistic consistency oracle: Construct a sequence S' using B and if the hypothesis represented by S' is consistent with the input sample S of length $\hat{m}$ then answer "yes" and otherwise answer "no". As this oracle answers no for all nonrepresented samples of $S \in \mathrm{sam}_*(\mathbf{x})$ and yes with probability at least half for all samples S consistent with a target, it is a probabilistic consistency oracle. □

Corollary 10.5: *A concept class $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ is polynomially weakly learnable if and only if there is a polynomial probabilistic consistency oracle for $\mathcal{F}$ on X.*

Proof: In view of Theorem 10.4 we only have to show that a weak learning algorithm can be constructed from a polynomial probabilistic consistency oracle $\mathcal{O}$. We first construct a new oracle, $\mathcal{O}_r$, as follows: apply oracle $\mathcal{O}$ a total of r times to the input sample and answer "yes" if any of the r calls to $\mathcal{O}$ returned "yes" and "no" otherwise. Clearly, $\mathcal{O}_r$ is a probabilistic consistency oracle with the more stringent property that if the input sample is consistent with a concept in $\mathcal{F}_s$, then the probability for answering "yes" is $1 - 2^{-r}$.

We say that $\mathcal{O}_r$ *fails* if when given an input sample consistent with a concept in $\mathcal{F}_s$ it answers "no". The failure probability of $\mathcal{O}_r$ is at most 2^{-r}. Under the assumption that $\mathcal{O}_r$ is not failing (i.e. $\mathcal{O}_r$ acts like a regular one-sided consistency oracle) the proof of Theorem 7.3 shows how $\widetilde{Q}$ and $\mathcal{O}_r$ can be used to get a weak learning algorithm whose probability of a mistake on the last instance is at most $\frac{1}{2} - \frac{1}{p(n,s)}$, for some polynomial p. By choosing $r = 1 + \lceil \lg(p(n,s)) \rceil$ the failure probability is at most $\frac{1}{2p(n,s)}$ and thus the probability of a mistake on the last instance without the assumption that $\mathcal{O}_r$ is failing is at most $\frac{1}{2} - \frac{1}{2p(n,s)}$. □

Note that there are other less restrictive definitions of polynomial probabilistic consistency oracles for which the above corollary would hold. We used a version that was well suited for the proof of Theorem 10.4.

11 A Characterization of Weak Learning Using Restricted Data Interpolators

In this section we characterize weak learning using certain "data interpolators" and discuss how they relate to weak Occam algorithms.

The randomized algorithm A is a *restricted data interpolator* for $\bigcup_s \mathcal{F}_s$ on $\bigcup_n X_n$ if there exist polynomials p_1 and p_2 such that the following holds for all $n, s \geq 1$, targets $f \in \mathcal{F}_s$, and $\mathbf{x} \in X_n^m$ for $m = p_1(n,s)$:

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

Although restricted data interpolators have a hypothesis cardinality constraint similar to that of weak Occam algorithms (see Section 4), the hypotheses class of a restricted data interpolator is allowed to depend on the particular instance sequence $\mathbf{x}$. Restricted data interpolators also have a probabilistic nature similar to the probabilistic consistency oracles of Section 10.

The hypotheses output by a restricted data interpolator using sample size $m = p_1(n,s)$ are *polynomially evaluatable* if there is an algorithm that (when given n, s, $\mathbf{x} \in X_n^m$, the representation of a hypothesis $h \in \mathcal{H}_{n,s,\mathbf{x}}$ and $x \in X_n$) can decide in time polynomial in n and s and the total bitlength of its input whether $x \in h$.

The hypotheses output by a restricted data interpolator using sample size $m = p_1(n, s)$ are *polynomially recognizable* if there is an algorithm that (when given n, s, $\mathbf{x} \in X_n^m$ and the representation of a hypothesis h) can decide in time polynomial in n and s and the total bitlength of its input whether $h \in \mathcal{H}_{n,s,\mathbf{x}}$.

A restricted data interpolator algorithm is called *polynomial* if:

- its running time is polynomial in n and s and the total bitlength of its input,
- its hypotheses are polynomially evaluatable, and
- its hypotheses are polynomially recognizable.

We now show that the existence of a polynomial restricted data interpolator for a class is a necessary and sufficient condition for the class to be polynomially weakly learnable.

Theorem 11.1: *A concept class $\mathcal{F} = \bigcup_s \mathcal{F}_s$ on $X = \bigcup_n X_n$ is polynomially weakly learnable if and only if there is a polynomial restricted data interpolator.*

Proof: Given a polynomial restricted data interpolator we can easily construct a probabilistic consistency oracle from it: the oracle says "yes" if the hypothesis produced by the polynomial restricted data interpolator on input n, s, $\mathbf{x} \in X_n^m$ is both consistent and lies in $\mathcal{H}_{n,s,\mathbf{x}}$. Consistency can be checked in polynomial time since the polynomial restricted data interpolator outputs polynomially evaluatable hypotheses. Membership in $\mathcal{H}_{n,s,\mathbf{x}}$ can be decided in polynomial time since the polynomial restricted data interpolator outputs polynomially recognizable hypotheses. Thus by Corollory 10.5 the existence of polynomial restricted data interpolators implies polynomial weak learning.

For the opposite direction we observe that the algorithm used in the proof of Corollory 10.5 is a polynomial restricted data interpolator. Its hypotheses are polynomially evaluatable and are represented by length bounded sequences of examples using instances from $\mathbf{x}$, and thus polynomially recognizable. □

As noted above, restricted data interpolators are a generalization of Occam algorithms as the restricted data interpolators are probabilistic and their hypothesis class can depend on the actual instances as well as n, s, and m. As far as we know the above theorem is the first characterizion of learning using generalized Occam-style algorithms. Previous characterizations of polynomial learnability by Occam algorithms were in terms of a specific hypothesis class used by the learning algorithms [BP92][HKLW91]. Our results place no restriction on the hypothesis class used by the Occam algorithm, other than being polynomially evaluatable and polynomially recognizable.

Recall that Theorem 4.2 shows that weak Occam algorithms are weak learning algorithms. The above theorem does not show the same for restricted data interpolators. In fact, the hypotheses produced by restricted data interpolators can be arbitrarily bad. Consider the concept class $\mathcal{F} = \{f_1, f_2, ..., f_{2^n}\}$ on $X = \{1, ..., 2^n\}$ where each $f_i = \{1, ..., i\}$. When given a sample S of size m, a restricted data interpolator for this class could output a hypotheses which labels only those instances labeled one in the sample with one, and labels everything else zero. We use restricted data interpolators in a more "sophisticated way," converting them into oracles which are used by the our query lookahead algorithms.

12 Conclusions and Directions for Further Research

We see two potential benefits from this line of research. First we hope that polynomial versions of the discussed one-sided consistency oracles and probabilistic consistency oracles can be found for concept classes that have not previously been known to be learnable.

Second, the interaction of our results with cryptography could be a promising direction. For example, there is no polynomial weak learning algorithm for DFAs given standard cryptographic assumptions [KV89]. More precisely, the concept class in this learning problem is $\mathcal{F} = \bigcup_s \mathcal{F}_s$ where $\mathcal{F}_s$ consists of all regular languages accepted by DFAs with at most s states, and the domain is $X = \bigcup_n X_n$ where X_n is the set of all words over $\{0,1\}$ of length at most n. Our results show that, under the same cryptographic assumptions, a polynomial probabilistic consistency oracle for $\mathcal{F}$ on X can not exist.

For any fixed sequence $\mathbf{x}$ of m instances, $|\mathrm{sam}_{\mathcal{F}_s}(\mathbf{x})|$ is at most $m^{O(s \log s)}$ [Sau72], since the VC dimension of $\mathcal{F}_s$ is $O(s \log s)$. When m is a polynomial in s of degree 2 or greater, then $|\mathrm{sam}_{\mathcal{F}_s}(\mathbf{x})|$ is much smaller than the 2^m samples in $\mathrm{sam}_*(\mathbf{x})$.

Yet (given the cryptographic assumptions) there is no polynomial-time algorithm that answers "yes" with probability at least half on all of the $m^{O(s \log s)}$ labelings consistent with DFAs of at most s states and "no" on at least $2^m/p_2(m)$ other labelings.

We show in Section 8 that the sample complexity of weak learning an arbitrary concept classes of VC dimension d (disreguarding computational considerations) is at most $2d - \Omega(\sqrt{d \lg d})$. It is shown in Goldman et al. [GKS90] that there are classes of VC dimension d where every weak learning algorithm requires at least $d - O(\log d)$ examples. We would like to see these bounds tightened. It is interesting to note that the natural dividing line of d samples lies above the best current lower bound and below the best current upper bound.

In another direction, if $|\mathrm{sam}_{\mathcal{F}}(\mathbf{x})| = 2^{|\mathbf{x}|-\alpha(\mathbf{x})}$ then by Corollary 6.6 the probability that Algorithm $\widetilde{Q}$ predicts incorrectly on the last instance (averaged over permutations of $\mathbf{x}$) is $\frac{1}{2} - \frac{1}{|\mathbf{x}|} + \frac{2^{-\alpha(\mathbf{x})}}{|\mathbf{x}|}$. Thus Algorithm $\widetilde{Q}$ does not effectively exploit large $\alpha(\mathbf{x})$. In contrast the probability that Algorithms $\widetilde{\mathrm{Gibbs}}$, $\widetilde{\mathrm{Bayes}}$, or $\widetilde{G}$ predict wrong on the last instance goes to 0 as $\alpha(\mathbf{x})$ goes to $|\mathbf{x}|$ (see Figure 9.3).

Although Algorithms $\widetilde{\mathrm{Gibbs}}$, $\widetilde{\mathrm{Bayes}}$, and $\widetilde{G}$ can not be implemented efficiently, they can be approximated by making many calls to a consistency oracle. We presented an approximation to $\widetilde{G}$ using this approach [HW92b]. However, this approximation to $\widetilde{G}$ uses $\Omega(\frac{2^{\alpha(\mathbf{x})}-1}{\alpha(\mathbf{x})})$ calls[14] to a consistency oracle. As $\alpha(\mathbf{x})$ goes to $|\mathbf{x}|$, the number of queries used grows exponentially in $\alpha(\mathbf{x})$. Furthermore, the probability that this algorithm predicts incorrectly fails to drop to 0 as $\alpha(\mathbf{x})$ goes to $|\mathbf{x}|$. Whether or not there exists an efficient (polynomial time) method whose probability of of predicting wrong on the last instance goes to 0 as $\alpha(\mathbf{x})$ goes to $|\mathbf{x}|$ is an open problem.

During our comparison of Algorithm $\widetilde{\mathrm{Gibbs}}$ with Algorithm $\widetilde{Q}$, we derived an improved bound on the expected total number of mistakes made by Algorithm $\widetilde{\mathrm{Gibbs}}$ (Theorem 9.3). Perhaps similar techniques will lead to better bounds on the Bayes and Information Gain prediction algorithms than those of inequalities 9.1 and 9.2, respectively.

When computational considerations are ignored the algorithm that minimizes

$$\sup_{f \in \mathcal{F}} \mathbf{E}_{\mathbf{y} \in U(\mathbf{x})} [\mathbf{M}(A, f, \mathbf{y})]$$

is the 1-inclusion graph algorithm of Haussler, Littlestone, and Warmuth [HLW] (as discussed in the Section 9). For that algorithm the supremum equals $\frac{\mathrm{maxdens}_{\mathcal{F}}(\mathbf{x})}{|\mathbf{x}|}$ where $\mathrm{maxdens}_{\mathcal{F}}(\mathbf{x})$ is the maximum density of any subgraph of the 1-inclusion graph with respect to $\mathcal{F}$ and $\mathbf{x}$. We know from Corollary 9.5 that $\mathrm{maxdens}_{\mathcal{F}}(\mathbf{x}) \leq \min\{d, \frac{1}{2} \lg |\mathrm{sam}_{\mathcal{F}}(\mathbf{x})|, \frac{1}{2} +$

[14] Fewer calls suffice for weak learning but result in poorer predictions.

$(|\text{sam}_{\mathcal{F}}(\mathbf{x})|/2^{|\mathbf{x}|}) - (1/|\mathbf{x}|)\}$, where d is the VC dimension of $\mathcal{F}$. However, this upper bound on $\text{maxdens}_{\mathcal{F}}(\mathbf{x})$ is not tight. Another open problem is to determine the best possible upper bound on $\text{maxdens}_{\mathcal{F}}(\mathbf{x})$ as a function of d and $|\text{sam}_{\mathcal{F}}(\mathbf{x})|$ (or possibly other statistics of the 1-inclusion graph with respect to $\mathcal{F}$ and $\mathbf{x}$).

In Sections 10 we show that a concept class has a polynomial weak learning algorithm if and only if the concept class has a polynomial probabilistic consistency oracle. Furthermore, Section 11 shows that a concept class has a polynomial weak learning algorithm if and only if the concept class has a polynomial restricted data interpolator. Are there other characterizations of polynomial learnability?

Finally, we conjecture that the counterexamples at the end of Section 4 can be strengthened to show the following.

> There is an Occam-style algorithm that uses sample size $m = p(n, s)$, where p is a polynomial, and a hypothesis class of size $2^m - c$, where c is a positive constant, that is not a weak learning algorithm.

The existence of such an Occam-style non-learner would strengthen our belief that if the $2^{m-1/p_2(n,s)}$ bound on the size of the hypothesis class in the definition of weak Occam algorithm is increased then weak Occam algorithms will no longer be weak learning algorithms. In other words, if an Occam-style algorithm does not compress a sample containing m bit labels down to a hypothesis from a polynomially evaluatable class that can be represented by $m - 1/p_2(n, s)$ bits, then combinatorial arguments alone cannot show that the Occam-style algorithm is a weak learning algorithm.

13 Acknowledgements

We are grateful to Lenny Pitt, Peter Frankle, Yoav Freund, David Haussler, Phil Long, and Phokian Kolaitis for valuable discussions and Hikoe Enomoto for decreasing the bound on size of the hypothesis class in the example following Theorem 4.2 by one.

References

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[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. *J. Comb. Theory*, 12:201–202, 1972.

[BP92] R. Board and L. Pitt. On the necessity of Occam algorithms. *Theoretical Computer Science*, 100:157–184, 1992.

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

[GKS90] Sally A. Goldman, Michael J. Kerns, and Robert E. Schapire. On the sample complexity of weak learning. In *Proceedings of the 1990 Workshop on Computational Learning Theory*, pages 217–231, San Mateo, CA, August 1990. Morgan Kaufmann.

[GT90] G. Gyorgyi and N. Tishby. Statistical theory of learning a rule. In K. Thuemann and R. Koeberle, editors, *Neural Networks and Spin Glasses*. World Scientific, 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*, San Mateo, CA, August 1991. Morgan Kaufmann. 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.

[HO91] D. Haussler and M. Opper. Calculation of the learning curve of Bayes optimal classification algorithm for learning a perceptron with noise. In *Proceedings of the 1991 Workshop on Computational Learning Theory*. Morgan Kaufmann, San Mateo, CA, August 1991.

[HS90] D. Hansel and H. Sompolinsky. Learning from examples in a single-layer neural network. *Europhys. Lett.*, 11(7):687–692, 1990.

[HW92a] D. Helmbold and M. K. Warmuth. On weak learning. In *Proceedings of the Third NEC Research Symposium on Computational Learning and Cognition*, 3600 University City Science Center, Philadelphia, PA 19104-2688, May 1992. SIAM.

[HW92b] D. Helmbold and M. K. Warmuth. Some weak learning results. In *Proceedings of the 1992 Workshop on Computational Learning Theory*. ACM, July 1992.

[Kha92] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. unpublished manuscript, 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, New York, May 1989. ACM. To appear in *Journal of the ACM*.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, 1977.

[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.

[STS90] H. Sompolinsky, N. Tishby, and H.S. Seung. Learning from examples in large neural networks. *Phys. Rev. Lett.*, 65:1683–1686, 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, San Mateo, CA, August 1990. Morgan Kaufmann.

A Bounds on the Information Gain Prediction Algorithm

Definition A.1: *For all $\mathcal{P}$ on $\mathcal{F}$, $S \in (X \times \{0,1\})^*$, $x \in X$ and $b \in \{0,1\}$, let the* information gain $I^{\mathcal{P}}$ *be defined as follows:*

$$I^{\mathcal{P}}(\langle S,(x,b)\rangle) := -lg\frac{V^{\mathcal{P}}(\langle S,(x,b)\rangle)}{V^{\mathcal{P}}(S)}.$$

Thus, the Information Gain Prediction Algorithm predicts b (see Figure 9.1) with probability

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

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

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

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

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

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

Since $V^{\mathcal{P}}(S) = V^{\mathcal{P}}(\langle S,(x,0)\rangle)+V^{\mathcal{P}}(\langle S,(x,1)\rangle) > 0$, 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 also holds when either $V^{\mathcal{P}}(\langle S,(x,0)\rangle) = 0$ or $V^{\mathcal{P}}(\langle S,(x,1)\rangle) = 0$, as then $I^{\mathcal{P}}(\langle S,(x,0)\rangle)+I^{\mathcal{P}}(\langle S,(x,1)\rangle) = \infty$.

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

Lemma A.3: *If $V^{\mathcal{P}}(\mathrm{sam}_f(\mathbf{x})) > 0$ then*

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

Proof:

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

□

We are now ready to bound the probability that Algorithm $G^{\mathcal{P}}$ predicts incorrectly. This bound is a special case ($\beta = 0$) of a bound proven by Vovk [Vov90] for a more general aggregating strategy.

Theorem A.4: *For all $\mathcal{P}$ on $\mathcal{F}$, $f \in \mathcal{F}$, and $\mathbf{x} \in X^*$:*

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

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

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

using Lemma A.2. By the additivity of information (Lemma A.3),

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

as desired. □

B A Simple Upper Bound on the Density of the 1-inclusion Graph

In Corollary 9.5 three upper bounds on $\mathrm{maxdens}_{\mathcal{F}}(\mathbf{x})$ are given. Here we give a simple inductive proof of the second bound.

Let X be an arbitrary set and $\mathcal{F}$ be any concept class on X. For any $m \geq 1$ and $\mathbf{x} \in X^m$, we construct an undirected graph called the *1-inclusion graph* of $\mathcal{F}$ with respect to $\mathbf{x}$, denoted by $G_{\mathcal{F}}(\mathbf{x})$ [Bon72, AHW87, HLW]. The vertices of $G_{\mathcal{F}}(\mathbf{x})$ are the samples of $\mathrm{sam}_{\mathcal{F}}(\mathbf{x})$ and there is an edge between two vertices S and S' if S and S' disagree on the label of exactly one of the instances appearing in $\mathbf{x}$, and that instance appears only once in $\mathbf{x}$.

The *density* of a graph is the number of vertices over the number of edges. The graph $G = (V, E)$ is a *subgraph* of $G' = (V', E')$ if $V \subseteq V'$ and $E \subseteq G'$. Let $\mathrm{maxdens}_{\mathcal{F}}(\mathbf{x})$ be the maximum density of any subgraph of $G_{\mathcal{F}}(\mathbf{x})$.

Theorem B.1: *For any non-empty subgraph of a Boolean hypercube with g vertices the number of edges is at most $\frac{g}{2} \lg g$ and thus $\mathrm{maxdens}_{\mathcal{F}}(\mathbf{x}) \leq \frac{1}{2} \lg |\mathrm{sam}_F(\mathbf{x})|$.*

Proof: Any subgraph of a 1-inclusion graph corresponds to a subgraph of a Boolean hypercube: Each instance in the sequence $\mathbf{x}$ is responsible for one dimension of the cube and the Boolean labeling of $\mathbf{x}$ are the coordinates of the cube. Thus the fact that $\mathrm{maxdens}_{\mathcal{F}}(\mathbf{x}) \leq \frac{1}{2} \lg |\mathrm{sam}_F(\mathbf{x})|$ follows from the number of edges being at most $\frac{g}{2} \lg g$ and that the number of vertices in $G_{\mathcal{F}}(\mathbf{x})$ is $|\mathrm{sam}_F(\mathbf{x})|$.

We prove that the number of edges in any subgraph of a boolean hypercube containing f vertices is at most $\frac{g}{2}\lg g$ by induction on the dimension of the hypercube. It clearly holds for the 1-dimensional hypercube and $g = 0, 1$ or 2. For the induction step split the hypercube of dimension d into two subcubes of dimension $d-1$. Let g_1 and g_2 be the number of vertices in the two subcubes. If either g_1 or g_2 is zero then the bound follows directly from the inductive hypothesis. Thus we assume $1 \leq g_1 \leq g_2$. Clearly the number of vertices crossing between the subcubes is at most g_1. Applying the inductive hypothesis, the total number of vertices is at most

$$\begin{aligned}
\frac{g_1 \lg g_1}{2} + \frac{g_2 \lg g_2}{2} + g_1 &= \frac{(g_1+g_2)\lg(g_1+g_2)}{2} - \frac{g_1}{2}\lg\frac{(g_1+g_2)}{g_1} - \frac{g_2}{2}\lg\frac{(g_1+g_2)}{g_2} + g_1 \\
&\leq \frac{g\lg g}{2} - \frac{g_1}{2}\lg\frac{(g_1+g_2)^2}{g_1 g_2} + g_1 \\
&= \frac{g\lg g}{2} - \frac{g_1}{2}\lg(\frac{g_1}{g_2} + \frac{g_2}{g_1} + 2) + g_1 \\
&\leq \frac{g\lg g}{2} - \frac{g_1}{2}\lg(4) + g_1, \quad \text{since } x + (1/x) \geq 2 \text{ when } x > 0, \\
&= \frac{g\lg g}{2},
\end{aligned}$$

which completes the proof. □