Lower Bounds for Learning Discrete Distributions

Rocco A. Servedio*
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138
rocco@deas.harvard.edu

Abstract

We study the model of learning discrete probability distributions which was introduced by Kearns, Mansour, Ron, Rubinfeld, Schapire, and Sellie in [13]. Our main results are a lower bound on computational sample complexity and a lower bound on hypothesis size for this learning model.

We define a class of efficiently learnable distributions which has the following interesting property: while a computationally unbounded learning algorithm can learn the class from $O(1)$ examples, the *computational* sample complexity of the class is essentially $\Omega(1/\epsilon)$ in the sense that any polynomial-time learning algorithm must use at least this many examples. The construction is based on the existence of any length-preserving one-way permutation.

We also prove a lower bound on hypothesis size. Based on the existence of any one-way function, we construct a class of probability distributions learnable in polynomial time from $\tilde{O}(1/\epsilon^2)$ examples and prove that any polynomial-time learning algorithm for this class must use hypotheses of size essentially $\Omega(1/\epsilon^{1/2})$. This answers a conjecture posed by Kearns et al. in [13].

Note: After completing a draft of this paper, it was brought to our attention that Moni Naor [16] had already established, via a slightly different construction, results which are essentially equivalent to our main result of Section 5 (the lower bound on hypothesis size). We thank the unknown referee who brought [16] to our attention.

*Supported in part by an NSF Graduate Fellowship and by NSF grant CCR-95-04436.

1 Introduction

The PAC model of concept learning introduced by Valiant [21] has become one of the most popular models used in the computational study of supervised learning. A rich theory has been established within this framework which provides significant insight into the nature of efficient supervised learning. Kearns, Mansour, Ron, Rubinfeld, Schapire, and Sellie [13] have proposed a natural unsupervised variant of the standard PAC model, which we call the *discrete distribution learning model.* In this paper we continue the study of the discrete distribution learning model by proving lower bounds on sample complexity, computational sample complexity, and hypothesis size.

1.1 Background

As stated in [13], the discrete distribution learning model is inspired by the PAC (Probably Approximately Correct) model of concept learning. While the PAC model is used to study the learnability of classes of *Boolean functions*, the discrete distribution model is used to study the learnability of classes of *discrete probability distributions.* The learner in this model is given access to an oracle which outputs independent draws from an unknown probability distribution over $\{0,1\}^n$, and the task of the learner is to construct an approximation to the unknown distribution which is generating the draws. A full description of the discrete distribution learning model is given in Section 2.

As in the PAC model, the broad goal of research in the discrete distribution learning model is to study the relationship between the computational complexity of learning algorithms and the computational complexity of the objects (distributions) which are being learned. This focus on complexity is the chief point of departure between the model of learning discrete distributions and the substantial body of work in the statistics and pattern recognition literature on learning unknown distributions from random draws (see, e.g., [6, 12, 22]). The complexity of a distribution D over $\{0,1\}^n$ is measured in [13] by the size of an $n-$output circuit which transforms a source of truly random bits into the distribution D. All of the distributions considered in [13] and in this paper have polynomial-size circuits in this sense. Kearns et al. [13] gave polynomial-time learning algorithms for several simple classes of distributions over $\{0,1\}^n$ and proved computational hardness results for learning other, more complex distribution classes. Further results on the discrete distribution learning model were given in [7, 17, 18].

Kearns et al. closed their paper with an intriguing open question about the discrete distribution learning model. The principle that "compression implies learnability," commonly referred to as Occam's Razor, is a fundamental property of many learning models. Blumer et al. [2] established this property for the PAC model by showing that any efficient algorithm which can find a short hypothesis that correctly labels a large random sample is in fact an efficient PAC learning algorithm (such a hypothesis can be viewed as a compressed version of the string of example labels for the sample). In a sequence of well-known results, Schapire [19] and Freund [8] established a strong converse to Occam's Razor for the PAC model; their results can roughly be rephrased as "PAC learnability implies extreme compression." More precisely, they proved that if a concept class is polynomial-time learnable then it is polynomial-time learnable by an algorithm which uses hypotheses of size $O(\log 1/\epsilon)$, where ϵ is the desired accuracy level. Since PAC learning any nontrivial concept class to accuracy ϵ is known to require $\Omega(1/\epsilon)$ examples [3], these results show that any efficiently solvable PAC learning problem is in fact efficiently solvable using a hypothesis which is much smaller (as a function of ϵ) than the sample used to generate it.

Kearns et al. conjectured in [13] that no analogous strong converse to Occam's Razor holds in the discrete distribution learning model. More specifically, they defined a particular class of distributions which are efficiently learnable from $\text{poly}(1/\epsilon)$ examples and conjectured that if determining

quadratic residuosity modulo a composite is computationally hard, then any polynomial-time learning algorithm for this distribution class must use hypotheses of size $\Omega(1/\epsilon^{\alpha})$ for some $\alpha > 0$.

1.2 Our Contributions

We establish lower bounds on the resources required for learning discrete distributions. The resources which we consider are sample complexity (the number of draws from the distribution which the learning algorithm makes), computational sample complexity, and hypothesis size. Throughout the paper we restrict our attention to distribution classes which are generated by polynomial-size circuits and are learnable in polynomial time.

As a warmup for our later results, in Section 3 we give a lower bound on sample complexity. We describe a simple class of efficiently learnable distributions and prove that any algorithm which learns this class to accuracy ϵ must use $\Omega(1/\epsilon)$ examples. This lower bound is information-theoretic and holds for all learning algorithms regardless of their computational power.

In Section 4 we modify the construction of Section 3 to obtain a class of efficiently learnable distributions which has the following interesting property: while a computationally unbounded learning algorithm can learn the class from $O(1)$ examples, the *computational* sample complexity of the class is essentially $\Omega(1/\epsilon)$ in the sense that any polynomial-time learning algorithm must use at least this many examples. The construction is based on the existence of any length-preserving one-way permutation. This result demonstrates that as in the PAC model [5, 20], a strong tradeoff can exist in the discrete distribution learning model between the amount of computation time and the number of examples which a learning algorithm requires.

In Section 5 we prove that there is no strong converse to Occam's Razor for learning discrete distributions. We construct a class of distributions learnable in polynomial time from $\tilde{O}(1/\epsilon^2)$ examples and prove that any polynomial-time learning algorithm for this class must use hypotheses of size essentially $\Omega(1/\epsilon^{1/2})$. Although we do not address the specific class of distributions which was conjectured to require large hypotheses in [13], our construction has the advantage that it depends only on a general cryptographic hardness assumption (the existence of any one-way function). As discussed in Section 6, the existence of such a class of distributions is particularly interesting in light of the fact that many commonly used algorithms for learning an unknown distribution work by "memorizing" the sample they are given.

2 Preliminaries

In this section we describe the discrete distribution learning model introduced by Kearns et al. in [13]. The reader who is familiar with the PAC model of concept learning as described in, e.g., [1, 14] will notice many similarities between the two models.

Throughout this paper the size parameter n denotes an arbitrary positive integer. We write $\mathcal{D}_n$ to denote a class of probability distributions over $\{0,1\}^n$. For $D \in \mathcal{D}_n$ and $x \in \{0,1\}^n$ we write $D[x]$ to denote the probability weight which distribution D assigns to x; similarly for $X \subseteq \{0,1\}^n$ we write $D[X]$ to denote $\sum_{x \in X} D[x]$.

As described in [13] there are two natural representations for probability distributions over $\{0,1\}^n$. A *generator* for a distribution D over $\{0,1\}^n$ is an n-output circuit which, when provided with a uniformly chosen random input, outputs a random draw from D. More precisely, we have

Definition 1 *Let $\mathcal{D}_n$ be a class of distributions over $\{0,1\}^n$. We say that $\mathcal{D}_n$ has* polynomial-size generators *if there are polynomials $p(\cdot)$ and $r(\cdot)$ such that for any $n \geq 1$ and any distribution $D \in \mathcal{D}_n$, there is a circuit G_D of size at most $p(n)$, with $r(n)$ input bits and n output bits, such that*

if a string z is drawn from the uniform distribution on $\{0,1\}^{r(n)}$ then D is the distribution over $\{0,1\}^n$ induced by applying G_D to z. Such a circuit G_D is called a generator *for distribution D.*

We will only consider classes of distributions $\mathcal{D}_n$ which have polynomial-size generators.

Another natural representation for a distribution is an *evaluator,* which is a circuit that computes the probability weight which the distribution assigns to any point.

Definition 2 *Let $\mathcal{D}_n$ be a class of distributions over $\{0,1\}^n$. We say that $\mathcal{D}_n$ has* polynomial-size evaluators *if there is a polynomial $p(\cdot)$ such that for any $n \geq 1$ and any distribution $D \in \mathcal{D}_n$, there is an n-input circuit E_D of size at most $p(n)$ which outputs the binary representation of $D[x]$ when given $x \in \{0,1\}^n$ as its input. Such a circuit E_D is called an* evaluator *for distribution D.*

As in [13] we use the *Kullback-Liebler divergence* (also known as *relative entropy*) as our distance measure for distributions. The KL-divergence between distributions D_1 and D_2 over $\{0,1\}^n$ is

$$KL(D_1||D_2) = \sum_{x \in \{0,1\}^n} D_1[x] \log \frac{D_1[x]}{D_2[x]}.$$

(Here and throughout the paper "log" means log base 2 and "ln" means log base e.) The KL-divergence is widely used in information theory and has many nice properties: for instance, $KL(D_1||D_2)$ is always nonnegative and is 0 if and only if D_1 and D_2 are identical. The KL-divergence also bounds the standard L_1 distance [4]: for any two distributions D_1, D_2 over $\{0,1\}^n$ we have $KL(D_1||D_2) \geq (L_1(D_1, D_2))^2 / 2 \ln 2$, where $L_1(D_1, D_2) = \sum_{x \in \{0,1\}^n} |D_1(x) - D_2(x)|$.

Definition 3 *Let D be a distribution over $\{0,1\}^n$ and let G be a circuit with n output bits. We say that G is an* ϵ-good generator for D *if $KL(D||D_G) \leq \epsilon$, where D_G is the distribution over $\{0,1\}^n$ induced by providing uniform random input bits to G. If E is a circuit with n input bits, then we say that E is an* ϵ-good evaluator *for D if $KL(D||D_E) \leq \epsilon$, where D_E is the distribution over $\{0,1\}^n$ which assigns weight $E[x]$ to each string $x \in \{0,1\}^n$.*

For a distribution D over $\{0,1\}^n$ we let $GEN(D)$ denote an oracle which, when invoked, runs in unit time and returns an example $x \in \{0,1\}^n$ which is distributed according to D.

Definition 4 *Let $\mathcal{D}_n$ be a class of distributions over $\{0,1\}^n$. An algorithm A is said to* learn $\mathcal{D}_n$ with a generator (evaluator) *if (i) algorithm A takes as input a parameter $\epsilon > 0$ and has oracle access to $GEN(D)$, where the unknown target distribution D can be any distribution belonging to $\mathcal{D}_n$; and (ii) for any $D \in \mathcal{D}_n$ algorithm A outputs a circuit G (E) that with probability at least $9/10$ is an ϵ-good generator (evaluator) for D. We say that A is* efficient *if there is a fixed polynomial $p(\cdot,\cdot)$ such that for all $D \in \mathcal{D}_n$ algorithm A runs in time bounded by $p(n, 1/\epsilon)$.*

Although our chief interest is efficient learning, we will also have occasion to consider computationally unbounded learning algorithms which can have arbitrary (finite) runtimes. Note also that in the above definition we use the fixed confidence parameter $9/10$; this simplifies our presentation without significantly affecting any results.

Besides time complexity we will also be interested in the number of examples used by learning algorithms and the size of the hypotheses (generators or evaluators) they construct. Given values of n and ϵ, the *sample complexity* $m(n,\epsilon)$ of a learning algorithm A is the maximum number of calls (across all $D \in \mathcal{D}_n$) to $GEN(D)$ which algorithm A ever makes. For hypothesis size, fix some reasonable encoding scheme $\mathcal{S}$ which maps Boolean circuits to binary strings. We say that the size of a circuit C is $|\mathcal{S}(C)|$, i.e. the bitlength of the description of C; note that there can be at most 2^{k+1} circuits of size at most k. The *hypothesis size* $h(n,\epsilon)$ of a learning algorithm A is the maximum size (across all $D \in \mathcal{D}_n$) of any circuit which A ever outputs as its hypothesis.

3 A Lower Bound on Sample Complexity

In this section we present a simple class $\mathcal{US}_{n+1}$ of distributions and show that any successful learning algorithm for this class must use $\Omega(1/\epsilon)$ examples. The proof is information-theoretic and does not depend on any unproven computational hardness assumptions.

Given a string $s = s_1 \ldots s_n \in \{0,1\}^n$, let US_s be the distribution over $\{0,1\}^{n+1}$ defined by

- $US_s[1^i 0^{n-i} s_i] = 1/2^i \quad$ for $i \in \{1, \ldots, n-1\}$,
- $US_s[1^n s_n] = 1/2^{n-1}$,
- $US_s[x] = 0 \quad$ for all other strings $x \in \{0,1\}^{n+1}$.

The distribution US_s is thus completely characterized by the unknown string s (hence the name US_s), and a draw from US_s identifies one bit of s. However, draws corresponding to successive bits of s are assigned exponentially decreasing weights by US_s (except for the draw corresponding to the last bit s_n which has extra weight to make the sum of the probabilities equal 1). It is clear that the class $\mathcal{US}_{n+1} = \{US_s : s \in \{0,1\}^n\}$ has polynomial-size generators.

Theorem 5 *Let $\mathcal{US}_{n+1} = \{US_s : s \in \{0,1\}^n\}$. Then*

1. *Distribution class $\mathcal{US}_{n+1}$ can be learned with either a generator or an evaluator by a polynomial-time algorithm which has sample complexity $O(1/\epsilon)$.*
2. *Any algorithm for learning the class $\mathcal{US}_{n+1}$ with either a generator or an evaluator must have sample complexity $\Omega(1/\epsilon)$.*

Proof: For part (1) we show that a simple memorization-based algorithm achieves sample complexity $O(1/\epsilon)$. Let US_s be the unknown target distribution; for $t \in \{0, \ldots, n\}$ let s^t denote the n-bit string $s_1 \ldots s_t \overline{s}_{t+1} \ldots \overline{s}_n$ which disagrees with s in the last $n-t$ positions. A straightforward calculation shows that making $O(2^t)$ calls to $GEN(US_s)$ will reveal all of the bits $s_1, \ldots, s_t$ with probability at least $9/10$. Given only the t-bit prefix $s_1, \ldots, s_t$ of string s, let U_s^t be the distribution which is identical to U_s on the first t bits and splits its weight on the last $n-t$ bits, i.e. $U_s^t = (U_s + U_{s^t})/2$. It is easy to construct a generator (evaluator) for US_s^t from the bits $s_1, \ldots, s_t$. Finally, since $KL(US_s || US_s^t) = 1/2^t$, taking $t = \lceil \log 1/\epsilon \rceil$ proves part (1).

For part (2) the proof is by contradiction: Suppose that A' is an algorithm for learning $\mathcal{US}_{n+1}$ with a generator which has sample complexity $m < 1/4\epsilon$. For $i \in \{1, \ldots, n\}$ define the set $X_i \subset \{0,1\}^{n+1}$ to be $X_i = \{1^j 0^{n-j} s_j\}_{j=i}^n$, so $US_s[X_i] = 1/2^{i-1}$. Let $S = x^1, \ldots, x^m$ be the sample of draws which algorithm A' receives from its m calls to the oracle $GEN(US_s)$, and let G be the generator which A' outputs. We have that $\Pr[S \cap X_{k+1} = \emptyset] = (1 - 1/2^k)^m$, which is at least $1/2$ for $k = \lceil \log(2m) \rceil$. Hence with probability at least $1/2$ algorithm A' receives no information about the $(n-k)$-bit suffix $s_{k+1} \ldots s_n$. The following simple claim is proved in Appendix A:

Claim 6 *Let D_1, D_2 be distributions over a finite set. Then the distribution D which minimizes the quantity $KL(D_1||D) + KL(D_2||D)$ is $D = (D_1 + D_2)/2$.*

Applying the claim and using the fact that $US_s^k = (US_s + US_{s^k})/2$, we find that

$$KL(US_s||D_G) + KL(US_{s^k}||D_G) \geq KL(US_s||US_s^k) + KL(US_{s^k}||US_s^k) \geq 1/2^{k-1},$$

so either $KL(US_s||D_G) \geq 1/2^k$ or $KL(US_{s^k}||D_G) \geq 1/2^k$. Consequently, if s is chosen uniformly at random from $\{0,1\}^n$, then with overall probability at least $1/4$ we have that $KL(US_s||D_G) \geq 1/2^k > 1/4m > \epsilon$, which contradicts the fact that A' is a successful algorithm for learning $\mathcal{US}_{n+1}$ with a generator. An identical argument can be used for learning with an evaluator. ■

4 Computational Sample Complexity

In this section we describe a class $\mathcal{CSC}_{2n+1}$ of distributions over $\{0,1\}^{2n+1}$ which has the following interesting property: there is a computationally unbounded learning algorithm for $\mathcal{CSC}_{2n+1}$ which has sample complexity 1 (it needs only a single draw from the target distribution), but any polynomial-time learning algorithm for $\mathcal{CSC}_{2n+1}$ has sample complexity $\Omega(1/\epsilon^{1-c})$ for any constant $c > 0$. We thus say that this class of distributions has *computational sample complexity* $\Omega(1/\epsilon^{1-c})$. The existence of such a class of distributions demonstrates that as in the PAC model [5, 20], in the model of learning discrete distributions a strong tradeoff can exist between the amount of computation time and the number of examples which a learning algorithm requires.

The construction requires some facts and definitions from cryptography. We write $x \in D_n$ to indicate that x is drawn from distribution D_n over $\{0,1\}^n$ and write U_n to denote the uniform distribution over $\{0,1\}^n$.

Definition 7 *A* length-preserving one-way permutation f *is a permutation* $f : \{0,1\}^* \to \{0,1\}^*$ *which satisfies* $|f(x)| = |x|$ *for all* $x \in \{0,1\}^*$ *as well as the following conditions:*

- *(efficient evaluation) There is a deterministic algorithm which, given an argument* x*, runs in* $poly(|x|)$ *time and computes* $f(x)$*;*
- *(hard to invert) For all probabilistic polynomial-time algorithms* A*, for all polynomials* Q*, for all sufficiently large* n*, we have* $\Pr_{x \in U_n}[A(f(x)) = x] < \frac{1}{Q(n)}$.

Definition 8 *Two sequences of probability distributions* $\{X_n\}$ *and* $\{Y_n\}$ *over* $\{0,1\}^n$ *are* polynomial-time indistinguishable *if for all probabilistic polynomial-time decision algorithms* A*, for all polynomials* Q*, for all sufficiently large* n*, we have* $|\Pr_{z \in X_n}[A(z) = 1] - \Pr_{z \in Y_n}[A(z) = 1]| < \frac{1}{Q(n)}$.

One can think of the decision algorithm A in the above definition as outputing 1 if it thinks its input was drawn from X_n and outputing 0 if it thinks its input was drawn from Y_n.

The following fact is a consequence of Proposition 3.17 in [9].

Fact 9 *Let* f *be a length-preserving one-way permutation. Then there is a deterministic polynomial-time algorithm* $PRG : \{0,1\}^n \to \{0,1\}^n$ *such that the two sequences of distributions over* $\{0,1\}^{2n}$ $\{f^n(z)PRG(z)\}_{z \in U_n}$ *and* $\{f^n(z)w\}_{z \in U_n, w \in U_n}$ *are polynomial-time indistinguishable, where* $f^n(z)$ *denotes* f *applied repeatedly* n *times to* z*.*

Intuitively, this fact implies that for $z \in U_n$ the string $PRG(z)$ "looks random" to any polynomial-time algorithm even if the algorithm is given $f^n(z)$. Of course, a computationally unbounded algorithm which is given $f^n(z)$ can invert the permutation f^n and discover z.

4.1 Computational Sample Complexity for Learning Discrete Distributions

Let f be a length-preserving one-way permutation and let $PRG : \{0,1\}^n \to \{0,1\}^n$ be the deterministic poly-time algorithm whose existence is asserted in Fact 9. We write $PRG(s)_i$ to denote the ith bit of $PRG(s)$. Given any string $s = s_1 \ldots s_n \in \{0,1\}^n$, let CSC_s be the distribution over $\{0,1\}^{2n+1}$ which is defined as follows:

- $CSC_s[f^n(s)1^i 0^{n-i} PRG(s)_i] = 1/2^i \quad$ for $i \in \{1, \ldots, n-1\}$,
- $CSC_s[f^n(s)1^n PRG(s)_n] = 1/2^{n-1}$,

- $CSC_s[x] = 0$ for all other strings $x \in \{0,1\}^{2n+1}$.

It is clear that the class $\mathcal{CSC}_{2n+1} = \{CSC_s : s \in \{0,1\}^n\}$ has polynomial-size generators.

The idea behind our construction is simple: given a single example, a computationally unbounded learner can repeatedly invert f to obtain s from the prefix $f^n(s)$. However, Fact 9 implies that for any polynomial-time learning algorithm this prefix contains no useful information, and hence (as in Theorem 5) any polynomial-time algorithm needs essentially $1/\epsilon$ examples to learn to accuracy ϵ. We have

Theorem 10 *If length-preserving one-way permutations exist, then*

1. *Distribution class $\mathcal{CSC}_{2n+1}$ can be learned with either a generator or an evaluator by a polynomial-time algorithm which has sample complexity $O(1/\epsilon)$.*
2. *There is a computationally unbounded algorithm for learning $\mathcal{CSC}_{2n+1}$ (with either a generator or an evaluator) which has sample complexity 1.*
3. *Any polynomial-time algorithm which learns $\mathcal{CSC}_{2n+1}$ with either a generator or an evaluator must have sample complexity $\Omega(1/\epsilon^{1-c})$ for any constant $0 < c \leq 1$.*

Proof: The proof of part (1) is entirely similar to the proof of part (1) of Theorem 5. Part (2) is also straightforward: a single draw from the target distribution CSC_s yields a string of the form $f^n(s)1^i0^{n-i}PRG(s)_i$ for some $i \in \{1,\ldots,n\}$. Since f is a permutation, f^n is also a permutation, so a computationally unbounded algorithm can invert $f^n(s)$ and obtain the string s.

The proof of part (3) is by contradiction; so suppose first that A' is a polynomial-time algorithm for learning $\mathcal{CSC}_{2n+1}$ with a generator which uses $m \leq n^d/\epsilon^{1-c}$ examples, where c, d are nonnegative absolute constants and $0 < c \leq 1$. Let $k = 1 + \lceil (1 + d/c - c)\log n \rceil$. The following algorithm, which we call algorithm B, takes as input the string $f^n(s)$ and the first k bits of the n-bit string $PRG(s)$, runs in poly(n) time, and outputs a single bit. (The idea is that using algorithm A' as a subroutine, algorithm B succeeds in guessing the bit $PRG(s)_{k+1}$ with probability significantly better than $1/2$.)

Algorithm B:

1. Run learning algorithm A' with ϵ set to $1/n^{1+d/c}$, using the k known bits of $PRG(s)$ to simulate the oracle $GEN(CSC_s)$. If the simulation fails (i.e. a draw comes up $f^n(s)1^t0^{n-t}PRG(s)_t$ for some $t > k$), abort and output a random bit. Otherwise, denote by G the generator which A' outputs as its hypothesis, and let r be the number of inputs to G.
2. For $i = 1,\ldots,T = 224 \cdot 2^{k+1}$ let x^i be a uniformly chosen r-bit string. For $b \in \{0,1\}$ let

$$\hat{p}_b = \frac{|\{i \in \{1,\ldots,T\} : G(x^i) = f^n(s)1^{k+1}0^{n-k-1}b\}|}{T}.$$

   Output the bit b such that $\hat{p}_b \geq \hat{p}_{\overline{b}}$ (breaking ties randomly).

Comment on Step 1: Since algorithm A' is being run with $\epsilon = 1/n^{1+d/c}$, we have that $m \leq n^d/\epsilon^{1-c} = n^{1+d/c-c}$. Under distribution CSC_s, draws corresponding to the first k bits of $PRG(s)$ have total probability $1 - 1/2^k$, so the probability that all m simulated draws are performed successfully is

$$\left(1 - \frac{1}{2^k}\right)^m \geq \left(1 - \frac{1}{2n^{1+d/c-c}}\right)^{n^{1+d/c-c}} \geq 1/2,$$

where the last inequality holds because $n^{1+d/c-c} \geq 1$ and $(1-\frac{1}{2t})^t \geq 1/2$ for $t \geq 1$. Hence, with probability at least $1/2$ algorithm B reaches step 2.

Comment on Step 2: For $b \in \{0,1\}$ let $p_b = D_G[f^n(s)1^{k+1}0^{n-k-1}b]$; so p_b is the true probability of the string $f^n(s)1^{k+1}0^{n-k-1}b$ under the distribution generated by G, and $\hat{p}_b$ is an estimate of this probability obtained by sampling. Let α denote the bit $PRG(s)_{k+1}$. Since A' is a learning algorithm, we have that G is an ϵ-good generator for CSC_s with probability at least $9/10$. The following claim can be proved in a manner entirely analogous to Claim 16:

Claim 11 *If D_G is ϵ-good (i.e. $KL(CSC_s, D_G) \leq \epsilon$) then $p_\alpha \geq \frac{7}{8} \cdot \frac{1}{2^{k+1}}$ and $p_{\overline{\alpha}} \leq \frac{1}{8} \cdot \frac{1}{2^{k+1}}$.*

A straightforward application of Chernoff bounds ([14], p. 190) now shows that

$$\Pr\left[\hat{p}_\alpha \geq \frac{3}{4} \cdot \frac{1}{2^{k+1}}\right] \geq 1 - e^{-2} \qquad \text{and} \qquad \Pr\left[\hat{p}_{\overline{\alpha}} \leq \frac{1}{4} \cdot \frac{1}{2^{k+1}}\right] \geq 1 - e^{-2}$$

(this motivated our choice of T). Hence, conditioned on the fact that G is ϵ-good, we have that $\Pr[B \text{ outputs } \alpha] \geq 1 - 2e^{-2}$. Consequently, assuming that algorithm B reaches step 2, we have

$$\begin{aligned} \Pr[B \text{ outputs } \alpha] &\geq \Pr[G \text{ is } \epsilon\text{-good}] \cdot \Pr[B \text{ outputs } \alpha \mid G \text{ is } \epsilon\text{-good}] \\ &\geq (9/10)(1 - 2e^{-2}). \end{aligned}$$

Since algorithm B reaches step 2 with probability at least $1/2$, the overall probability that algorithm B outputs α is at least $(1/2)(9/10)(1-2e^{-2}) + (1/2)(1/2) > 11/20$.

Thus, on input $f^n(s)$ and $PRG(s)_1 \ldots PRG(s)_k$, algorithm B outputs $PRG(s)_{k+1}$ with probability at least $11/20$. This is true for any $s \in \{0,1\}^n$, so it is true with overall probability at least $11/20$ for $s \in U_n$. However, on input $f^n(s)$ and $w_1 \ldots w_k$ where $s \in U_n$ and $w \in U_n$, algorithm B will output the unknown bit w_{k+1} with probability exactly $1/2$. This gap means that algorithm B can be used to construct a polynomial-time distinguisher for the distributions $\{f^n(s)PRG(s)\}_{s \in U_n}$ and $\{f^n(s)w\}_{s \in U_n, w \in U_n}$ as follows: run algorithm B on the first $n+k$ bits of the input string and output 1 if its answer equals the $(n+k+1)$st bit, otherwise output 0. Thus, the existence of algorithm B violates the polynomial-time indistinguishability which is asserted in Fact 9; this contradiction proves part (3) for the case of learning with a generator.

The proof of part (3) for learning with an evaluator is similar but slightly simpler. Algorithm B now outputs an evaluator, which we denote by E, as its hypothesis in step 1. The evaluator E can be used to find $p_b = D_E[f^n(s)1^{k+1}0^{n-k-1}b]$ exactly in step 2 without sampling; the output of step 2 is the bit b which maximizes p_b. Claim 11 now implies that $\Pr[B \text{ outputs } \alpha \mid E \text{ is } \epsilon\text{-good}] = 1$, and the proof goes through as before. ■

5 No Strong Converse to Occam's Razor

In this section we prove our main result, that there is no strong converse to Occam's Razor for learning discrete distributions. We do this by exhibiting a class $\mathcal{PF}_n$ of distributions which is learnable by a polynomial-time algorithm but is such that any polynomial-time learning algorithm must have hypothesis size $\Omega(1/\epsilon^{1/2-c})$ for any constant $c > 0$.

Roughly speaking, each distribution in the classes $\mathcal{US}_{n+1}$ and $\mathcal{CSC}_{2n+1}$ which we have considered so far is characterized by only n bits of uncertainty (the $n-$bit string s for a distribution US_s and the $n-$bit string $PRG(s)$ for a distribution CSC_s). Furthermore, these distributions have the property

that adding one more "known bit" halves the error rate; as a consequence, these distribution classes are learnable by algorithms whose hypothesis size depends only logarithmically on $1/\epsilon$.

Now we want a class of distributions where the required hypothesis size depends *polynomially* on $1/\epsilon$; hence we want it to be the case that the number of known bits must be essentially *doubled* in order to halve the error rate. Thus, instead of having each distribution contain only a linear number of unknown bits, we now want each distribution to contain an exponential amount of uncertainty (as far as any polynomial-time learning algorithm is concerned). We also, of course, want each distribution in the class to have a polynomial-size generator. We simultaneously achieve these two objectives by using pseudorandom functions in combination with a novel weighting scheme.

In the following definition, which is adapted from [10], the notation $M^g(1^n)$ denotes the result of running oracle machine M on the vacuous input string 1^n using an oracle for the function g (so if M is a polynomial-time machine then $M^g(1^n)$ runs for at most poly(n) steps).

Definition 12 *A* pseudorandom function family *is a collection of functions* $\{f_s\}_{s\in\{0,1\}^*}$ *where each* f_s *maps* $\{0,1\}^{|s|} \to \{0,1\}$ *such that:*

- *(efficient evaluation) There is a deterministic algorithm which, given a seed* $s \in \{0,1\}^n$ *and an argument* $x \in \{0,1\}^n$*, runs in time poly*(n) *and returns the value* $f_s(x)$*.*
- *(pseudorandomness) For all probabilistic poly-time oracle machines* M*, for all polynomials* Q*, for all sufficiently large* n*, we have* $|\Pr_{s\in U_n}[M^{f_s}(1^n) = 1] - \Pr_{F\in\mathcal{F}_n}[M^F(1^n) = 1]| < \frac{1}{Q(n)}$ *where* $\mathcal{F}_n$ *is the uniform distribution over all* 2^{2^n} *functions mapping* $\{0,1\}^n \to \{0,1\}$*.*

Intuitively, for a randomly chosen seed s, the function f_s "looks like" a truly random function to any polynomial-time algorithm which has oracle access to f_s. It is shown in [11] that a pseudorandom function family exists if one-way functions exist.

5.1 No Converse to Occam's Razor

Given a number $0 \le j \le 2^n - 1$, we write $j_\sharp$ to indicate the n-bit string which is the binary representation of j. For any function $g : \{0,1\}^n \to \{0,1\}$, we define the distribution $DIST_g$ over $\{0,1\}^{n+1}$ as follows:

- $DIST_g[j_\sharp g(j_\sharp)] = 1/2^{2i-1}$ for j such that $2^{i-1} - 1 \le j < 2^i - 1$, for each $i \in \{1,\ldots,n-1\}$,
- $DIST_g[j_\sharp g(j_\sharp)] = 1/2^{2n-2}$ for $2^{n-1} - 1 \le j < 2^n - 1$,
- $DIST_g[x] = 0$ for all other strings $x \in \{0,1\}^{n+1}$.

So the distribution $DIST_g$ puts weight $1/2$ on the single point $0_\sharp g(0_\sharp)$, puts weight $1/8$ on each of the points $1_\sharp g(1_\sharp)$ and $2_\sharp g(2_\sharp)$, puts weight $1/32$ on each of the four points $3_\sharp g(3_\sharp),\ldots,6_\sharp g(6_\sharp)$, and so on. In general there are 2^{i-1} points of weight $1/2^{2i-1}$; the only exception is when $i = n$ since we need the sum of the probabilities to be 1.

Now let $\{f_s\}_{s\in\{0,1\}^*}$ be a pseudorandom function family, so each n-bit string s defines a function $f_s : \{0,1\}^n \to \{0,1\}$. We write PF_s (for pseudorandom function) to denote the distribution $DIST_{f_s}$. The efficient evaluation property of Definition 12 implies that the class $\mathcal{PF}_{n+1} = \{PF_s : s \in \{0,1\}^n\}$ has polynomial-size generators.

The intuition behind this construction is simple: the weighting system used by distribution PF_s ensures that in order to achieve high accuracy with respect to PF_s, a hypothesis must contain information about many values of the function f_s. But this can only be achieved by a large

hypothesis, since to a polynomial-time learning algorithm the values $f_s(j_\sharp)$ appear totally random and "a random string admits no short description."

Theorem 13 *If one-way functions exist, then*

1. *Distribution class $\mathcal{PF}_{n+1}$ can be learned with either a generator or an evaluator by a polynomial-time algorithm which has sample complexity $O(\log(1/\epsilon)/\epsilon^2)$.*
2. *Any polynomial-time algorithm which learns $\mathcal{PF}_{n+1}$ with either a generator or an evaluator must have hypothesis size $\Omega(1/\epsilon^{1/2-c})$ for any constant $c > 0$.*

Proof: The proof of part (1) is similar to the corresponding proof for Theorem 5. Given the $2^t - 1$ bits $f_s(0_\sharp), \ldots, f_s((2^t-2)_\sharp)$, a "safe" guess at PF_s is the distribution PF_s^t such that

- $PF_s^t[j_\sharp b] = PF_s[j_\sharp b]$ for $j \in \{0, \ldots, 2^t - 2\}$ and $b \in \{0,1\}$,
- $PF_s^t[j_\sharp 0] = PF_s^t[j_\sharp 1] = PF_s[j_\sharp f_s(j_\sharp)]/2$ for $j \in \{2^t - 1, \ldots, 2^n - 1\}$.

It is simple to construct a generator (evaluator) for PF_s^t from the bits $f_s(0_\sharp), \ldots, f_s((2^t-2)_\sharp)$. A simple calculation shows that making $O(t2^{2t})$ calls to $GEN(PF_s)$ will reveal all of the bits $f_s(0_\sharp), \ldots, f_s((2^t-2)_\sharp)$ with probability at least 9/10. Since $KL(PF_s||PF_s^t) = 1/2^t$, taking $t = \lceil \log 1/\epsilon \rceil$ proves part (1).

The proof of part (2) is by contradiction; so suppose that A' is a polynomial-time algorithm for learning $\mathcal{PF}_{n+1}$ with a generator which has hypothesis size $h \leq n^d/\epsilon^{1/2-c}$ for some nonnegative absolute constants c, d with $0 < c \leq 1/2$. The following algorithm, which we call algorithm B, has oracle access to a function $g : \{0,1\}^n \to \{0,1\}$, runs in poly($n$) time, and outputs a single bit. (The idea is that algorithm B outputs 1 if it thinks that the function g is drawn from $\{f_s\}_{s \in U_n}$ and outputs 0 if it thinks that g is drawn from $\mathcal{F}_n$.)

Algorithm B:

1. Run learning algorithm A' using the example oracle $GEN(DIST_g)$ with ϵ set to $1/n^{1+d/c}$. Let G be the generator which A' outputs as its hypothesis and let r be the number of inputs to G. If size$(G) > n^d/\epsilon^{1/2-c} = n^{1/2+d/2c-c}$ abort and output 0, else proceed to step 2.
2. Let $t = \lceil (1/2 + d/2c - c/2) \log n \rceil$ and let $S_t = \{2^{t-1} - 1, \ldots, 2^t - 2\}$. Choose $j \in S_t$ uniformly at random. For $i = 1, \ldots, T = 448 \cdot 2^{2t-1}$ let x^i be a random r-bit string. For $b \in \{0,1\}$ let
$$\hat{p}_{j,b} = \frac{|\{i \in \{1, \ldots, T\} : G(x^i) = j_\sharp b\}|}{T}.$$
Let β be the bit such that $\hat{p}_{j,\beta} \geq \hat{p}_{j,\overline{\beta}}$ (breaking ties randomly). If $\beta = g(j_\sharp)$ then output 1, else output 0.

For $j \in S_t$ and $b \in \{0,1\}$ let $p_{j,b} = D_G[j_\sharp b]$; so $p_{j,b}$ is the true probability of the string $j_\sharp b$ under the distribution generated by G, and $\hat{p}_{j,b}$ is an estimate of this probability obtained by sampling.

In the next two subsections we prove the following lemmas:

Lemma 14 *(Generator)* $\Pr[B \text{ outputs } 1 \mid g \in \{f_s\}_{s \in U_n}] \geq 8/10$.

Lemma 15 *(Generator)* $\Pr[B \text{ outputs } 1 \mid g \in \mathcal{F}_n] \leq 3/4 + o(1)$.

(Note that each of the above probabilities is over the random choice of g, the random responses of $GEN(DIST_g)$ in step 1, any internal randomness in the execution of algorithm A', the random choice of j in step 2, and the random choice of strings $x^1, \ldots, x^T$ in step 2.) Since algorithm B runs in poly(n) time, together these two lemmas contradict the fact that $\{f_s\}_{s \in \{0,1\}^*}$ is a pseudorandom function family, thus proving part (2) of Theorem 13 for learning with a generator.

5.1.1 Learning using a pseudorandom oracle

In this subsection we prove Lemma 14, so we assume that g is selected from the distribution $\{f_s\}_{s\in U_n}$ and hence that $g = f_s$ for some $s \in \{0,1\}^n$ (so consequently $DIST_g = PF_s$).

Note that in step 1 of algorithm B, to simulate a call to the example oracle $GEN(PF_s)$ algorithm B can generate a string $j_\sharp$ with the appropriate probability, obtain the value of $f_s(j_\sharp)$ from the oracle for f_s, and outputs the string $j_\sharp f_s(j_\sharp)$. By our assumption on the hypothesis size of algorithm A' we have that $\text{size}(G) \leq n^d/\epsilon^{1/2-c} = n^{1/2+d/2c-c}$, so algorithm B will reach step 2.

Since A' is a learning algorithm for $\mathcal{PF}_{n+1}$, we have that G is an ϵ-good generator for PF_s with probability at least $9/10$. Let α_j denote the bit $f_s(j_\sharp)$. The next claim is proved in Appendix B:

Claim 16 *If D_G is ϵ-good (i.e. $KL(PF_s||D_G) \leq \epsilon$), then for all $j \in \{2^{t-1}-1,\ldots,2^t-2\}$ we have $p_{j,\alpha_j} \geq \frac{7}{8}\cdot\frac{1}{2^{2t-1}}$ and $p_{j,\overline{\alpha}_j} \leq \frac{1}{8}\cdot\frac{1}{2^{2t-1}}$.*

A straightforward application of Chernoff bounds ([14], p. 190) now shows that

$$\Pr\left[\hat{p}_{j,\alpha_j} \geq \frac{3}{4}\cdot\frac{1}{2^{2t-1}}\right] \geq 1-e^{-4} \qquad \text{and} \qquad \Pr\left[\hat{p}_{j,\overline{\alpha}_j} \leq \frac{1}{4}\cdot\frac{1}{2^{2t-1}}\right] \geq 1-e^{-4}$$

Hence, conditioned on the fact that G is ϵ-good, we have that $\Pr[\hat{p}_{j,\alpha_j} > \hat{p}_{j,\overline{\alpha}_j}] \geq 1-2e^{-4}$ and thus $\Pr[B \text{ outputs } 1] \geq 1-2e^{-4}$. Since G is ϵ-good with probability at least $9/10$, we have that the overall probability that algorithm B outputs 1 is at least $(9/10)\cdot(1-2e^{-4}) > 8/10$.

Thus, for any $s \in \{0,1\}^n$ we have $\Pr[B \text{ outputs } 1 \mid g = f_s] > 8/10$. This clearly implies that $\Pr[B \text{ outputs } 1 \mid g \in \{f_s\}_{s\in U_n}] > 8/10$. (Lemma 13) ■

5.1.2 Learning using a truly random oracle

In this subsection we prove Lemma 15, so now we assume that $g = F$ is selected from the distribution $\mathcal{F}_n$ which is uniform over all functions mapping $\{0,1\}^n$ to $\{0,1\}$.

In this scenario it is possible that algorithm B will abort and output 0 in step 1, since algorithm A' is guaranteed to return a generator of size $h \leq n^d/\epsilon^{1/2-c}$ only if it is invoked with an oracle $GEN(D)$ for some $D = PF_s$. However, since we want to upper bound the probability that B outputs 1, we can safely assume that $\text{size}(G) \leq n^{1/2+d/2c-c}$ so algorithm B will reach step 2.

Now we show that the condition $\text{size}(G) \leq n^{1/2+d/2c-c}$ forces the output of step 2 to be the bit 1 with probability at most $3/4 + o(1)$. An intuitive argument goes as follows: a generator of size at most $n^{1/2+d/2c-c}$ can encode at most $n^{1/2+d/2c-c}$ of the 2^{t-1} bits $\{F(j_\sharp)\}_{j\in S_t}$. Since $2^{t-1} = \Theta(n^{1/2+d/2c-c/2})$, intuitively the probability that the generator encodes $F(j_\sharp)$ for a randomly chosen $j \in S_t$ is $O(n^{-c/2})$. Since the function F is totally random, if the generator does not encode the bit $F(j_\sharp)$ then the probability that the generator predicts $F(j_\sharp)$ correctly is $1/2$. Hence the overall probability that the generator predicts correctly on a randomly chosen j should be $1/2 + O(n^{-c/2})$. (This intuition could be made rigorous; however, the bound of $3/4 + o(1)$ is simpler to prove and suffices for our purposes.)

Given a generator G', a function $F : \{0,1\}^n \to \{0,1\}$, and a number $0 \leq \rho \leq 1$, we say that F *$\rho-$agrees with G'* if

$$|\{j \in S_t : D_{G'}[j_\sharp F(j_\sharp)] > D_{G'}[j_\sharp \overline{F(j_\sharp)}\}| \;\geq\; \rho\cdot|S_t|.$$

Let $\rho = 1/2+\gamma$. For any fixed G' we have that the probability (over the random choice of $F \in \mathcal{F}_n$) that F $\rho-$agrees with G' is at most

$$\frac{1}{2^{|S_t|}}\cdot\sum_{k=\lceil\rho|S_t|\rceil}^{|S_t|}\binom{|S_t|}{k} \leq e^{-|S_t|\gamma^2/6},$$

where the inequality follows from Chernoff bounds on the tail of the unbiased binomial distribution. Since, as noted in Section 2, there are at most $2 \cdot 2^{n^{1/2+d/2c-c}}$ generators of size at most $n^{1/2+d/2c-c}$, the probability (over the random choice of F) that there is some generator of size at most $n^{1/2+d/2c-c}$ which F $\rho-$agrees with is at most $2 \cdot 2^{n^{1/2+d/2c-c}} \cdot e^{-|S_t|\gamma^2/6}$. Using the fact that $|S_t| \geq (n^{1/2+d/2c-c/2})/2$, if we take $\gamma = n^{-c/8}$ then this is at most

$$a(n) \equiv 2 \cdot 2^{n^{1/2+d/2c-c}} \cdot e^{-(n^{1/2+d/2c-3c/4})/12} = o(1).$$

Since A' outputs some generator G with size$(G) \leq n^{1/2+d/2c-c}$, it follows that with probability at least $1 - a(n)$ (over the random choice of F) the function F does not $(1/2+\gamma)-$agree with G, i.e.

$$|\{j \in S_t : D_G[j_\sharp F(j_\sharp)] > D_G[j_\sharp \overline{F(j_\sharp)}]\}| \ < \ (1/2+\gamma) \cdot |S_t|. \tag{1}$$

We thus suppose that F does not $(1/2+\gamma)-$agree with G. Inequality (1) implies that for at least a $(1/2-\gamma)-$fraction of the values j in S_t, we have

$$D_G[j_\sharp \overline{F(j_\sharp)}] \geq D_G[j_\sharp F(j_\sharp)]. \tag{2}$$

If we now suppose that the value of j which is chosen in step 2 is such that equation (2) holds, it remains only to bound the probability (across the random choice of $x^1, \ldots, x^T$) that $\hat{p}_{j,F(j_\sharp)} \geq \hat{p}_{j,\overline{F(j_\sharp)}}$. The following simple claim is proved in Appendix C:

Claim 17 *For fixed j, F and G we have* $\Pr[\hat{p}_{j,\overline{F(j_\sharp)}} \geq \hat{p}_{j,F(j_\sharp)} \mid D_G[j_\sharp \overline{F(j_\sharp)}] \geq D_G[j_\sharp F(j_\sharp)]] \ \geq \ 1/2$, *where the probability is over the random choice of $x^1, \ldots, x^T$ that determines $\hat{p}_{j,\overline{F(j_\sharp)}}$ and $\hat{p}_{j,F(j_\sharp)}$.*

Since ties are broken randomly if $\hat{p}_{j,\overline{F(j_\sharp)}} = \hat{p}_{j,F(j_\sharp)}$, it follows from Claim 17 that inequality (2) implies $\Pr[B \text{ outputs } 0] \geq 1/2$. Thus, the overall probability that algorithm B outputs 0 is at least $(1-a(n))(1/2-\gamma)(1/2) = 1/4 - o(1)$, which proves lemma 15. Recapitulating, we have shown that

$$\begin{aligned}
\Pr[B \text{ outputs } 0] \ &\geq \ \Pr[(F \text{ does not } (1/2+\gamma) - \text{agree with } G) \ \& \\
&\qquad (j \text{ satisfies } D_G[j_\sharp \overline{F(j_\sharp)}] \geq D_G[j_\sharp F(j_\sharp)]) \ \& \ (B \text{ outputs } 0)] \\
&= \ \Pr[F \text{ does not } (1/2+\gamma) - \text{agree with } G] \ \cdot \\
&\quad \Pr[j \text{ satisfies } D_G[j_\sharp \overline{F(j_\sharp)}] \geq D_G[j_\sharp F(j_\sharp)] \mid F \text{ does not } (1/2+\gamma) - \text{agree with } G] \cdot \\
&\quad \Pr[B \text{ outputs } 0 \mid (j \text{ satisfies } D_G[j_\sharp \overline{F(j_\sharp)}] \geq D_G[j_\sharp F(j_\sharp)]) \ \& \\
&\qquad\qquad (F \text{ does not } (1/2+\gamma) - \text{agree with } G)] \\
&\geq \ (1-a(n)) \cdot (1/2-\gamma) \cdot 1/2 \\
&= \ 1/4 - o(1). \qquad \text{(Lemma 15)} \ \blacksquare
\end{aligned}$$

5.1.3 Wrapping up

We have proved Lemmas 14 and 15 for the case of learning with a generator, and thus have established part (2) of Theorem 13 for the case of learning with a generator. The corresponding proofs for learning with an evaluator are slightly simpler. Algorithm B now outputs an evaluator E as its hypothesis in step 1. In step 2 the values t, S_t and j are as before, but we need only call the evaluator twice to find $p_{j,0} = D_E[j_\sharp 0]$ and $p_{j,1} = D_E[j_\sharp 1]$ exactly without sampling.

In the case where $g \in \{f_s\}_{s \in U_n}$, Claim 16 now implies that $\Pr[B \text{ outputs } 1 \mid E \text{ is } \epsilon-\text{good}] = 1$. Since E is ϵ-good with probability at least $9/10$, we obtain the following analogue to Lemma 14.

Lemma 18 *(Evaluator)* $\Pr[B \text{ outputs } 1 \mid g \in \{f_s\}_{s \in U_n}] \geq 9/10$.

The argument when $g = F \in \mathcal{F}_n$ is entirely analogous to the proof of Section 5.1.2. The only difference is that we no longer need Claim 17, since now there is no sampling procedure. Since we break ties at random if $D_E[j_\sharp F(j_\sharp)] = D_E[j_\sharp \overline{F(j_\sharp)}]$, we still have that $\Pr[B \text{ outputs } 0 \mid D_E[j_\sharp \overline{F(j_\sharp)}] \geq D_E[j_\sharp F(j_\sharp)]] \geq 1/2$. The proof goes through as before, and we obtain

Lemma 19 *(Evaluator)* $\Pr[B \text{ outputs } 1 \mid g \in \mathcal{F}_n] \leq 3/4 + o(1)$.

This concludes the proof of Theorem 13. ■

6 Discussion

As mentioned earlier, the general problem of learning an approximation to an unknown probability distribution from random draws has received considerable attention in the pattern recognition and statistics literature. In particular, the field of *nonparametric density estimation* in statistics is concerned with the problem of estimating an unknown distribution D on the basis of a random sample, where no parametric structure is assumed for D (see Izenman [12] for an overview). Many commonly used algorithms in nonparametric density estimation and pattern recognition, such as kernel-based approaches and nearest-neighbor algorithms, have as a first step the "memorization" of the entire sample. Theorem 13 is particularly interesting in light of the widespread use of such algorithms, since it implies that even simple processes (computable by polynomial-size circuits) can transform a pure source of unbiased random bits into a distribution which requires any efficient learning algorithm to use large hypotheses, i.e. to effectively perform some sort of memorization. This result can thus be viewed as providing theoretical justification for the use of memorization-based algorithms for nonparametric distribution learning problems.

7 Acknowledgements

We thank Steven Gortler and Les Valiant for helpful discussions and advice.

References

[1] M. Anthony and N. Biggs. *Computational Learning Theory: an Introduction.* Cambridge Univ. Press, 1997.

[2] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth. Occam's razor, *Inf. Proc. Let.,* 24(6), 1987, 377-380.

[3] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension, *J. ACM,* 36(4), 1989, 929-965.

[4] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley, 1991.

[5] S. Decatur, O. Goldreich and D. Ron. Computational sample complexity, *Proc. 10th Conf. on Comp. Learning Theory,* 1997, 130-142.

[6] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* Wiley, 1973.

[7] F. Ergun, S.R. Kumar, R. Rubinfeld. Learning distributions from random walks, *Proc. 10th Conf. on Comput. Learning Theory,* 1997, 243-249.

[8] Y. Freund. Boosting a weak learning algorithm by majority, *Inf. and Comput.,* 121(2), 1995, 256-285.

[9] O. Goldreich. *Foundations of Cryptography (Fragments of a Book),* available at `http://www.wisdom.weizmann.ac.il/~oded/frag.html`, 1995.

[10] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudo-randomness.* Algorithms and Combinatorics series (Vol. 17), Springer, 1998.

[11] O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions, *J. ACM,* 33(4), 1986, 792-807.

[12] A. Izenman. Recent developments in nonparametric density estimation, *J. Amer. Stat. Assoc.,* 86(413), 1991, 205-224.

[13] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. Schapire and L. Sellie, On the learnability of discrete distributions, *Proc. 26th Symp. on Theory of Computing,* 1994, 273-282.

[14] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994.

[15] M. Luby. *Pseudorandomness and Cryptographic Applications.* Princeton Univ. Press, 1999.

[16] M. Naor. Evaluation may be easier than generation, *Proc. 28th Symp. on Theory of Computing,* 1996, 74-83.

[17] D. Ron, Y. Singer, N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length, *Machine Learning,* **25**(2-3) (1998), 117-149.

[18] D. Ron, Y. Singer, N. Tishby. On the learnability and usage of acyclic probabilistic finite automata, *J. Comput. Syst. Sci,* **56**(2) (1998), 133-152.

[19] R. Schapire. The strength of weak learnability, *Machine Learning,* 5, 1990, 197-227.

[20] R. Servedio. Computational sample complexity and attribute-efficient learning, *J. Comput. Syst. Sci.,* **60**(1) (2000), 161-178.

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

[22] V. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer-Verlag, 1982.

A Proof of Claim 6

We represent distribution D_1 by the probability vector $(p_1, \ldots, p_k)$, distribution D_2 by the probability vector $(q_1, \ldots, q_k)$ and the unknown distribution D by the probability vector $(r_1, \ldots, r_k)$. Our goal is to minimize

$$KL(D_1||D) + KL(D_2||D) = \sum_{i=1}^{k} p_i \log \frac{p_i}{r_i} + \sum_{i=1}^{k} q_i \log \frac{q_i}{r_i}$$

subject to the constraint $\sum_{i=1}^{k} r_i = 1$. Using Lagrange multipliers, we want to minimize

$$J = \sum_{i=1}^{k} p_i \log \frac{p_i}{r_i} + \sum_{i=1}^{k} q_i \log \frac{q_i}{r_i} + \lambda \sum_{i=1}^{k} r_i.$$

Differentiating with respect to r_i, we obtain $\frac{\partial J}{\partial r_i} = -\frac{p_i}{r_i} - \frac{q_i}{r_i} + \lambda$, which implies that $r_i = (p_i + q_i)/\lambda$. Summing this equation across $i = 1, \ldots, k$ we obtain that $\lambda = 2$ which proves the claim. ■

B Proof of Claim 16

For the sake of clarity we prove a slightly more general version of the claim.

Claim 20 *Let D_1 be a distribution represented by the probability vector $(q_0, \ldots, q_{2^n-1})$ where*

- $q_j = 1/2^{2i-1}$ *for j such that $2^{i-1} - 1 \leq j < 2^i - 1$, for each $i \in \{1, \ldots, n-1\}$,*
- $q_j = 1/2^{2n-2}$ *for $2^{n-1} - 1 \leq j < 2^n - 1$,*
- $q_{2^n-1} = 0$,

and let D_2 be a distribution represented by $(p_0, \ldots, p_{2^n-1})$ such that $KL(D_1||D_2) \leq 1/2^{r \log n}$ for some constant $r \geq 1$. Then for any constant $s < r/2$, for all sufficiently large values of n, for all $j \in \{2^{\lceil s \log n \rceil - 1} - 1, \ldots, 2^{\lceil s \log n \rceil} - 2\}$, we have

$$p_j \geq \frac{7}{8} \cdot \frac{1}{2^{2\lceil s \log n \rceil - 1}} \quad (*) \qquad \textit{and} \qquad p_{2^n-1} \leq \frac{1}{8} \cdot \frac{1}{2^{2\lceil s \log n \rceil - 1}}. \quad (**)$$

Claim 16 is the special case of Claim 20 obtained by taking $D_1 = PF_s$, $D_2 = D_G$, $r = 1 + d/c$ (so $1/2^{r \log n} = 1/n^{1+d/c} = \epsilon$), and $s = 1/2 + d/2c - c$ (so $2\lceil s \log n \rceil - 1 = 2t - 1$).

Proof of Claim 20: We prove the second inequality first: Consider the problem of minimizing $KL(D_1||D_2)$ subject to the constraint that p_{2^n-1} is some fixed value. This is equivalent to minimizing $\sum_{i=0}^{2^n-2} q_i \log(q_i/p_i)$ subject to the constraint that $\sum_{i=0}^{2^n-2} p_i = 1 - p_{2^n-1}$. Using Lagrange multipliers, we want to minimize

$$J = \sum_{i=0}^{2^n-2} q_i \log \frac{q_i}{p_i} + \lambda \sum_{i=0}^{2^n-2} p_i.$$

Differentiation yields $\frac{\partial J}{\partial p_i} = -\frac{q_i}{p_i} + \lambda$, which implies that $p_i = q_i/\lambda$ for $i = 0, \ldots, 2^n - 2$. Summing these equations yields $\lambda = 1/(1 - p_{2^n-1})$, which in turn implies that

$$\frac{1}{2^{r \log n}} \geq KL(D_1||D_2) = \sum_{i=0}^{2^n-2} q_i \log \frac{q_i}{p_i} = \log \frac{1}{1 - p_{2^n-1}} \geq p_{2^n-1},$$

where we used the fact that $1 - x \leq 2^{-x}$ for $x \geq 0$ to obtain the last inequality. Inequality $(**)$ follows on noting that $s < r/2$ implies $(1/8) \cdot (1/2^{2\lceil s \log n \rceil - 1}) \geq 1/2^{r \log n}$ for n sufficiently large.

We take a similar approach to prove inequality $(*)$. Fix any $j \in 2^{\lceil s \log n \rceil - 1} - 1, \ldots, 2^{\lceil s \log n \rceil} - 2\}$. We now consider the problem of minimizing $KL(D_1||D_2)$ subject to the constraint that p_j is some fixed value; this is equivalent to minimizing $\sum_{i \neq j} q_i \log(q_i/p_i)$ subject to $\sum_{i \neq j} p_i = 1 - p_j$. Again applying Lagrange multipliers, we want to minimize

$$J = \sum_{i \neq j} q_i \log \frac{q_i}{p_i} + \lambda \sum_{i \neq j} p_i.$$

As before, differentiation yields $\frac{\partial J}{\partial p_i} = -q_i/p_i + \lambda$ and hence $p_i = q_i/\lambda$ for $i \neq j$. Summing these equations and solving for λ, since $q_j = 1/2^{2\lceil s \log n \rceil - 1}$ we obtain $\lambda = (1 - 1/2^{2\lceil s \log n \rceil - 1})/(1 - p_j)$. If we let $\tau = 2\lceil s \log n \rceil - 1$, we thus have

$$KL(D_1||D_2) = \sum_{i \neq j} q_i \log \frac{q_i}{p_i} + q_j \log \frac{q_j}{p_j} = \left(1 - \frac{1}{2^\tau}\right) \log \frac{1 - 1/2^\tau}{1 - p_j} + \frac{1}{2^\tau} \log \frac{1/2^\tau}{p_j}. \quad (3)$$

If we let $g(p_j)$ denote the right hand side of (3) viewed as a function of p_j, then $g(p_j)$ is convex and achieves its minimum at $p_j = 1/2^\tau$, so the minimum of $g(p_j)$ over the interval $[0, (7/8)\cdot(1/2^\tau)]$ occurs at $C = (7/8)\cdot(1/2^\tau)$, where we have

$$g(C) = \left(1 - \frac{1}{2^\tau}\right) \log \frac{1 - 1/2^\tau}{1 - (7/8)(1/2^\tau)} + \frac{1}{2^\tau} \log \frac{8}{7}.$$

If we substitute y for $1/2^\tau$, this becomes $(1-y)\log\frac{1-y}{1-7y/8} + y\log\frac{8}{7}$, which can be shown to be at least $y/100$ for $y \in [0,1]$.

Thus, we have shown that if p_j is less than $(7/8)\cdot(1/2^{2^{\lceil s\log n\rceil - 1}})$, then $KL(D_1||D_2)$ is greater than $(1/100)\cdot(1/2^{2^{\lceil s\log n\rceil - 1}})$. Since this quantity asymptotically exceeds $1/2^{r\log n}$, it follows that if $KL(D_1||D_2) \leq 1/2^{r\log n}$ then $p_j \geq (7/8)\cdot(1/2^{2^{\lceil s\log n\rceil - 1}})$ for sufficiently large n. ■

C Proof of Claim 17

This is equivalent to showing that $\Pr\left[\sum_{i=1}^T X_i \geq 0\right] \geq 1/2$, where $X_1, \ldots, X_T$ are i.i.d. random variables with $\Pr[X_i = 1] = p$, $\Pr[X_i = -1] = q \leq p$, and $\Pr[X_i = 0] = 1 - p - q$. This inequality clearly follows from the inequalities $\Pr\left[\sum_{i=1}^T X_i \geq 0 \mid \sum_{i=1}^T |X_i| = k\right] \geq 1/2$ for $k \in \{0, \ldots, T\}$. Each of these inequalities is in turn equivalent to $\Pr\left[\sum_{i=1}^k Y_i \geq 0\right] \geq 1/2$, where $Y_1, \ldots, Y_k$ are i.i.d. Bernoulli random variables with $\Pr[Y_i = 1] = p' \geq 1/2$ and $\Pr[Y_i = -1] = 1 - p' \leq 1/2$. This last inequality is a standard fact about the binomial distribution. (Claim 17) ■