Optimal Cryptographic Hardness of Learning Monotone Functions*

Dana Dachman-Soled, Homin K. Lee, Tal Malkin,
Rocco A. Servedio, Andrew Wan, and Hoeteck Wee

Columbia University
{dglasner,homin,tal,rocco,atw12,hoeteck}@cs.columbia.edu

Abstract. A wide range of positive and negative results have been established for learning different classes of Boolean functions from uniformly distributed random examples. However, polynomial-time algorithms have thus far been obtained almost exclusively for various classes of *monotone* functions, while the computational hardness results obtained to date have all been for various classes of general (nonmonotone) functions. Motivated by this disparity between known positive results (for monotone functions) and negative results (for nonmonotone functions), we establish strong computational limitations on the efficient learnability of various classes of monotone functions.

We give several such hardness results which are provably almost optimal since they nearly match known positive results. Some of our results show cryptographic hardness of learning polynomial-size monotone circuits to accuracy only slightly greater than $1/2 + 1/\sqrt{n}$; this accuracy bound is close to optimal by known positive results (Blum *et al.*, FOCS '98). Other results show that under a plausible cryptographic hardness assumption, a class of constant-depth, sub-polynomial-size circuits computing monotone functions is hard to learn; this result is close to optimal in terms of the circuit size parameter by known positive results as well (Servedio, Information and Computation '04). Our main tool is a complexity-theoretic approach to hardness amplification via noise sensitivity of monotone functions that was pioneered by O'Donnell (JCSS '04).

1 Introduction

More than two decades ago Valiant introduced the Probably Approximately Correct (PAC) model of learning Boolean functions from random examples [Val84]. Since that time a great deal of research effort has been expended on trying to understand the inherent abilities and limitations of computationally efficient learning algorithms. This paper addresses a discrepancy between known positive and negative results for uniform distribution learning by establishing strong computational hardness results for learning various classes of *monotone* functions.

* Supported by NSF award CNS-0716245, DARPA grant HR0011-07-1-0012, NSF CAREER award CCF-0347282, NSF CAREER CCF-03-047839, NSF award SBE-0245014, NSF award CCF-0523664, and a Sloan Foundation Fellowship.

1.1 Background and Motivation

In the uniform distribution PAC learning model, a learning algorithm is given access to a source of independent random examples $(x, f(x))$ where each x is drawn uniformly from the n-dimensional Boolean cube and f is the unknown Boolean function to be learned. The goal of the learner is to construct a high-accuracy hypothesis function h, *i.e.,* one which satisfies $\Pr[f(x) \neq h(x)] \leq \epsilon$ where the probability is with respect to the uniform distribution and ϵ is an error parameter given to the learning algorithm. Algorithms and hardness results in this framework have interesting connections with topics such as discrete Fourier analysis [Man94], circuit complexity [LMN93], noise sensitivity and influence of variables in Boolean functions [KKL88, KOS04, OS07], and cryptography [BFKL93, Kha95]. For these reasons, and because the model is natural and elegant in its own right, the uniform distribution learning model has been intensively studied for almost two decades.

Monotonicity makes learning easier. For many classes of functions, uniform distribution learning algorithms have been devised which substantially improve on a naive exponential-time approach to learning via brute-force search. However, despite intensive efforts, researchers have not yet obtained poly(n)-time learning algorithms in this model for various simple classes of functions. Interestingly, in many of these cases restricting the class of functions to the corresponding class of *monotone* functions has led to more efficient – sometimes poly(n)-time – algorithms. We list some examples:

1. A simple algorithm learns monotone $O(\log n)$-juntas to perfect accuracy in poly(n) time, and a more complex algorithm [BT96] learns monotone $\tilde{O}(\log^2(n))$-juntas to any constant accuracy in poly(n) time. In contrast, the fastest known algorithm for learning arbitrary k-juntas runs in time $n^{.704k}$ [MOS04].
2. The fastest known uniform distribution learning algorithm for the general class of s-term DNF, due to Verbeurgt [Ver90], runs in time $n^{O(\log s)}$ to learn to any constant accuracy. In contrast, for s-term monotone DNF [Ser04] gives an algorithm which runs in $s^{O(\log s)}$ time. Thus the class of $2^{O(\sqrt{\log n})}$-term monotone DNF can be learned to any constant accuracy in poly(n) time, but no such result is known for $2^{O(\sqrt{\log n})}$-term general DNF.
3. The fastest known algorithm for learning poly(n)-size general decision trees to constant accuracy takes $n^{O(\log n)}$ time (this follows from [Ver90]), but poly(n)-size decision trees that compute monotone functions can be learned to any constant accuracy in poly(n) time [OS07].
4. No poly(n)-time algorithm can learn the general class of all Boolean functions on $\{0,1\}^n$ to accuracy better than $\frac{1}{2} + \frac{\text{poly}(n)}{2^n}$, but a simple polynomial-time algorithm can learn the class of all monotone Boolean functions to accuracy $\frac{1}{2} + \frac{\Omega(1)}{\sqrt{n}}$ [BBL98]. We note also that the result of [BT96] mentioned above follows from a $2^{\tilde{O}(\sqrt{n})}$-time algorithm for learning arbitrary

monotone functions on n variables to constant accuracy (it is easy to see that no comparable algorithm can exist for learning arbitrary Boolean functions to constant accuracy).

Cryptography and hardness of learning. Essentially all known representation-independent hardness of learning results (*i.e.,* results which apply to learning algorithms that do not have any restrictions on the syntactic form of the hypotheses they output) rely on some cryptographic assumption, or an assumption that easily implies a cryptographic primitive. For example, under the assumption that certain subset sum problems are hard on average, Kharitonov [Kha95] showed that the class AC^1 of logarithmic-depth, polynomial-size AND/OR/NOT circuits is hard to learn under the uniform distribution. Subsequently Kharitonov showed [Kha93] that if factoring Blum integers is $2^{n^{\epsilon}}$-hard for some fixed $\epsilon > 0$, then even the class AC^0 of constant-depth, polynomial-size AND/OR/NOT circuits similarly cannot be learned in polynomial time under the uniform distribution. In later work, Naor and Reingold [NR04] gave constructions of pseudorandom functions with very low circuit complexity; their results imply that if factoring Blum integers is super-polynomially hard, then even depth-5 TC^0 circuits (composed of MAJ and NOT gates) cannot be learned in polynomial time under uniform. We note that all of these hardness results apply even to algorithms which may make black-box "membership queries" to obtain the value $f(x)$ for inputs x of their choosing.

Monotonicity versus cryptography? Given that cryptography precludes efficient learning while monotonicity seems to make efficient learning easier, it is natural to investigate how these phenomena interact. One could argue that prior to the current work there was something of a mismatch between known positive and negative results for uniform-distribution learning: as described above a fairly broad range of polynomial-time learning results had been obtained for various classes of monotone functions, but there were no corresponding computational hardness results for monotone functions. Can all monotone Boolean functions computed by polynomial-size circuits be learned to 99% accuracy in polynomial time from uniform random examples? As far as we are aware, prior to our work answers were not known even to such seemingly basic questions about learning monotone functions as this one. This gap in understanding motivated the research presented in this paper (which, as we describe below, lets us answer "no" to the above question in a strong sense).

1.2 Our Results and Techniques: Cryptography Trumps Monotonicity

We present several different constructions of "simple" (polynomial-time computable) monotone Boolean functions and prove that these functions are hard to learn under the uniform distribution, even if membership queries are allowed. We now describe our main results, followed by a high-level description of how we obtain them.

In [BBL98] Blum *et al.* showed that arbitrary monotone functions cannot be learned to accuracy better than $\frac{1}{2} + \frac{O(\log n)}{\sqrt{n}}$ by any algorithm which makes poly(n) many membership queries. This is an information-theoretic bound which is proved using randomly generated monotone DNF formulas of size (roughly) $n^{\log n}$. A natural goal is to obtain *computational* lower bounds for learning polynomial-time-computable monotone functions which match, or nearly match, this level of hardness (which is close to optimal by the $(\frac{1}{2} + \frac{\Omega(1)}{\sqrt{n}})$-accuracy algorithm of Blum *et al.* described above). We prove near-optimal hardness for learning polynomial-size monotone circuits:

Theorem 1 (informal). *If one-way functions exist, then there is a class of* poly(n)*-size monotone circuits that cannot be learned to accuracy* $\frac{1}{2} + \frac{1}{n^{1/2-\epsilon}}$ *for any fixed* $\epsilon > 0$.

Our approach yields even stronger lower bounds if we make stronger assumptions:

– Assuming the existence of subexponential one-way functions, we improve the bound on the accuracy to $1/2 + \text{polylog}(n)/n^{1/2}$.
– Assuming the hardness of factoring Blum integers, our hard-to-learn functions may be computed in monotone NC^1.
– Assuming that Blum integers are 2^{n^ϵ}-hard to factor on average (the same hardness assumption used in Kharitonov's work [Kha93]), we obtain a lower bound for learning constant-depth circuits of sub-polynomial size that almost matches the positive result in [Ser04]. More precisely, we show that for any (sufficiently large) constant d, the class of monotone functions computed by depth-d AND/OR/NOT circuits of size $2^{(\log n)^{O(1)/(d+1)}}$ cannot be learned to accuracy 51% under the uniform distribution in poly(n) time. In contrast, the positive result of [Ser04] shows that monotone functions computed by depth-d AND/OR/NOT circuits of size $2^{O((\log n)^{1/(d+1)})}$ can be learned to any constant accuracy in poly(n) time.

These results are summarized in Figure 1.

Proof techniques. A natural first approach is to try to "pseudorandomize" [BBL98]'s construction of random $n^{\log n}$-term monotone DNFs. While we were not able to do this directly, it turns out that a closely related approach does yield some results along the desired lines. In the full version of the paper (available online), we present and analyze a simple variant of the [BBL98] information-theoretic construction and then show how to "pseudorandomize" the variant. Since our variant gives a weaker quantitative bound on the information-theoretic hardness of learning than [BBL98], this gives a construction of polynomial-time-computable monotone functions which, assuming the existence of one-way functions, cannot be learned to accuracy $\frac{1}{2} + \frac{1}{\text{polylog}(n)}$ under the uniform distribution. While this answers the question posed above (even with "51%" in place of "99%"), the $\frac{1}{\text{polylog}(n)}$ factor is rather far from the $\frac{O(\log n)}{\sqrt{n}}$ factor that one might hope for as described above.

In Section 2 we use a different construction to obtain much stronger quantitative results; another advantage of this second construction is that it enables

Hardness assumption	**Complexity of f**	**Accuracy bound**	**Ref.**
none	random $n^{\log n}$-term mono. DNF	$\frac{1}{2} + \frac{\omega(\log n)}{n^{1/2}}$	[BBL98]
OWF (poly)	poly-size monotone circuits	$\frac{1}{2} + \frac{1}{n^{1/2-\epsilon}}$	Thm. 1
OWF (2^{n^α})	poly-size monotone circuits	$\frac{1}{2} + \frac{\mathrm{poly}(\log n)}{n^{1/2}}$	Thm. 3
factoring BI (poly)	monotone NC^1-circuits	$\frac{1}{2} + \frac{1}{n^{1/2-\epsilon}}$	FV
factoring BI (2^{n^α})	AND/OR/NOT circuits of size $2^{(\log n)^{O(1)/(d+1)}}$ and depth d	$\frac{1}{2} + o(1)$	FV

Fig. 1. Summary of known hardness results for learning monotone Boolean functions. The meaning of each row is as follows: under the stated hardness assumption, there is a class of monotone functions computed by circuits of the stated complexity which no poly(n)-time membership query algorithm can learn to the stated accuracy. In the first column, OWF and BI denote one-way functions and Blum Integers respectively, and "poly" and "2^{n^α}" means that the problems are intractable for poly(n)- and 2^{n^α}-time algorithms respectively (for some fixed $\alpha > 0$). Recall that the poly(n)-time algorithm of [BBL98] for learning monotone functions implies that the best possible accuracy bound for monotone functions is $\frac{1}{2} + \frac{\Omega(1)}{n^{1/2}}$. "FV" means the result is established in the full version of this paper.

us to show hardness of learning *monotone circuits* rather than just circuits computing monotone functions. We start with the simple observation that using standard tools it is easy to construct polynomial-size monotone circuits computing "slice" functions which are pseudorandom on the middle layer of the Boolean cube $\{0,1\}^n$. Such functions are easily seen to be mildly hard to learn, *i.e.,* hard to learn to accuracy $1 - \frac{\Omega(1)}{\sqrt{n}}$. We then use the elegant machinery of hardness amplification of monotone functions which was pioneered by O'Donnell [O'D04] to amplify the hardness of this construction to near-optimal levels (rows 2–4 of Figure 1). We obtain our result for constant depth, sub-polynomial-size circuits (row 5 of Figure 1) by augmenting this approach with an argument which at a high level is similar to one used in [AHM$^+$06], by "scaling down" and modifying our hard-to-learn functions using a variant of Nepomnjaščiĭ's theorem [Nep70].

1.3 Preliminaries

We consider Boolean functions $f : \{0,1\}^n \to \{0,1\}$. We view $\{0,1\}^n$ as endowed with the natural partial order $x \leq y$ iff $x_i \leq y_i$ for all $i = 1, \ldots, n$. A Boolean function f is *monotone* if $x \leq y$ implies $f(x) \leq f(y)$.

We establish that a class $\mathcal{C}$ of functions is hard to learn by showing that for a uniform random $f \in \mathcal{C}$, the expected error of any poly(n)-time learning algorithm L is close to $1/2$ when run with f as the target function. Thus we bound the quantity

$$\Pr_{f \in \mathcal{C}, x \in \{0,1\}^n}[L^f(1^n) \to h; h(x) = f(x)] \tag{1}$$

where the probability is also taken over any internal randomization of the learning algorithm L. We say that *class $\mathcal{C}$ is hard to learn to accuracy* $\frac{1}{2} + \epsilon(n)$ if for every poly(n)-time membership query learning algorithm L (*i.e.,* p.p.t. oracle algorithm), we have (1) $< \frac{1}{2} + \epsilon(n)$ for all sufficiently large n. As noted in [BBL98], it is straightforward to transform a lower bound of this sort into a lower bound for the usual ϵ, δ formulation of PAC learning.

2 Lower Bounds Via Hardness Amplification of Monotone Functions

In this section we prove our main hardness results, summarized in Figure 1, for learning various classes of monotone functions under the uniform distribution with membership queries.

Let us start with a high-level explanation of the overall idea. Inspired by the work on hardness amplification within NP initiated by O'Donnell [O'D04, HVV06], we study constructions of the form

$$f(x_1, \ldots, x_m) = C(f'(x_1), \ldots, f'(x_m))$$

where C is a Boolean "combining function" with low noise stability (we give precise definitions later) which is both *efficiently computable* and *monotone*. Recall that O'Donnell showed that if f' is weakly hard to compute and the combining function C has low noise stability, then f is very hard to compute. This result holds for general (non-monotone) functions C, and thus generalizes Yao's XOR lemma, which addresses the case where C is the XOR of m bits (and hence has the lowest noise stability of all Boolean functions, see [O'D04]).

Roughly speaking, we establish an analogue of O'Donnell's result for learning. Our analogue, given in Section 2.2, essentially states that for certain well-structured[1] functions f' that are hard to learn to high accuracy, if C has low noise stability then f is hard to learn to accuracy even slightly better than $1/2$. Since our ultimate goal is to establish that "simple" classes of monotone functions are hard to learn, we shall use this result with combining functions C that are computed by "simple" monotone Boolean circuits. In order for the overall function f to be monotone and efficiently computable, we need the initial f' to be well-structured, monotone, efficiently computable, and hard to learn to high accuracy. Such functions are easily obtained by a slight extension of an observation of Kearns *et al.* [KLV94]. They noted that the middle slice f' of a random Boolean function on $\{0,1\}^k$ is hard to learn to accuracy greater than $1 - \Theta(1/\sqrt{k})$ [BBL98, KLV94]; by taking the middle slice of a *pseudorandom* function instead, we obtain an f' with the desired properties. In fact, by a result of Berkowitz [Ber82] this slice function is computable by a polynomial-size monotone circuit, so the overall hard-to-learn functions we construct are computed by polynomial-size monotone circuits.

[1] As will be clear, the proof requires that f' be balanced and have a "hard-core set."

Organization. In Section 2.2 we adapt the analysis in [O'D04, HVV06] to reduce the problem of constructing hard-to-learn monotone Boolean functions to constructing monotone combining functions C with low noise stability. In Section 2.3 we show how constructions and analyses from [O'D04, MO03] can be used to obtain a "simple" monotone combining function with low noise stability. In Section 2.4 we establish Theorems 2 and 3 (lines 2 and 3 of Figure 1) by making different assumptions about the hardness of the initial pseudorandom functions. Finally, by making specific number-theoretic assumptions (namely, the hardness of factoring Blum integers), we obtain hard-to-learn monotone Boolean functions that can be computed by very simple circuits (lines 4 and 5 of Figure 1); we defer the formal statements and proofs of these results to the full version.

2.1 Preliminaries

Functions. Let $C : \{0,1\}^m \rightarrow \{0,1\}$ and $f' : \{0,1\}^k \rightarrow \{0,1\}$ be Boolean functions. We write $C \circ f'^{\otimes m}$ to denote the Boolean function over $(\{0,1\}^k)^m$ given by

$$C \circ f'^{\otimes m}(x) = C(f'(x_1), \ldots, f'(x_m)), \qquad \text{where } x = (x_1, \ldots, x_m).$$

For $g : \{0,1\}^k \rightarrow \{0,1\}$, we write $\mathsf{slice}(g)$ to denote the "middle slice" function:

$$\mathsf{slice}(g)(x) = \begin{cases} 1 & \text{if } |x| > \lfloor k/2 \rfloor \\ g(x) & \text{if } |x| = \lfloor k/2 \rfloor \\ 0 & \text{if } |x| < \lfloor k/2 \rfloor. \end{cases}$$

It is immediate that $\mathsf{slice}(g)$ is a monotone Boolean function for any g.

Bias and noise stability. Following the analysis in [O'D04, HVV06], we shall study the bias and noise stability of various Boolean functions. Specifically, we adopt the following notations and definitions from [HVV06]. The *bias* of a 0-1 random variable X is defined to be

$$\text{Bias}[X] \stackrel{\text{def}}{=} |\Pr[X = 0] - \Pr[X = 1]|.$$

Recall that a probabilistic Boolean function h on $\{0,1\}^k$ is a probability distribution over Boolean functions on $\{0,1\}^k$ (so for each input x, the output $h(x)$ is a 0-1 random variable). The *expected bias* of a probabilistic Boolean function h is

$$\text{ExpBias}[h] \stackrel{\text{def}}{=} \mathrm{E}_x[\text{Bias}[h(x)]].$$

Let $C : \{0,1\}^m \rightarrow \{0,1\}$ be a Boolean function and $0 \leq \delta \leq \frac{1}{2}$. The *noise stability of* C *at noise rate* δ, denoted $\text{NoiseStab}_\delta[C]$, is defined to be

$$\text{NoiseStab}_\delta[C] \stackrel{\text{def}}{=} 2 \cdot \Pr_{x,\eta}[C(x) = C(x \oplus \eta)] - 1$$

where $x \in \{0,1\}^m$ is uniform random, $\eta \in \{0,1\}^m$ is a vector whose bits are each independently 1 with probability δ, and $\oplus$ denotes bitwise XOR.

2.2 Hardness Amplification for Learning

Throughout this subsection we write m for $m(n)$ and k for $k(n)$. We shall establish the following:

Lemma 1. *Let $C : \{0,1\}^m \rightarrow \{0,1\}$ be a polynomial-time computable function. Let $\mathcal{G}'$ be the family of all 2^{2^k} functions from $\{0,1\}^k$ to $\{0,1\}$, where $n = mk$ and $k = \omega(\log n)$. Then the class $\mathcal{C} = \{f = C \circ \mathsf{slice}(g)^{\otimes m} \mid g \in \mathcal{G}'\}$ of Boolean functions over $\{0,1\}^n$ is hard to learn to accuracy*

$$\frac{1}{2} + \frac{1}{2}\sqrt{\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C]} + o(1/n).$$

This easily yields Corollary 1, which is an analogue of Lemma 1 with pseudorandom rather than truly random functions, and which we use to obtain our main hardness of learning results.

Proof of Lemma 1: Let k, m be such that $mk = n$, and let $C : \{0,1\}^m \rightarrow \{0,1\}$ be a Boolean combining function. We prove the lemma by upper bounding

$$\Pr_{g \in \mathcal{G}', x \in \{0,1\}^n}\Big[L^f(1^n) \rightarrow h;\ h(x) = f(x)\Big] \tag{2}$$

where L is an arbitrary p.p.t. oracle machine (running in time poly(n) on input 1^n) that is given oracle access to $f \overset{\mathrm{def}}{=} C \circ \mathsf{slice}(g)^{\otimes m}$ and outputs some hypothesis $h : \{0,1\}^n \rightarrow \{0,1\}$.

We first observe that since C is computed by a uniform family of circuits of size poly$(m) \le$ poly(n), it is easy for a poly(n)-time machine to simulate oracle access to f if it is given oracle access to g. So (2) is at most

$$\Pr_{g \in \mathcal{G}',\ x \in \{0,1\}^n}\Big[L^g(1^n) \rightarrow h;\ h(x) = (C \circ \mathsf{slice}(g)^{\otimes m})(x)\Big]. \tag{3}$$

To analyze the above probability, suppose that in the course of its execution L never queries g on any of the inputs $x_1, \ldots, x_m \in \{0,1\}^k$, where $x = (x_1, \ldots, x_m)$. Then the *a posteriori* distribution of $g(x_1), \ldots, g(x_m)$ (for uniform random $g \in \mathcal{G}'$) given the responses to L's queries that it received from g is identical to the distribution of $g'(x_1), \ldots, g'(x_m)$, where g' is an independent uniform draw from $\mathcal{G}'$: both distributions are uniform random over $\{0,1\}^m$. (Intuitively, this just means that if L never queries the random function g on any of $x_1, \ldots, x_m$, then giving L oracle access to g does not help it predict the value of f on $x = (x_1, \ldots, x_m)$.) Since L runs in poly(n) time, for any fixed $x_1, \ldots, x_m$ the probability that L queried g on any of $x_1, \ldots, x_m$ is at most $\frac{m \cdot \mathrm{poly}(n)}{2^k}$. Hence (3) is bounded by

$$\Pr_{g, g' \in \mathcal{G}',\ x \in \{0,1\}n}\Big[L^g(1^n) \rightarrow h;\ h(x) = (C \circ \mathsf{slice}(g')^{\otimes m})(x)\Big] + \frac{m \cdot \mathrm{poly}(n)}{2^k}. \tag{4}$$

The first summand in (4) is the probability that L correctly predicts the value $C \circ \mathsf{slice}(g')^{\otimes m}(x)$, given oracle access to g, where g and g' are independently random functions and x is uniform over $\{0,1\}^n$. It is clear that the best possible strategy for L is to use a maximum likelihood algorithm, *i.e.,* predict according to the function h which, for any fixed input x, outputs 1 if and only if the random variable $(C \circ \mathsf{slice}(g')^{\otimes m})(x)$ (we emphasize that the randomness here is over the choice of g') is biased towards 1. The expected accuracy of this h is precisely

$$\frac{1}{2} + \frac{1}{2}\,\mathrm{ExpBias}[C \circ \mathsf{slice}(g')^{\otimes m}]. \tag{5}$$

Now fix $\delta \overset{\mathrm{def}}{=} \binom{k}{\lfloor k/2 \rfloor}/2^k = \Theta(1/\sqrt{k})$ to be the fraction of inputs in the "middle slice" of $\{0,1\}^k$. We observe that the probabilistic function $\mathsf{slice}(g')$ (where g' is truly random) is "δ-random" in the sense of ([HVV06], Definition 3.1), *i.e.,* it is balanced, truly random on inputs in the middle slice, and deterministic on all other inputs. This means that we may apply a technical lemma [HVV06, Lemma 3.7]) to $\mathsf{slice}(g')$ (see also [O'D04]) to obtain

$$\mathrm{ExpBias}[C \circ \mathsf{slice}(g')^{\otimes m}] \le \sqrt{\mathrm{NoiseStab}_\delta[C]}. \tag{6}$$

Combining (4), (5) and (6) and recalling that $k = \omega(\log n)$, we obtain Lemma 1. □

Corollary 1. *Let $C : \{0,1\}^m \to \{0,1\}$ be a polynomial-time computable function. Let $\mathcal{G}$ be a pseudorandom family of functions from $\{0,1\}^k$ to $\{0,1\}$ which are secure against* $\mathrm{poly}(n)$*-time adversaries, where $n = mk$ and $k = \omega(\log n)$. Then the class $\mathcal{C} = \{f = C \circ \mathsf{slice}(g)^{\otimes m} \mid g \in \mathcal{G}\}$ of Boolean functions over $\{0,1\}^n$ is hard to learn to accuracy*

$$\frac{1}{2} + \frac{1}{2}\sqrt{\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C]} + o(1/n).$$

Proof. The corollary follows from the fact that (3) must differ from its pseudorandom counterpart,

$$\Pr_{g \in \mathcal{G},\, x \in \{0,1\}^n}\Big[L^g(1^n) \to h;\ h(x) = (C \circ \mathsf{slice}(g)^{\otimes m})(x)\Big], \tag{7}$$

by less than $1/n^2$ (in fact by less than any fixed $1/\mathrm{poly}(n)$). Otherwise, we would easily obtain a $\mathrm{poly}(n)$-time distinguisher that, given oracle access to g, runs L to obtain a hypothesis h and checks whether $h(x) = (C \circ \mathsf{slice}(g)^{\otimes m})(x)$ for a random x to determine whether g is drawn from $\mathcal{G}$ or $\mathcal{G}'$. □

By instantiating Corollary 1 with a "simple" monotone function C having low noise stability, we obtain strong hardness results for learning simple monotone functions. We exhibit such a function C in the next section.

2.3 A Simple Monotone Combining Function with Low Noise Stability

In this section we combine known results of [O'D04, MO03] to obtain:

Proposition 1. *Given a value k, let $m = 3^\ell d2^d$ for ℓ, d satisfying $3^\ell \le k^6 < 3^{\ell+1}$ and $d \le O(k^{.35})$. Then there exists a monotone function $C : \{0,1\}^m \to \{0,1\}$ computed by a uniform family of* poly(m)*-size,* $\log(m)$*-depth monotone circuits such that*

$$\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C] \le O\Big(\frac{k^6 \log m}{m}\Big). \tag{8}$$

Note that in this proposition we may have m as large as $2^{\Theta(k^{.35})}$ but not larger. O'Donnell[O'D04] gave a lower bound of $\Omega(\frac{\log^2 m}{m})$ on $\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C]$ for every monotone m-variable function C, so the above upper bound is fairly close to the best possible (within a polylog(m) factor if $m = 2^{k^{\Theta(1)}}$).

Following [O'D04, HVV06], we use the "recursive majority of 3" function and the "tribes" function in our construction. We require the following results on noise stability:

Lemma 2 ([O'D04]). *Let* $\text{Rec-Maj-3}_\ell : \{0,1\}^{3^\ell} \to \{0,1\}$ *be defined as follows: for $x = (x^1, x^2, x^3)$ where each $x^i \in \{0,1\}^{3^{\ell-1}}$,*

$$\text{Rec-Maj-3}_\ell(x) \overset{def}{=} \mathrm{Maj}(\text{Rec-Maj-3}_{\ell-1}(x^1), \text{Rec-Maj-3}_{\ell-1}(x^2), \text{Rec-Maj-3}_{\ell-1}(x^3)).$$

Then for $\ell \ge \log_{1.1}(1/\delta)$, we have $\mathrm{NoiseStab}_\delta[\text{Rec-Maj-3}_\ell] \le \delta^{-1.1}(3^\ell)^{-.15}$.

Lemma 3 ([MO03]). *Let* $\mathrm{Tribes}_d : \{0,1\}^{d2^d} \to \{0,1\}$ *denote the "tribes" function on $d2^d$ variables, i.e., the read-once 2^d-term monotone d-DNF*

$$\mathrm{Tribes}_d(x_1, \ldots, x_{d2^d}) \overset{def}{=} (x_1 \wedge \cdots \wedge x_d) \vee (x_{d+1} \wedge \cdots \wedge x_{2d}) \vee \cdots .$$

Then if $\eta \le O(1/d)$, we have $\mathrm{NoiseStab}_{\frac{1-\eta}{2}}[\mathrm{Tribes}_d] \le O\Big(\frac{\eta d^2}{d2^d}\Big) \le O\Big(\frac{1}{2^d}\Big)$.

Lemma 4 ([O'D04]). *If h is a balanced Boolean function and $\psi : \{0,1\}^r \to \{0,1\}$ is arbitrary, then for any δ we have*

$$\mathrm{NoiseStab}_\delta[\psi \circ h^{\otimes r}] = \mathrm{NoiseStab}_{\frac{1}{2} - \frac{\mathrm{NoiseStab}_\delta[h]}{2}}[\psi].$$

Proof of Proposition 1: We take C to be $\mathrm{Tribes}_d \circ \text{Rec-Maj-3}_\ell^{\otimes d2^d}$. Since Rec-Maj-3_ℓ is balanced, by Lemma 4 we have

$$\mathrm{NoiseStab}_\delta[C] = \mathrm{NoiseStab}_{\frac{1}{2} - \frac{\mathrm{NoiseStab}_\delta[\text{Rec-Maj-3}_\ell]}{2}}[\mathrm{Tribes}_d].$$

Setting $\delta = \Theta(1/\sqrt{k})$ and recalling that $3^\ell \le k^6$, we have $\ell \ge \log_{1.1}(1/\delta)$ so we may apply Lemma 2 to obtain

$$\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[\text{Rec-Maj-3}_\ell] \le \Theta((\sqrt{k})^{1.1})(k^6)^{-.15} = O(k^{-.35}).$$

Since $O(k^{-.35}) \le O(1/d)$, we may apply Lemma 3 with the previous inequalities to obtain

$$\mathrm{NoiseStab}_{\Theta(1/\sqrt{k})}[C] \le O\Big(\frac{1}{2^d}\Big).$$

The bound (8) follows from some easy rearrangement of the bounds on k, m, d and ℓ. It is easy to see that C can be computed by monotone circuits of depth $O(\ell) = O(\log m)$ and size poly(m), and the proposition is proved. $\square$

2.4 Nearly Optimal Hardness of Learning Polynomial-Size Monotone Circuits

Given a value of k, let $m = 3^\ell d 2^d$ for ℓ, d as in Proposition 1. Let $\mathcal{G}$ be a pseudorandom family of functions $\{g : \{0,1\}^k \to \{0,1\}\}$ secure against poly(n)-time adversaries, where $n = mk$. Since we have $k = \omega(\log n)$, we may apply Corollary 1 with the combining function from Proposition 1 and conclude that the class $\mathcal{C} = \{C \circ \mathsf{slice}(g)^{\otimes m} \mid g \in \mathcal{G}\}$ is hard to learn to accuracy

$$\frac{1}{2} + O\Big(\frac{k^3\sqrt{\log m}}{\sqrt{m}}\Big) + o(1/n) \leq \frac{1}{2} + O\Big(\frac{k^{3.5}\sqrt{\log n}}{\sqrt{n}}\Big). \tag{9}$$

We claim that in fact the functions in $\mathcal{C}$ can be computed by poly(n)-size monotone circuits. This follows from a result of Berkowitz [Ber82] which states that if a k-variable slice function is computed by a AND/OR/NOT circuit of size s and depth d, then it is also computed by a monotone AND/OR/MAJ circuit of size $O(s + k)$ and depth $d + 1$. Combining these monotone circuits for $\mathsf{slice}(g)$ with the monotone circuit for C, we obtain a poly(n)-size monotone circuit for each function in $\mathcal{C}$.

By making different assumptions on the hardness of pseudorandom function families (in fact, the assumptions may be made about one-way functions–a more formal treatment is given in the full version), we obtain quantitative relationships between k (the input length for the pseudorandom functions) and n (the running time of the adversaries against which they are secure), and thus different quantitative hardness results from (9) above:

Theorem 2. *Suppose that standard one-way functions exist. Then for any constant $\epsilon > 0$ there is a class $\mathcal{C}$ of poly(n)-size monotone circuits that is hard to learn to accuracy $\frac{1}{2} + \frac{1}{n^{1/2-\epsilon}}$.*

Proof. If poly(n)-hard one-way functions exist then for any arbitrarily small constant c there exists a pseudorandom family of functions from $\{0,1\}^k \to \{0,1\}$ with $k = n^c$; this corresponds to taking $d = C \log k$ for C a large constant in Proposition 1. The claimed bound on (9) easily follows. (We note that while not every n is of the required form $mk = 3^\ell d 2^d k$, it is not difficult to see that this and our subsequent theorems hold for all (sufficiently large) input lengths n by padding the hard-to-learn functions.) □

Theorem 3. *Suppose that subexponentially hard (2^{n^α} for some fixed $\alpha > 0$) one-way functions exist. Then there is a class $\mathcal{C}$ of poly(n)-size monotone circuits that is hard to learn to accuracy $\frac{1}{2} + \frac{\mathrm{polylog}(n)}{n^{1/2}}$.*

Proof. As above, but now we take $k = \log^C n$ for some sufficiently large constant C (*i.e.*, $d = c \log k$ for a small constant c). □

References

[AHM$^+$06] Allender, E., Hellerstein, L., McCabe, P., Pitassi, T., Saks, M.: Minimizing DNF Formulas and AC_d^0 Circuits Given a Truth Table. In: CCC, pp. 237–251 (2006)

[BBL98] Blum, A., Burch, C., Langford, J.: On learning monotone boolean functions. In: 39th FOCS, pp. 408–415 (1998)

[Ber82] Berkowitz, S.J.: On some relationships between monotone and non-monotone circuit complexity. Technical Report, University of Toronto (1982)

[BFKL93] Blum, A., Furst, M., Kearns, M., Lipton, R.: Cryptographic Primitives Based on Hard Learning Problems. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1993)

[BT96] Bshouty, N., Tamon, C.: On the Fourier spectrum of monotone functions. Journal of the ACM 43(4), 747–770 (1996)

[HVV06] Healy, A., Vadhan, S., Viola, E.: Using Nondeterminism to Amplify Hardness. SIAM Journal on Computing 35(4), 903–931 (2006)

[Kha93] Kharitonov, M.: Cryptographic hardness of distribution-specific learning. In: 25th STOC, pp. 372–381 (1993)

[Kha95] Kharitonov, M.: Cryptographic lower bounds for learnability of Boolean functions on the uniform distribution. JCSS 50, 600–610 (1995)

[KKL88] Kahn, J., Kalai, G., Linial, N.: The influence of variables on boolean functions. In: 29th FOCS, pp. 68–80 (1988)

[KLV94] Kearns, M.J., Li, M., Valiant, L.G.: Learning boolean formulas. J. ACM 41(6), 1298–1328 (1994)

[KOS04] Klivans, A., O'Donnell, R., Servedio, R.: Learning intersections and thresholds of halfspaces. JCSS 68(4), 808–840 (2004)

[LMN93] Linial, N., Mansour, Y., Nisan, N.: Constant depth circuits, Fourier transform and learnability. Journal of the ACM 40(3), 607–620 (1993)

[Man94] Mansour, Y.: Learning Boolean functions via the Fourier transform, pp. 391–424. Kluwer Academic Publishers, Dordrecht (1994)

[MO03] Mossel, E., O'Donnell, R.: On the noise sensitivity of monotone functions. Random Struct. Algorithms 23(3), 333–350 (2003)

[MOS04] Mossel, E., O'Donnell, R., Servedio, R.: Learning functions of k relevant variables. J. Comput. & Syst. Sci. 69(3), 421–434 (2004)

[Nep70] Nepomnjaščiĭ, V.A.: Rudimentary predicates and Turing calculations. Math Dokl. 11, 1462–1465 (1970)

[NR04] Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. Journal of the ACM 51(2), 231–262 (2004)

[O'D04] O'Donnell, R.: Hardness amplification within NP. JCSS 69(1), 68–94 (2004)

[OS07] O'Donnell, R., Servedio, R.: Learning monotone decision trees in polynomial time. SIAM Journal on Computing 37(3), 827–844 (2007)

[Raz85] Razborov, A.: Lower bounds on the monotone network complexity of the logical permanent. Mat. Zametki 37, 887–900 (1985)

[Ser04] Servedio, R.: On learning monotone DNF under product distributions. Information and Computation 193(1), 57–74 (2004)

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

[Ver90] Verbeurgt, K.: Learning DNF under the uniform distribution in quasi-polynomial time. In: 3rd COLT, pp. 314–326 (1990)