Electronic Colloquium on Computational Complexity, Report No. 22 (2010)

Lower Bounds and Hardness Amplification for Learning Shallow Monotone Formulas

Vitaly Feldman
IBM Almaden Research Center
vitaly@post.harvard.edu

Homin K. Lee
UT Austin
homin@cs.utexas.edu

Rocco A. Servedio*
Columbia University
rocco@cs.columbia.edu

February 20, 2010

Abstract

Much work has been done on learning various classes of "simple" monotone functions under the uniform distribution. In this paper we give the first unconditional lower bounds for learning problems of this sort by showing that polynomial-time algorithms cannot learn constant-depth monotone Boolean formulas under the uniform distribution in the well-studied Statistical Query model.

Using a recent characterization of Strong Statistical Query learnability due to Feldman [15], we first show that depth-3 monotone formulas of size $n^{o(1)}$ cannot be learned by any polynomial-time Statistical Query algorithm to accuracy $1-1/(\log n)^{\Omega(1)}$. We then build on this result to show that depth-4 monotone formulas of size $n^{o(1)}$ cannot be learned even to a certain $\frac{1}{2}+o(1)$ accuracy in polynomial time. This improved hardness is achieved using a general technique that we introduce for amplifying the hardness of "mildly hard" learning problems in either the PAC or Statistical Query framework. This hardness amplification for learning builds on the ideas in the work of O'Donnell [28] on hardness amplification for approximating functions using small circuits, and may be of independent interest.

*Supported by NSF grants CCF-0347282, CCF-0523664 and CNS-0716245, and by DARPA award HR0011-08-1-0069.

ISSN 1433-8092

1 Introduction

Motivation. Over the past several decades much work in computational learning theory has focused on developing efficient algorithms for learning monotone Boolean functions under the uniform distribution, see *e.g.,* [1, 3, 10, 17, 19, 22, 27, 29, 30, 34] and other works. An intriguing question, which has driven much of this research and remains open, is whether there is an efficient algorithm to learn *monotone DNF formulas* under the uniform distribution. Such an algorithm A would have the following performance guarantee: for any target function $f : \{0,1\}^n \to \{0,1\}$ that is a monotone DNF formula with poly(n) terms, given access to independent uniform random examples $(x, f(x))$, algorithm A would run in poly($n, 1/\epsilon$) time and with high probability output a hypothesis h that disagrees with f on at most an ϵ fraction of inputs from $\{0,1\}^n$.

Several partial positive results toward learning monotone DNF have been obtained: for constant ϵ, algorithms are known that can learn $2^{\sqrt{\log n}}$-term monotone DNF [34] and poly(n)-size monotone decision trees [27] in poly(n) time. Partial negative results have also been given: [13] has shown that (under a strong cryptographic hardness assumption) for a sufficiently large absolute constant d there is no poly(n)-time algorithm that can learn depth-d, size-$n^{o(1)}$ Boolean formulas that compute monotone functions to a certain accuracy $\frac{1}{2} + o(1)$. Given these partial results, it is of considerable interest to obtain refined results on the efficient learnability of simple monotone functions.

In this work we give *unconditional* lower bounds showing that simple monotone functions – computed by monotone Boolean formulas of depth 3 or 4 and size $n^{o(1)}$ – cannot be learned under the uniform distribution in polynomial time. Of course these results are not in the PAC model of learning from random examples (since unconditional lower bounds in this model would prove $P \neq NP$!); instead we work in the well-studied *Statistical Query* learning model, which we describe briefly below.

Statistical Query learning. Kearns [21] introduced the *statistical query (SQ)* learning model as a natural variant of the usual PAC learning model. In the SQ model, instead of having access to independent random examples $(x, f(x))$ drawn from distribution $\mathcal{D}$, the learner has access to a *statistical query oracle* $SQ_{f,\mathcal{D}}$. The oracle $SQ_{f,\mathcal{D}}$ takes as input a *query function* $g : X \times \{+1, -1\} \to \{+1, -1\}$ and a *tolerance parameter* $\tau \in [0, 1]$ and outputs a value v such that:

$$|v - \mathbf{E}_{\mathcal{D}}[g(x, f(x))]| \leq \tau.$$

The learner's goal – to output a hypothesis h such that $\Pr_{x\sim\mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$ – is the same as in PAC learning. A poly($n, 1/\epsilon$)-time SQ algorithm is only allowed to make queries in which g can be computed by a poly($n, 1/\epsilon$)-size circuit and τ is at most a fixed $1/\text{poly}(n, 1/\epsilon)$ (and of course the algorithm must run for at most poly($n, 1/\epsilon$) time steps).

The SQ model is an important and well-studied learning model which has received much research attention in the 15 years since it was introduced, see *e.g.,* [2, 4, 5, 41, 9, 12, 11, 14, 16, 25, 20, 39, 40] and many other works.[1] One reason for this intense interest is that any concept class that is efficiently learnable from statistical queries is also efficiently PAC learnable in the presence of random classification noise at any noise rate bounded away from $\frac{1}{2}$ [21]. In fact, since the introduction of the SQ-model virtually all known noise-tolerant learning algorithms have been obtained from (or rephrased as) SQ algorithms.[2] Thus the study of SQ learning is now an integral part of the study of noise-tolerant learning and of PAC learning in general.

For us, an attractive feature of the SQ-model is that it is a powerful and well-studied learning model within which it is possible to prove *unconditional* information-theoretic lower bounds showing that certain

[1] The SQ-model also has deep connections to other areas such as communication complexity, see *e.g.,* [35].

[2] One prominent exception is the work of [7], which gives an algorithm for learning parities which is tolerant to random noise, although in a weaker sense than the algorithms derived from Statistical Queries.

classes of functions cannot be learned in polynomial time (we will say much more about this below). As Kearns [21] and others have shown, many PAC learning algorithms can be converted into Statistical Query algorithms (with algorithms based on Gaussian elimination for learning parities being virtually the only exception). Thus an SQ lower bound for a class of functions gives a very strong indication that the class is likely to be difficult to learn using virtually all known techniques for learning from random examples.

Background on hardness results for SQ learning. In his paper introducing the SQ model [21], Kearns already showed that the class of all parity functions cannot be SQ-learned in polynomial time under the uniform distribution. Soon after this Blum *et al.* [6] characterized the weak learnability of every function class $\mathcal{F}$ in the SQ model in terms of the *statistical query dimension* of $\mathcal{F}$; roughly speaking, this is the largest number of functions from $\mathcal{F}$ that are pairwise nearly orthogonal to each other (we give a precise definition in Section 2). The results of [6] imply that if a class $\mathcal{F}$ has SQ-Dimension $n^{\omega(1)}$, then no SQ algorithm can even weakly learn $\mathcal{F}$ to any accuracy $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ in poly(n) time. This bound was already used to give SQ hardness results for weak learning classes such as DNF and decision trees in [6], and more recently for weak-learning intersections of halfspaces in [25]. However, it is well known that the entire class of all monotone Boolean functions over $\{0,1\}^n$ *can* be weakly learned to accuracy $\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}$ in poly(n) time (an algorithm that achieves optimal accuracy $\frac{1}{2} + \frac{\Theta(\log n)}{\sqrt{n}}$ was recently given in [29]), and indeed the class of all monotone functions can easily be shown to have SQ-dimension $O(n)$. Thus the notion of SQ-dimension alone is not enough to yield SQ lower bounds on learning monotone functions.

Much more recently, Simon [36] introduced a combinatorial parameter of a function class $\mathcal{F}$ called its *strong Statistical Query dimension*, and showed that this parameter at error rate ϵ characterizes the information-theoretic *strong* learnability of $\mathcal{F}$ to accuracy $1 - \epsilon$. (We give a precise definition of the strong SQ-dimension in Section 2.) This characterization, which was later simplified by Feldman [15] (and independently by Szörényi [37]), plays a crucial role in our results, which we now describe. (Throughout the following description of our results, the underlying distribution is always taken to be uniform over $\{0,1\}^n$.)

Our Results: Unconditional Hardness of Learning Simple Monotone Functions. We give the first strong SQ-dimension lower bound for a class of "simple" monotone functions. More precisely, as our first main result, we show that the class of size-$n^{o(1)}$, depth-3 monotone formulas has strong SQ-dimension $n^{\omega(1)}$ at a certain error rate $1/(\log n)^{\Theta(1)}$. By the results of Simon and Feldman, this implies that such formulas cannot be efficiently learned to accuracy $1 - 1/(\log n)^{\Theta(1)}$ by any polynomial-time SQ learning algorithm. Roughly speaking, our proof works by constructing a class of slice functions of "well-separated" parities over poly $\log(n)$ variables. We show that this class of functions has the (somewhat delicate) combinatorial properties required to satisfy the strong SQ-dimension criterion, and that every function in the class can be computed by a small monotone formula of depth 3.

This result is the first unconditional lower bound known for SQ learning "simple" monotone functions, and is intriguingly close to a lower bound for SQ learning monotone DNF formulas (which are of course depth-2 rather than depth-3 monotone formulas). However, these results only say that monotone depth-3 formulas cannot be SQ-learned to a rather high $(1 - o(1))$ accuracy. Thus a natural goal is to obtain stronger hardness results which show that simple monotone functions are hard to learn even to coarse accuracy – ideally to some accuracy level $\frac{1}{2} + o(1)$ only slightly better than random guessing. Of course, we might expect that to achieve this we must use somewhat more complicated functions than depth-3 formulas, and this does turn out to be the case – but only a bit more complicated, as we describe below.

We introduce a general method of amplifying the hardness of a class of functions that are "mildly hard to learn" (*i.e.,* hard to learn to high accuracy), to obtain a class of functions that are "very hard to learn" (*i.e.,* hard to learn to accuracy even slightly better than random guessing). We show that our method, which

builds on O'Donnell's hardness amplification for approximating Boolean functions using small circuits, can be applied both within the uniform-distribution PAC model and within the uniform-distribution Statistical Query model. The latter is of course our main interest in this paper, but we feel that the technique in general may be of independent interest and find further applications elsewhere. We note that our hardness amplification results require several refinements of the argument of O'Donnell: we show how randomness can be used in place of nonuniformity in [28], and how "smooth boosting" algorithms can be used in place of Impagliazzo's hard-core lemma. In particular, new ideas are required to successfully translate the high-level ideas of [28] into the SQ setting (see Section 4.2).

Using this hardness amplification for SQ learning together with our first main result, we obtain our second main result: we show that the class of size-$n^{o(1)}$, depth-4 monotone formulas cannot be SQ-learned even to $\frac{1}{2} + 2^{-(\log n)^{\gamma}}$ accuracy in poly(n) time for any $\gamma < 1/2$. We are able to increase the depth by only one (from 3 to 4) by a careful choice of the combining function in our hardness amplification framework; we use a depth-2 combining function due to Talagrand which has useful extremal noise stability properties as shown by O'Donnell and Mossel [26].

Relation to previous work. To the best of our knowledge even the "mild hardness" result that we prove for depth-3 monotone formulas is the first unconditional negative result known for learning a class of polynomial-time computable monotone functions in the uniform-distribution SQ model. We note that the strong $\frac{1}{2} + o(1)$ hardness that we establish for depth-4 monotone formulas is provably near-optimal, since as mentioned earlier the class of *all* monotone functions over $\{0,1\}^n$ can be learned to accuracy $\frac{1}{2} + \frac{\Theta(\log n)}{\sqrt{n}}$ in polynomial time [29].

While the recent work in [13] also gave negative results for learning constant-depth monotone formulas, those results are different from ours in significant ways. [13] used a strong cryptographic hardness assumption – that Blum integers are $2^{n^{\epsilon}}$-hard to factor on average for some fixed $\epsilon > 0$ – to show that for some sufficiently large absolute constant d, the class of monotone functions computed by size-$n^{o(1)}$, depth-d formulas cannot be PAC learned, under the uniform distribution, to a certain accuracy $\frac{1}{2} + o(1)$. In contrast, our hardness result applies to the more restricted class of size-$n^{o(1)}$, depth-4 *monotone* formulas, and gives *unconditional* hardness for polynomial-time algorithms in the Statistical Query model.

Finally, we remark here that our hardness amplification method for PAC and SQ learning may be viewed as a significant strengthening and generalization of some earlier results. Boneh and Lipton [8] described a form of uniform-distribution hardness amplification for PAC learning based on the XOR lemma; our PAC hardness amplification generalizes their results. More recently, [13] dealt with a very special case of hardness amplification in which the "mildly hard" class of functions $\mathcal{F}$ consists of all functions of the form slice(f), where f may be any Boolean function and slice(f) is the function which agrees with Majority everywhere except the middle layer of the Boolean hypercube. An easy argument shows that $\mathcal{F}$ is a class of monotone functions that is hard to learn to accuracy $1 - \Theta(1)/\sqrt{n}$. Using the fact that a random function in $\mathcal{F}$ is trivial to predict off of the middle layer and is totally random on the middle layer, simplified versions of arguments from [28] are used in [13] to show information-theoretic hardness of learning the combined function class $\mathcal{F}^g$. In contrast, the hardness amplification results of this paper are far more general, since we do not assume any particular structure of the base class $\mathcal{F}$, only that it is "mildly hard to learn."

Organization. Section 2 gives background on Statistical Query learning, the SQ-dimension, and the strong SQ-dimension. In Section 3 we describe our class of depth-3 monotone formulas and show that it is "mildly" hard to learn in the SQ model by giving a superpolynomial lower bound on its strong SQ-dimension. Section 4 presents our general hardness amplification results for the uniform-distribution PAC model and the uniform-distribution Statistical Query learning model. We apply our hardness amplification technique from

Section 4 to obtain our second main result, strong SQ hardness for depth-4 formulas, in Section 5.

2 The Statistical Query Model, SQ-Dimension, and Strong SQ-Dimension

Recall the definition of Statistical Query learning from Section 1. Blum *et al.* [6] introduced the notion of the *SQ-dimension of function class $\mathcal{F}$ under distribution $\mathcal{D}$*, and showed that it characterizes the weak-learnability of $\mathcal{F}$ under $\mathcal{D}$ in the SQ model. Bshouty and Feldman [9] and Yang [40] later generalized and sharpened the Blum *et al.* result. We will use Yang's version here extended to sets of arbitrary real-valued functions. We write "$\langle f, g\rangle_{\mathcal{D}}$" to denote $\mathbf{E}_{x\sim\mathcal{D}}[f(x)g(x)]$ and "$\|f\|_{\mathcal{D}}$" to denote $(\langle f, f\rangle_{\mathcal{D}})^{1/2}$.

Definition 2.1 *Given a set C of real-valued functions, the* SQ-dimension *of C with respect to $\mathcal{D}$ (written SQ-DIM$(C, \mathcal{D})$) is the largest number d such that $\exists\{f_1, \ldots, f_d\} \subseteq C$ with the property that $\forall i \neq j$,*

$$\left|\langle f_i, f_j\rangle_{\mathcal{D}}\right| \leq \frac{1}{d}. \tag{1}$$

When $\mathcal{D}$ is the uniform distribution we simply write SQ-DIM(C). We refer to the LHS of Equation (1) as the *correlation* between f_i and f_j under $\mathcal{D}$.

Intuitively, this condition says that C contains d "nearly-uncorrelated" functions. It is easy to see that if C is a concept class with SQ-DIM$(C, \mathcal{D}) = d$ then C can be weakly learned with respect to $\mathcal{D}$ to accuracy $\frac{1}{2} + \frac{\Theta(1)}{d}$ using d Statistical Queries with tolerance $\frac{\Theta(1)}{d}$; simply ask for the correlation between the unknown target function f and each function in the set $\{f_1, \ldots, f_d\}$. Since the set is maximal, the target function must have correlation at least $1/d$ with at least one of the functions.

Blum *et al.* showed that the other direction is true as well; if C is efficiently weakly learnable, then C must have small SQ-dimension.

Theorem 2.2 ([6] Theorem 12) *Given a concept class C and a distribution $\mathcal{D}$, let SQ-DIM$(C, \mathcal{D}) = d$. Then if the tolerance τ of each query is always at least $1/d^{1/3}$, at least $\frac{1}{2}d^{1/3} - 1$ queries are required to learn C with advantage $1/d^3$.*

As an example, the class *PAR* of all parity functions over n variables has SQ-DIM(*PAR*) $= 2^n$, and thus any SQ algorithm for learning parities over the uniform distribution $\mathcal{U}$ to accuracy $\frac{1}{2} + \frac{1}{2^{O(n)}}$ requires exponential time.

2.1 The Strong SQ-Dimension

The statistical query dimension only characterizes the weak SQ-learnability of a class and is not sufficient to characterize its strong SQ-learnability. The first characterization of strong SQ learning was given by Simon [36], but for our application a subsequent accuracy-preserving characterization by Feldman will be more convenient to use [15].

Let $\mathcal{F}_1^\infty$ denote the set of all functions from $\{0,1\}^n \to [-1,1]$, *i.e.,* all functions with L_∞-norm bounded by 1. For a Boolean function f, we define $B_{\mathcal{D}}(f, \epsilon)$ to be $\{g : \{0,1\}^n \to \{-1,1\} : \mathbf{Pr}_{\mathcal{D}}[g \neq f] \leq \epsilon\}$, *i.e.,* the ϵ-ball around f, and $\mathbf{0}$ to be the constant 0 function. The sign function is defined as $\mathtt{sign}(z) = 1$ for $z \geq 0$, $\mathtt{sign}(z) = -1$ for $z < 0$. Finally, for a set of real-valued functions C, let $C - g = \{f - g : f \in C\}$.

Definition 2.3 *Given a concept class C and $\epsilon > 0$, the* strong SQ-dimension *of C with respect to $\mathcal{D}$ is defined to be:*

$$\textit{SQ-SDIM}(C, \mathcal{D}, \epsilon) = \sup_{g\in\mathcal{F}_1^\infty} \textit{SQ-DIM}((C \setminus B_{\mathcal{D}}(\mathtt{sign}(g), \epsilon)) - g, \mathcal{D}).$$

Just as for the weak SQ-dimension, the strong SQ-dimension completely characterizes the strong SQ-learnability of a concept class.

Theorem 2.4 ([15]) *Let C be a concept class over $\{0,1\}^n$, $\mathcal{D}$ be a probability distribution over $\{0,1\}^n$ and $\epsilon > 0$. If there exists a polynomial $p(\cdot,\cdot)$ such that C is SQ learnable over $\mathcal{D}$ to accuracy ϵ from $p(n, 1/\epsilon)$ queries of tolerance $1/p(n, 1/\epsilon)$ then SQ-SDIM$(C, \mathcal{D}, \epsilon + 1/p(n, 1/\epsilon)) \leq p'(n, 1/\epsilon)$ for some polynomial $p'(\cdot,\cdot)$. Further, if SQ-SDIM$(C, \mathcal{D}, \epsilon) \leq q(n, 1/\epsilon)$ for some polynomial $q(\cdot,\cdot)$ then C is SQ learnable over $\mathcal{D}$ to accuracy ϵ from $q'(n, 1/\epsilon)$ queries of tolerance $1/q'(n, 1/\epsilon)$ for some polynomial $q'(\cdot,\cdot)$.*

Armed with Definition 2.3 and Theorem 2.4, we can show that a concept class C is not polynomial-time learnable to high accuracy by choosing a suitable $\epsilon = \Omega(1/\operatorname{poly}(n))$ and a suitable function $g \in \mathcal{F}_1^\infty$ and proving that SQ-DIM$((C \setminus B_{\mathcal{U}}(\mathtt{sign}(g), 2\epsilon)) - g) = n^{\omega(1)}$ (we can assume without loss of generality that ϵ upper bounds the tolerance of an SQ algorithm). We do just this, for a class of depth-3 monotone formulas, in the next section.

3 Strong SQ Lower Bound for Depth-3 Monotone Formulas

In this section we exhibit a family of monotone functions that cannot be strong SQ-learned in polynomial time under the uniform distribution. The high-level idea is that we embed a family of non-monotone functions with high SQ-dimension – a family of parity functions – into the middle level of the k-dimensional Boolean cube, and thus obtain a class of monotone functions with high strong SQ-dimension.

A *k-variable slice function for f*, where f is a real-valued function over $\{0,1\}^k$, is denoted slice_f. For $x \in \{0,1\}^k$ the value of $\text{slice}_f(x)$ is 1 if x has more than $\lceil k/2 \rceil$ ones, -1 if x has fewer than $\lceil k/2 \rceil$ ones, and $f(x)$ if x has exactly $\lceil k/2 \rceil$ ones. The functions we consider will only be defined over the first k out of n variables. Throughout the rest of this section, without loss of generality, we will always assume that k is even.

Theorem 3.1 *Let $\mathcal{P}$ be the class of 2^{k-1} parity functions $\chi\colon \{0,1\}^k \to \{+1, -1\}$ over an odd number of the first k variables. Let $\mathcal{M}$ be the class of corresponding k-variable slice functions slice_χ for $\chi \in \mathcal{P}$. Let $k = \log^{2-\beta}(n)$ for β any absolute constant in $(0,1)$. Then for every $\epsilon = o(1/\sqrt{k})$, we have SQ-SDIM$(\mathcal{M}, \epsilon) = n^{\Theta(\log^{1-\beta} n)}$, and every function in $\mathcal{M}$ is balanced.*

Proof: We first show that every function $\text{slice}_\chi \in \mathcal{M}$ is balanced, i.e. outputs $+1$ and -1 with equal probability. As k is even, the number of inputs with greater than $k/2$ ones is the same as the number of inputs with fewer than $k/2$ ones. As for the middle layer, given an input with exactly $k/2$ ones on which χ outputs $+1$, flipping all the bits gives another point with exactly $k/2$ ones on which χ outputs -1 (as χ is a parity on an odd number of bits). Thus every $\text{slice}_\chi \in \mathcal{M}$ is balanced on the middle layer and thus is balanced overall.

Let $g = \text{slice}_{\mathbf{0}}$. We will show that SQ-DIM$(\mathcal{M} \setminus B_{\mathcal{U}}(\mathtt{sign}(g), 2\epsilon) - g) = n^{\omega(1)}$. By Stirling's approximation, the middle layer of the k-dimensional hypercube is a $\lambda_k = \binom{k}{k/2}/2^k = \Theta(1/\sqrt{k})$ fraction of the 2^k points. Thus for $\epsilon = o(1/\sqrt{k})$ we have that $\mathcal{M}$ is disjoint from $B_{\mathcal{U}}(\mathtt{sign}(g), 2\epsilon) = \emptyset$ (since $\mathtt{sign}(g)$ equals $+1$ everywhere on the middle layer and every function in $\mathcal{M}$ is balanced on the middle layer), and it is enough to lower-bound SQ-DIM$(\mathcal{M} - g)$ in order to lower-bound the strong SQ-dimension of $\mathcal{M}$.

The functions in $\mathcal{M} - g$ have a nice structure as they output 0 everywhere except the middle layer of $\{0,1\}^k$, where they output ± 1. Thus, the correlation between any two functions in $\mathcal{M} - g$ depends only on

the values on the middle slice. Let $\chi_A, \chi_B \in \mathcal{P}$ be the parity functions over the sets of variables $A, B \subseteq [k]$. Recalling Equation (1),

$$|\langle \mathrm{slice}_{\chi_A} - g, \mathrm{slice}_{\chi_B} - g \rangle_{\mathcal{U}}| = |\mathbf{E}_{\mathcal{U}}[\mathbf{1}_{|x|=k/2} \cdot \chi_A \cdot \chi_B]| = \mathbf{E}_{\mathcal{U}}[\mathbf{1}_{|x|=k/2} \cdot \chi_{A \oplus B}] = \widehat{\mathbf{1}_{|x|=k/2}}(A \oplus B)$$

where $A \oplus B$ denotes the symmetric difference between the sets A and B, $\mathbf{1}_{|x|=k/2}$ is the indicator function of the middle slice, and $\widehat{h}(A)$ is the Fourier coefficient of h with index χ_A. In other words, the correlation between $(\mathrm{slice}_{\chi_A} - g)$ and $(\mathrm{slice}_{\chi_B} - g)$ is exactly the Fourier coefficient of $\mathbf{1}_{|x|=k/2}$ with index $A \oplus B$. Let $s = |A \oplus B|$. By symmetry, all $\binom{k}{s}$ of the degree-s Fourier coefficients of $\mathbf{1}_{|x|=k/2}$ are the same, and since by Parseval's identity the squares of all the Fourier coefficients sum to $\mathbf{E}_{\mathcal{U}}[\mathbf{1}^2_{|x|=k/2}] = \lambda_k$, we have $|\langle \mathrm{slice}_{\chi_A} - g, \mathrm{slice}_{\chi_B} - g \rangle_{\mathcal{U}}| \leq \sqrt{\lambda_k / \binom{k}{s}} \leq \binom{k}{s}^{-1/2}$.

It remains to identify a large collection of these slice functions such that the pairwise correlations are small. This can be done easily by picking any $\chi_A \in \mathcal{P}$, removing all $\chi_B \in \mathcal{P}$ such that $|A \oplus B| \notin [k/3, 2k/3]$, and repeating this process. Since each removal step removes at most a $\frac{1}{2^{\Theta(k)}}$ fraction of all 2^{k-1} elements of $\mathcal{P}$, in this fashion we can construct a set S of size $2^{\Theta(k)}$. Every pair of parities in S has symmetric difference s for some $s \in [k/3, 2k/3]$, and for such an s we have $\binom{k}{s} = 2^{\Theta(k)}$. Thus the set $\{\mathrm{slice}_\chi - g\}_{\chi \in S}$ is a collection of $2^{\Theta(k)} = n^{\Theta(\log^{1-\beta} n)}$ functions in $\mathcal{M} - g$ whose pairwise correlations are each at most $1/2^{\Theta(k)} = 1/n^{\Theta(\log^{1-\beta} n)}$, and thus the SQ-dimension of $\mathcal{M} - g$ is at least $n^{\Theta(\log^{1-\beta} n)}$.

□

3.1 The depth-3 construction

It remains to show that every function in $\mathcal{M}$ has a depth-3 monotone formula.

Theorem 3.2 *Let χ be any parity function over some subset of the variables $x_1, \ldots, x_k$ where $k = \log^{2-\beta}(n)$ for β any absolute constant in $(0, 1)$. Then the k-variable slice function slice_χ is computed by an $n^{o(1)}$-size, depth-3 monotone formula.*

Proof: Let Th^k_j be the k-variable threshold function that outputs TRUE if at least j of the k inputs are set to 1, and FALSE otherwise. The threshold function Th^k_j can be computed by a monotone formula of size $n^{o(1)}$ and depth 3 using the construction of Klawe *et al.* [23].

Let χ be a parity function on j out of the first k variables. For $x \in \{0,1\}^k$ let x^1 refer to the j variables of χ and x^2 refer to the remaining $k - j$ variables. We claim that

$$\mathrm{slice}_\chi(x) = \bigvee_{\text{odd } i < j} [\mathrm{Th}^j_i(x^1) \wedge \mathrm{Th}^{k-j}_{k/2-i}(x^2)] \,.$$

To see this, note that if an input x has fewer than $k/2$ ones, then there can be no i such that $\mathrm{Th}^j_i(x^1)$ and $\mathrm{Th}^{k-j}_{k/2-i}(x^2)$ both hold, so this function outputs FALSE as it should. If x has more than $k/2$ ones, some ℓ of them are in x^1, and at least $k/2 - \ell + 1$ of them are in x^2. If ℓ is odd then $i = \ell$ makes the OR output TRUE, and if ℓ is even then $i = (\ell - 1)$ makes the OR output TRUE. Finally, if x has exactly $k/2$ ones, and an odd number of them are in x^1, the formula is satisfied; if an even number of them are in χ, the formula is not satisfied.

Each Th_i^j and $\mathrm{Th}_{k/2-i}^{k-j}$ can be computed by a $n^{o(1)}$-size, depth-3 monotone formula with an OR on top [23]. Using the distributive law we can convert $\mathrm{Th}_i^j(x^1) \wedge \mathrm{Th}_{k/2-i}^{k-j}(x^2)$ to also be a $n^{o(1)}$-size, depth-3 monotone formula with an OR on top. This OR can be collapsed with the top $\lceil j/2 \rceil$-wise OR, yielding a $n^{o(1)}$-size, depth-3 monotone formula for slice_χ. □

We have thus established:

Theorem 3.3 *For some $\epsilon = 1/(\log n)^{\Theta(1)}$, the class of $n^{o(1)}$-size, depth-3 monotone formulas has Strong SQ-Dimension $n^{\omega(1)}$.*

As an immediate corollary, by Theorem 2.4 we get:

Corollary 3.4 *The class of $n^{o(1)}$-size, depth-3 monotone formulas is not SQ-learnable to some accuracy $1 - 1/(\log n)^{\Theta(1)}$ in $\mathrm{poly}(n)$ time.*

Remark 3.5 *We note that Corollary 3.4 can also be obtained via a reduction from the problem of weak SQ learning of parities over the uniform distribution on the middle slice to the problem of the uniform-distribution SQ learning $\mathcal{M}$ to accuracy $o(1/\sqrt{k})$. Using this simple reduction it is possible to obtain the same lower bound by applying the SQ-DIM–based characterization of weak SQ learning [6]. Therefore our application of the more general SQ-SDIM–based method is not crucial in this case but it might be a useful example in the analysis of the SQ learning complexity of other concept classes.*

In the next section we introduce hardness amplification machinery that will enable us to extend the above hardness result to accuracy $\frac{1}{2} + o(1)$ (for depth-4 formulas).

4 Hardness Amplification for Uniform Distribution Learning

In [28] O'Donnell developed a general technique for hardness amplification. His approach, which may be viewed as a generalization of Yao's XOR lemma, gives a bound on the hardness of $g \otimes f = g(f(x_1), \ldots, f(x_k))$ where f is a "mildly" hard function and g is an arbitrary k-bit combining function.

In this section we prove a uniform analogue of O'Donnell's hardness amplification for PAC and SQ learning with respect to the uniform distribution. The high-level idea is based on O'Donnell's proof, which first shows the existence of a weakly approximating circuit for a function f if there exists a circuit for $g \otimes f$ that outperforms the expected bias of g; for this step we use randomness in place of non-uniformity. The other part of O'Donnell's proof uses Impagliazzo's hard-core lemma, which is highly nonuniform; in its place we use "smooth boosting," a technique from computational learning theory which is known to be equivalent to hard-core set constructions [24].

Additional steps are required to apply this amplification in the SQ model. Specifically, we give a new method to simulate the SQ oracle for $g \otimes f$ using the SQ oracle for f that is based on ideas from Feldman's strong SQ characterization [15].

Notation and Terminology. For g a k-variable Boolean function and f an n-variable Boolean function, we write $g \otimes f$ to denote the nk-variable function $g(f(x_1), \ldots, f(x_k))$. For $\mathcal{F}$ a class of n-variable functions and g a fixed k-variable combining function, we write $\mathcal{F}^g$ to denote the class $\{g \otimes f : f \in \mathcal{F}\}$.

Let P_δ^k denote the distribution of random restrictions ρ on k coordinates, in which each coordinate is mapped independently to $\star$ with probability δ, to 0 with probability $(1-\delta)/2$, and to 1 with probability $(1-\delta)/2$. We write h_ρ for the function given by applying restriction ρ to the function h. For a k-variable ± 1-valued function h we write $\mathrm{bias}(h)$ to denote $\max\{\Pr[h = -1], \Pr[h = 1]\}$. The *expected bias of h at δ* is $\mathrm{ExpBias}_\delta(h) = \mathbf{E}_\rho[\mathrm{bias}(h_\rho)]$, where ρ is a random restriction from P_δ^k.

4.1 Hardness Amplification in the PAC Setting

We first prove an average-case version of O'Donnell's Lemma 5 [28]; intuitively, this says that if all of a function's inputs look completely random then it is impossible to guess its value with probability better than its bias [28].

Lemma 4.1 *Given two functions $h : \{0,1\}^k \to \{-1,1\}$ and $p : \{0,1\}^k \to [0,1]$, suppose that*

$$\frac{1}{2^k}\left(\sum_{x:h(x)=1} p(x) + \sum_{x:h(x)=-1} (1-p(x))\right) \geq \text{bias}(h) + \epsilon\,. \tag{2}$$

Then $\mathbf{E}_{(x,y)}[|p(x)-p(y)|] \geq 4\epsilon^2/k$ where (x,y) is a randomly and uniformly chosen edge in the Boolean hypercube $\{0,1\}^k$.

Proof: Let us assume without loss of generality that h is biased towards 1. By the Poincaré inequality over the discrete cube we know that for any function p over $\{0,1\}^k$:

$$\mathbf{Var}[p] = \mathbf{E}[p^2] - \mathbf{E}[p]^2 \leq \frac{k}{4}\mathbf{E}_{(x,y)}[(p(x)-p(y))^2].$$

The range of p is $[0,1]$, so $\mathbf{E}_{(x,y)}[|p(x)-p(y)|] \geq \mathbf{E}_{(x,y)}[(p(x)-p(y))^2] \geq 4\mathbf{Var}[p]/k$. It is now sufficient to prove that $\mathbf{Var}[p] \geq \epsilon^2$.

Let $b := \text{bias}(h) = \Pr[h=1] \geq 1/2$. We can rewrite Equation 2 as

$$\begin{aligned} b+\epsilon &\leq \frac{1}{2^k}\left(\sum_{x:h(x)=1} p(x) + \sum_{x:h(x)=-1} (1-p(x))\right) \\ &= \mathbf{E}[h(x)(p(x)-\mathbf{E}[p(x)])] + \mathbf{E}[p(x)]b + (1-\mathbf{E}[p(x)])(1-b). \end{aligned}$$

As $b \geq 1/2$, $b\mathbf{E}[p] + (1-b)(1-\mathbf{E}[p]) < b$, and thus $\mathbf{E}[h(x)(p(x)-\mathbf{E}[p(x)])] \geq \epsilon$. Because $h(x) \in \{-1,1\}$ we obtain $\mathbf{E}[|p-\mathbf{E}[p]|] \geq \epsilon$. Using the Cauchy-Schwarz inequality, we get $\mathbf{Var}[p] = \mathbf{E}[(p-\mathbf{E}[p])^2] \geq \mathbf{E}[|p-\mathbf{E}[p]|]^2 \geq \epsilon^2$. □

Suppose we are given a circuit C that approximates $g \otimes f$ sufficiently well that it outperforms the expected bias of g. Roughly speaking, the following lemma shows that for any large enough set S, from C we can extract a circuit C' that weakly approximates f over the inputs in S.

Lemma 4.2 *There is a randomized algorithm* `Extract` *with the following property: For any:*

1. *Parameters $0 < \epsilon \leq 1/2$, $0 < \eta < 1$, subset $S \subseteq \{0,1\}^n$ such that $|S| = \eta 2^n$, Boolean function g over $\{0,1\}^k$, and*
2. *Boolean function f such that $\text{bias}(f) \leq 1/2 + \epsilon/(8k)$ and $\text{bias}(f|_S) \leq 1/2 + \epsilon^2/(4k)$,*

given a circuit C over $\{0,1\}^{k\times n}$ s.t.

$$\mathbf{Pr}_{\mathcal{U}^k}[C = g \otimes f] = \mathbf{Pr}_{(x_1,\ldots,x_k)\in\{0,1\}^{k\times n}}[C(x_1,\ldots,x_k) = g(f(x_1),\ldots,f(x_k))] \geq \text{ExpBias}_\eta(g) + \epsilon,$$

the algorithm `Extract` *returns an n-input circuit C' such that with probability at least ϵ^2/k (over the randomness of* `Extract`*) we have $\mathbf{Pr}_{x\in S}[C'(x) = f(x)] \geq 1/2 + \epsilon^2/(2k)$. The algorithm* `Extract` *runs in time $O(nk + |C|)$ and the circuit C' is of size at most $|C|$.*

Proof: The algorithm `Extract` is very simple: it chooses $i \in [k]$ and $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k \in \{0,1\}^n$ randomly and uniformly. With probability $1/2$ `Extract` outputs the circuit C_1 such that $C_1(x_i) = C(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k)$, and with probability $1/2$ it outputs the circuit $-C_1$.

We first assume for simplicity that f is unbiased on $\{0,1\}^n$ and S, and consequently also on $\{0,1\}^n \setminus S$. For a given restriction $\rho \in P_\eta^k$, we say that an input $(x_1, \ldots, x_k)$ *matches* ρ if for all i we have $x_i \in S \Rightarrow \rho(i) = \star$, and $x_i \notin S \Rightarrow \rho(i) = f(x_i)$. Since we assumed that f is unbiased, the probability that x matches ρ is exactly the probability of ρ in P_η^k. Let c_ρ be the probability that C correctly computes $g \otimes f$ conditioned on the event that $(x_1, \ldots, x_k)$ matches ρ. Then, by definition, $\mathbf{E}_\rho[c_\rho - \mathrm{bias}(g_\rho)] = \mathbf{Pr}_{\mathcal{U}^k}[C = g \otimes f] - \mathrm{ExpBias}_\eta(g) \geq \epsilon$. Let adv_ρ denote $\max\{0, c_\rho - \mathrm{bias}(g_\rho)\}$. Clearly, $\mathbf{E}_\rho[\mathrm{adv}_\rho] \geq \epsilon$. For a given ρ let $I_\rho = \{i \mid \rho(i) = \star\}$ and let $k_\rho = |I_\rho|$. For $y \in \{-1,1\}^{k_\rho}$, let $p_\rho(y)$ be the probability that $C(x) = 1$ conditioned on the event that $(x_1, \ldots, x_k)$ matches ρ and $f(x_i) = y_i$ for each i such that $\rho(i) = \star$ (we refer to this condition as x *matching* (ρ, y)). We index the bits of y with subscripts from I_ρ. Note that under our assumption of f being unbiased on S the probability of x matching (ρ, y) is exactly $2^{-k_\rho} P_\eta^k(\rho)$.

Then it follows that:

$$c_\rho = 2^{-k_\rho}\left(\sum_{y: g_\rho(y)=1} p_\rho(y) + \sum_{y: g_\rho(y)=-1} (1 - p_\rho(y))\right).$$

If $\mathrm{adv}_\rho > 0$ then $c_\rho = \mathrm{bias}(g_\rho) + \mathrm{adv}_\rho$, and we can invoke Lemma 4.1 on $g_\rho(y)$ and $p_\rho(y)$. By this lemma, the expected value of $|p_\rho(z) - p_\rho(z^i)|$ for a random edge (z, z^i) in $\{-1,1\}^{k_\rho}$ is at least $4\mathrm{adv}_\rho^2/k_\rho \geq 4\mathrm{adv}_\rho^2/k$. (Here z^i is z with the bit z_i flipped, where i is an element of I_ρ.) Therefore we have

$$\mathbf{E}_{\rho, i \in I_\rho, z \in \{-1,1\}^{k_\rho}}[|p_\rho(z) - p_\rho(z^i)|] \geq \mathbf{E}_{\rho, i \in I_\rho, z \in \{-1,1\}^{k_\rho}}[4\mathrm{adv}_\rho^2/k] = 4\mathbf{E}_\rho[\mathrm{adv}_\rho^2]/k \geq 4\epsilon^2/k\,, \tag{3}$$

where we used $\mathbf{E}_\rho[\mathrm{adv}_\rho] \geq \epsilon$ and the Cauchy-Schwartz inequality to obtain the last inequality. We now observe that $|p_\rho(z) - p_\rho(z^i)|$ is the expected correlation of $C(x)$ and $f(x_i)$ when conditioned on x matching either (ρ, z) or (ρ, z^i). Specifically,

$$\begin{aligned}&\left|\mathbf{E}_{\mathcal{U}^k}[C(x) \cdot f(x_i) \mid x \text{ matches } (\rho, z) \text{ or } (\rho, z^i)]\right| \\ &= \left|\mathbf{E}_{\mathcal{U}^k}[C(x) \mid x \text{ matches } (\rho, z)] - \mathbf{E}_{\mathcal{U}^k}[C(x) \mid x \text{ matches } (\rho, z^i)]\right| = 2 \cdot |p_\rho(z) - p_\rho(z^i)|.\end{aligned}$$

We denote by $x^{(i)} = x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k$ and let z' be equal to z without the i-th bit. With this notation

$$\mathbf{E}_{\mathcal{U}^k}[C(x) \cdot f(x_i) \mid x \text{ matches } (\rho, z) \text{ or } (\rho, z^i)] = \mathbf{E}_{\mathcal{U}^{k-1}}\left[\mathbf{E}_{x_i \in S}[C(x) \cdot f(x_i)] \mid x^{(i)} \text{ matches } (\rho, z')\right]$$

and hence

$$\mathbf{E}_{\mathcal{U}^{k-1}}\left[\left|\mathbf{E}_{x_i \in S}[C(x) \cdot f(x_i)]\right| \mid x^{(i)} \text{ matches } (\rho, z')\right] \geq 2 \cdot |p_\rho(z) - p_\rho(z^i)|.$$

By combining this with equation (3) we obtain:

$$\mathbf{E}_{\rho, i \in I_\rho, z \in \{-1,1\}^{k_\rho}}\left[\mathbf{E}_{\mathcal{U}^{k-1}}\left[\left|\mathbf{E}_{x_i \in S}[C(x) \cdot f(x_i)]\right| \mid x^{(i)} \text{ matches } (\rho, z')\right]\right] \geq 8\epsilon^2/k\,,$$

which is equivalent to

$$\mathbf{E}_{i \in [k], x^{(i)} \sim \mathcal{U}^{k-1}}\left[\left|\mathbf{E}_{x_i \in S}[C(x) \cdot f(x_i)]\right|\right] \geq 8\epsilon^2/k\,.$$

This implies that for randomly chosen i and $x^{(i)}$, with probability at least $4\epsilon^2/k$, we have that $C_1(x_i) = C(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_k)$ satisfies $\left|\mathbf{E}_{x_i \in S}[C_1(x_i) \cdot f(x_i)]\right| \geq 4\epsilon^2/k$. Hence either $\mathbf{Pr}_S[C_1 = f] \geq$

$1/2 + 2\epsilon^2/k$ or $\mathbf{Pr}_S[-C_1 = f] \geq 1/2 + 2\epsilon^2/k$. This means that with probability at least $2\epsilon^2/k$ the output of our algorithm satisfies $\mathbf{Pr}_S[C' = f] \geq 1/2 + 2\epsilon^2/k$. Thus we have proved the lemma under the additional assumption that f is unbiased on $\{0,1\}^n$ and on S.

We now show that a similar but slightly weaker bound can obtained even when f is slightly biased. Let $X' = \{0,1\}^n \cup Z$ for a set Z of size $\epsilon 2^k/(4k)$. Let f' be a function over X' such that $f'|_{\{0,1\}^n} = f$ and $f'|_Z$ is defined in any way that makes f' balanced over X'. This is possible since $\text{bias}(f) \leq 1/2 + \epsilon/(8k)$. Further let S' be $S \cup S_Z$ where $S_Z \subseteq X'$ is a set (disjoint from S) of size at most $|S|\epsilon^2/(2k)$ defined so as to make f' balanced on S'. This is possible since f' is balanced on X' and bias $f'|_S \leq 1/2 + \epsilon^2/(4k)$. Let C_a be the circuit over X'^k that equals to C on $\{0,1\}^{nk}$ and 1 everywhere else. We now apply our argument to C_a and f'.

First,

$$\begin{aligned}\Pr_{\mathcal{U}'^k}[C_a = g \otimes f'] &\geq \Pr_{\mathcal{U}'^k}[x \in \{0,1\}^{nk}] \cdot \Pr_{\mathcal{U}^k}[C = g \otimes f] \geq (1 - \epsilon/(4k))^k(\text{ExpBias}_\eta(g) + \epsilon) \\ &\geq \text{ExpBias}_\eta(g) + 3\epsilon/4 \geq \text{ExpBias}_{\eta'}(g) + 3\epsilon/4\,,\end{aligned}$$

where $\mathcal{U}'$ denotes the uniform distribution over X' and $\eta' = |S'|/|X'| \leq \frac{1+\epsilon^2/(2k)}{1+\epsilon/(4k)}\eta \leq \eta$. Note that here we used the fact that for every function g, $\text{ExpBias}_\eta(g)$ is a monotone function of η [28]. Therefore `Extract` applied to C_a over X'^k gives us a circuit C'_a such that with probability at least $2(3\epsilon/4)^2/k$, we have $\Pr_{S'}[C'_a = f'] \geq 1/2 + 2(3\epsilon/4)^2/k$. There are at most $|S|\epsilon^2/(2k)$ points in $S' \setminus S$ and therefore $\Pr_S[C'_a|_S = f] \geq 1/2 + 2(3\epsilon/4)^2/k - \epsilon^2/(2k) > 1/2 + \epsilon^2/(2k)$. Finally, let $\mathcal{D}'$ denote the the distribution over $C'_a|_S$ that is produced when `Extract` is applied to C_a over X' conditioned on C'_a not hardwiring any x_j to an element of Z. We observe that $\mathcal{D}'$ is exactly the same as the distribution that `Extract` produces when applied to C over $\{0,1\}^n$. However, if $\Pr_S[C'_a|_S = f] > 1/2 + \epsilon^2/(2k)$ then C'_a cannot have any x_j hardwired to an element of Z, since, by definition of C_a, hardwiring x_j to an element of Z turns the circuit into the constant-1 function, whereas we know that $\Pr_S[1 = f] \leq \text{bias}(f|_S) < 1/2 + \epsilon^2/(2k)$. We can therefore conclude that `Extract` applied to C over $\{0,1\}^n$ will output C' that with probability at least $2(3\epsilon/4)^2/k > \epsilon^2/k$ (over the randomness of `Extract`) satisfies $\mathbf{Pr}_S[C' = f] \geq 1/2 + \epsilon^2/(2k)$. □

As we will see below, two key properties of this lemma are that `Extract` is uniform and is oblivious of both f and S. In the second part of the proof, we show how an algorithm A that learns the combined class $\mathcal{F}^g$ to moderate accuracy can be used to obtain an algorithm B that learns the original class $\mathcal{F}$ to high accuracy. This is exactly the well-studied "weak learning $\implies$ strong learning" paradigm of *boosting* in computational learning theory (see [31, 32] for introductions to boosting). Roughly speaking, boosting algorithms are automatic procedures that can be used to convert any weak learning algorithm (that only achieves low accuracy slightly better than $1/2$) into a strong learning algorithm (that achieves high accuracy close to 1). Boosting algorithms work by repeatedly running the weak learning algorithm under a sequence of carefully chosen probability distributions $\mathcal{D}_1, \mathcal{D}_2, \ldots$, obtaining weak hypotheses $h_1, h_2, \ldots$. If each hypothesis h_i has non-negligible accuracy under the distribution $\mathcal{D}_i$ that was used to generate it, then the boosting guarantee ensures that the final hypothesis h (which combines $h_1, h_2, \ldots$) has high accuracy under the original distribution.

Since we require the set $|S|$ to be "large" (recall the statement of Lemma 4.2), we will need to use a so-called "smooth boosting algorithm" such as the algorithm of [33]. A $1/\delta$-*smooth boosting algorithm* is a boosting algorithm with the following property: if the original distribution is uniform over a finite domain X (as is the case for us here), then in learning to final accuracy δ, every distribution $\mathcal{D}_i$ that the smooth boosting algorithm constructs will be "$1/\delta$-smooth," meaning that it puts probability weight at most $\frac{1}{\delta} \cdot \frac{1}{|X|}$

on any example $x \in X$. Such $1/\delta$-smooth distributions correspond naturally to large sets S (of size $\delta 2^n$) in Lemma 4.2.

So at a high level, we use a smooth boosting algorithm, and for each smooth distribution that it constructs we use `Extract` several times to generate a set of candidate weak hypotheses (recall that `Extract` constructs a "good" C' only with some nonnegligible probability). These hypotheses are then tested using uniform examples (filtered according to the current smooth distribution; since the distribution is smooth this does not incur much overhead), and we identify one which has the required nonnegligible accuracy. The boosting guarantee ensures that the combined hypothesis has accuracy $1-\delta$ under the original (uniform) distribution.

Having sketched the intuition for the second stage, we now state the main hardness amplification theorem for PAC learning.

Theorem 4.3 *Let $\mathcal{F}$ be a class of functions such that for every $f \in \mathcal{F}$, $\mathrm{bias}(f) \leq 1/2 + \epsilon/(8k)$. Let A be a uniform distribution PAC learning algorithm that learns $\mathcal{F}^g$ to accuracy $\mathrm{ExpBias}_\delta(g) + \epsilon$. There exists a uniform-distribution PAC learning algorithm B that learns $\mathcal{F}$ to accuracy $1-\delta$ in time $O(T_1 \cdot T_2 \cdot poly(n, k, 1/\epsilon, 1/\delta))$ where T_1 is the time required to evaluate g and T_2 is the running time of A.*

Proof: Let f denote the unknown target function. We will first simulate A to obtain a circuit C that is $(\mathrm{ExpBias}_\delta(g)+\epsilon)$-close to $g \otimes f$. To generate a random example of $g \otimes f$ we simply draw k random examples of f: $(x_1, \ell_1), \ldots, (x_k, \ell_k)$ and give the example $((x_1, \ldots, x_k), g(\ell_1, \ldots, \ell_k))$ to A.

We now use C to produce weak hypotheses on distributions produced by the $1/\delta$-smooth boosting algorithm of Servedio [33] (here δ refers to the desired accuracy parameter).

Let $D_t(x)$ denote the distribution obtained at step t of boosting and let $h_1, \ldots, h_{t-1}$ be the hypotheses obtained in the previous stages of boosting. Let $M = 2^n L_\infty(D_t)$ and let S_t denote a set obtained by including each point $x \in \{0,1\}^n$ randomly with probability $D_t(x)/M$. As it is easy to see (*e.g.,* [18]), for any function h fixed independently of the random choices that determine S_t, with probability at least $1 - 2^{-n/2}$ (over the choice of S_t) $|\Pr_{D_t}[h = f] - \Pr_{S_t}[h = f]| \leq 2^{-n/2}$. Therefore for our purposes we can treat $\Pr_{S_t}[h = f]$ as equal to $\Pr_{D_t}[h = f]$.

If $\mathrm{bias}(f|_{S_t}) \geq 1/2 + \epsilon^2/(4k)$ then $\Pr_{S_t}[f = b] \geq 1/2 + \epsilon^2/(4k)$ for $b \in \{-1, 1\}$. Otherwise, by Lemma 4.2, with probability at least ϵ^2/k the algorithm `Extract` returns a circuit C_1 such that $\mathbf{Pr}_{S_t}[C_1 = f] \geq 1/2 + \epsilon^2/(2k)$. As it is easy to see from the analysis of [33], the value $D_t(x)/M$ equals $\mu_t(f(x), h_1(x), \ldots, h_{t-1}(x))$ for a fixed function μ_t defined by the boosting algorithm. This allows the learning algorithm to generate random examples from $D_t(x)$ by filtering random and uniform examples using μ_t. In particular, we can estimate $\mathbf{Pr}_{D_t}[h = f]$ to accuracy $\epsilon^2/(12k)$ and confidence $1/2$ using $\tilde{O}(k^2/\epsilon^4\delta)$ random and uniform examples in order to test whether either $-1, 1$ or C' give a good weak hypothesis (the $1/\delta$ factor in the number of examples suffices because we are using a $1/\delta$-smooth boosting algorithm). By repeating the execution of `Extract` a total of $O(\epsilon^{-2} \cdot k \log(k/\epsilon\delta))$ times we can ensure that with probability at least $2/3$ this weak learning step is successful in all $O(k^2/(\epsilon^4\delta))$ boosting stages that the booster of [33] requires. This implies that the boosting algorithm produces a $(1-\delta)$-accurate hypothesis with probability at least $2/3$. It is easy to verify that the running time of this algorithm is as claimed. □

Remark 4.4 *This hardness amplification also applies to algorithms using membership queries since membership queries to $g \otimes f$ can be easily simulated using membership queries to f.*

4.2 Hardness amplification in the Statistical Query setting

We now establish the SQ version of this result.

Theorem 4.5 *Let $\mathcal{F}$ be a class of functions such that for every $f \in \mathcal{F}$, $\mathrm{bias}(f) \leq 1/2 + \epsilon/(8k)$. Let A be a uniform-distribution SQ-learning algorithm that learns $\mathcal{F}^g$ to accuracy $\mathrm{ExpBias}_\delta(g) + \epsilon$ using queries of tolerance τ. There exists a uniform-distribution SQ learning algorithm B that learns $\mathcal{F}$ to accuracy $1 - \delta$ in time $O(T_1 \cdot T_2 \cdot poly(n, k, 1/\epsilon, 1/\delta))$ using queries of tolerance $\Omega(\delta \cdot \min\{\tau/k, \epsilon^2/k\})$, where T_1 is the time required to evaluate g and T_2 is the running time of A.*

Proof: The main challenge in translating the result to SQ-learning is to simulate SQs for $g \otimes f$ using SQs for f. Given the circuit C we can proceed exactly as in the proof of Theorem 4.3 but use SQs of tolerance $\Omega(\delta\epsilon^2/k)$ to estimate the bias of f on S_t or to test whether the output of `Extract` is a weak hypothesis.

We now describe how to simulate statistical queries to $g \otimes f$. The distribution is known to be uniform therefore it is sufficient to answer correlational statistical queries of A [9], namely, it is sufficient to be able to estimate $\mathbf{E}_{\mathcal{U}^k}[\phi \cdot (g \otimes f)]$ within $\tau/2$, where ϕ is a Boolean function over $\{0,1\}^{kn}$. To estimate $\mathbf{E}_{\mathcal{U}^k}[\phi \cdot (g \otimes f)]$ we plan to use random sampling with an approximation to f used in place of f. We refer to the approximation as $\psi_r(x)$. However before doing so, we first test whether ψ_r is suitable to be used as a replacement. In the main technical claim we prove that if $\psi_r(x)$ cannot be used to replace f then we can find a way to update ψ_r to ψ_{r+1} which is closer to f than ψ_r in L_2 distance. The number of such updates will be bounded and therefore we will eventually obtain ψ_r that can be used in place of f. Formally, let $\psi_0(x) \equiv 0$ and for a function $\psi_r(x) \in \mathcal{F}_1^\infty$ we denote by $\Psi_r(x)$ the random $\{-1,1\}$ variable with expectation $\psi_r(x)$. We also denote by $g \otimes \Psi_r$ the random variable obtained by applying g to k evaluations of Ψ_r.

Lemma 4.6 *For $i \in [k]$ and $y = y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k \in \{0,1\}^n$ and any function ϕ over $\{0,1\}^{kn}$, we denote $\phi_{i,y}(x_i) = \phi(y_1, y_2, y_{i-1}, x_i, y_{i+1}, \ldots, y_k)$. Let $\lambda = |\mathbf{E}_{\mathcal{U}^k}[\phi \cdot g \otimes f] - \mathbf{E}_{\mathcal{U}^k,\Psi}[\phi \cdot g \otimes \Psi]|$. Then for randomly and uniformly chosen i, y, with probability at least $\lambda/(4k)$, $|\mathbf{E}_{\mathcal{U}}[\phi_{i,y}(x) \cdot f(x)] - \mathbf{E}_{\mathcal{U}}[\phi_{i,y}(x) \cdot \psi(x)]| \geq \lambda/(2k)$.*

Proof: First we denote by $g \otimes f^{i,\Psi}$ the i-th hybrid between $g \otimes f$ and $g \otimes \Psi$. Namely, the randomized function $g \otimes f^{i,\Psi}(x) = g(f(x_1), \ldots, f(x_i), \Psi(x_{i+1}), \ldots, \Psi(x_k))$. Now, $g \otimes f^{k,\Psi} = g \otimes f$ and $g \otimes f^{0,\Psi} = g \otimes \Psi$. Hence we can write,

$$\left|\mathbf{E}_{\mathcal{U}^k}[\phi \cdot g \otimes f] - \mathbf{E}_{\mathcal{U}^k,\Psi}[\phi \cdot g \otimes \Psi]\right| = k \cdot \left|\mathbf{E}_{i \in [k]}\left[\mathbf{E}_{\mathcal{U}^k,\Psi}[\phi \cdot g \otimes f^{i,\Psi}] - \mathbf{E}_{\mathcal{U}^k,\Psi}[\phi \cdot g \otimes f^{i-1,\Psi}]\right]\right|$$

We now split the random and uniform choice over $\{0,1\}^{kn}$ into choosing $y = y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_k \in \{0,1\}^n$ and $x_i \in \{0,1\}^n$ randomly and uniformly.

$$\left|\mathbf{E}_{i \in [k], y}\left[\mathbf{E}_{\mathcal{U},\Psi}[(\phi \cdot g \otimes f^{i,\Psi})_{i,y}(x_i)] - \mathbf{E}_{\mathcal{U},\Psi}[(\phi \cdot g \otimes f^{i-1,\Psi})_{i,y}(x_i)]\right]\right| \geq \lambda/k \,. \tag{4}$$

We claim that

$$\mathbf{E}_{\mathcal{U},\Psi}[(\phi \cdot g \otimes f^{i,\Psi})_{i,y}(x_i)] = \mathbf{E}_{\mathcal{U}}\left[\phi_{i,y} \cdot \mathbf{E}_{\Psi}[(g \otimes f^{i,\Psi})_{i,y}(x_i)]\right] = \mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot (\alpha_{i,y} f(x_i) + \beta_{i,y})] \,.$$

Here $\alpha_{i,y}$ and $\beta_{i,y}$ are constants in $[-1,1]$.

To see this assume for simplicity that Ψ is deterministic. Then

$$(g \otimes f^{i,\Psi})_{i,y}(x_i) = g(f(y_1), \ldots, f(y_{k-1}), f(x_i), f(y_{k-1}), \ldots, f(y_k)) \,.$$

All the variables of g are fixed except for the i-th and therefore this restriction of g equals $1, -1, f(x_i)$ or $-f(x_i)$. This corresponds to $\alpha_{i,y}, \beta_{i,y} \in \{-1, 0, 1\}$ and exactly one of them is non-zero. For randomized Ψ we obtain a fixed convex combination of the deterministic cases that can be represented by $\alpha_{i,y}, \beta_{i,y} \in [-1,1]$. Similarly,

$$\mathbf{E}_{\mathcal{U},\Psi}[(\phi \cdot g \otimes f^{i-1,\Psi})_{i,y}(x_i)] = \mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot (\alpha_{i,y} \psi(x_i) + \beta_{i,y})] \,.$$

By substituting this into equation (4), we obtain

$$\left|\alpha_{i,y} \cdot \mathbf{E}_{i\in[k],y}\left[\mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot f(x_i)] - \mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot \psi(x_i)]\right]\right| \geq \lambda/k\,.$$

By the averaging argument, we obtain that with probability at least $\lambda/(4k)$ over the choice of i and y, $|\mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot f(x_i)] - \mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot \psi(x_i)]| \geq \lambda/(2k)$. □

If $|\mathbf{E}_{\mathcal{U}^k}[\phi \cdot g \otimes f] - \mathbf{E}_{\mathcal{U}^k,\Psi_r}[\phi \cdot g \otimes \Psi_r]| \geq \tau/3$ then with probability at least $\tau/(12k)$ for a randomly chosen $\phi_{i,y}$, $|\mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot f(x_i)] - \mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot \psi(x_i)]| \geq \tau/(6k)$. Let $\tau' = \tau/(6k)$. Now we sample $\phi_{i,y}$ and test if $|\mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot f(x_i)] - \mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot \psi(x_i)]| \leq 2\tau'/3$ using a single SQ of tolerance $\tau'/6$ and an estimate of $\mathbf{E}_{\mathcal{U}}[\phi_{i,y} \cdot \psi(x_i)]$ within $\tau'/6$ obtained using random sampling. It is easy to see that by repeating this procedure $O(k \log(1/\Delta)/\tau)$ times and using $O(k \log(1/\Delta)/\tau)$ random samples we can ensure that with probability at least $1 - \Delta$ some $\phi_{i',y'}$ will pass the test whenever $|\mathbf{E}_{\mathcal{U}^k}[\phi \cdot g \otimes f] - \mathbf{E}_{\mathcal{U}^k,\Psi_r}[\phi \cdot g \otimes \Psi_r]| \geq \tau/3$ and also that $|\mathbf{E}_{\mathcal{U}}[\phi_{i',y'} \cdot f(x_{i'})] - \mathbf{E}_{\mathcal{U}}[\phi_{i',y'} \cdot \psi(x_{i'})]| \geq 2\tau'/3 - \tau'/3 = \tau'/3$ whenever $\phi_{i',y'}$ passes the test.

If the test was not passed then we estimate $\mathbf{E}_{\mathcal{U}^k,\Psi}[\phi \cdot g \otimes \Psi]$ within $\tau/6$ using random sampling and return the estimate as the answer to the query. Using $O(k \log(1/\Delta)/\tau)$ random samples we can ensure that with probability $1 - 2\Delta$ the returned estimate is $\tau/2$ close to $\mathbf{E}_{\mathcal{U}^k}[\phi \cdot g \otimes f]$.

Otherwise, we use such $\phi_{i',y'}$ to update ψ_r as in the SQ characterization [15, Th.3.5]: $\psi_{r+1} = P_1(\psi_r + (\tau'/3) \cdot \phi_{i',y'})$. Here $P_1(a)$ is the function that equals a when $a \in [-1, 1]$ and equals $\mathtt{sign}(a)$ otherwise. As is proved in [15, Th.3.5], $|\mathbf{E}_{\mathcal{U}}[\phi_{i',y'} \cdot (f(x_{i'}) - \psi(x_{i'})]| \geq \tau'/3$ implies that $\mathbf{E}_{\mathcal{U}}[(f - \psi_{r+1})^2] \leq \mathbf{E}_{\mathcal{U}}[(f - \psi_r)^2] - (\tau'/3)^2$. Therefore at most $O(k^2/\tau^2)$ such updates are possible giving an upper bound on the additional time required to produce the desired estimates to all the SQs of A. For an appropriate $\Delta = \mathrm{poly}(k, 1/\tau)$ we can make sure that the success probability is at least $2/3$. □

5 Amplified Hardness for SQ Learning of Depth-4 Monotone Formulas

We begin this section by showing how a refinement of the constructions and analysis from [28, 26] can be used to obtain a small monotone CNF with low expected bias. Specifically, we prove the following lemma.

Lemma 5.1 *For every $0 < \gamma < 1/2$, there exists a circuit $C_{k,m}$ over k variables such that:*

$$\mathrm{ExpBias}_{1/\sqrt{m}}(C_{k,m}) \leq \frac{1}{2} + 2^{-(\log n)^{\gamma}},$$

where $k = 2^{(\log n)^{\alpha}}$ and $m = \log^{2-\beta}(n)$ for $\gamma < \alpha < \beta/2 < 1/2$, and $C_{k,m}$ is computable by a monotone CNF of size $n^{o(1)}$.

In Section 5.1 we use Lemma 5.1 together with our hardness amplification tools to give a strong hardness result for SQ learning depth-4 monotone formulas.

Before we prove Lemma 5.1 we review some of the constructions and tools from [28, 26].

As described in [28], instead of calculating the expected bias of a function directly, one can use *noise stability* to estimate the expected bias. For $x \in \{0,1\}^n$, let $N_\delta(x)$ be a random variable in $\{0,1\}^n$ obtained by flipping each bit of x independently with probability δ.

Definition 5.2 *The* noise stability *of f at noise rate δ is:*

$$\mathrm{NoiseStab}_\delta(f) = \mathbf{E}[f(x) \cdot f(N_\delta(x))],$$

where the probability is taken over the noise and a uniform choice of x.

The following two facts about the noise stability will be useful in proving Lemma 5.1.

Fact 5.3 *If f is a balanced function and g is any function, then*

$$\mathrm{NoiseStab}_\delta(g \otimes f) = \mathrm{NoiseStab}_{\frac{1-\mathrm{NoiseStab}_\delta(f)}{2}}(g).$$

The noise stability of a function provides a bound on its expected bias.

Lemma 5.4 ([28]) $\mathrm{ExpBias}_{2\delta}(f) \leq \frac{1}{2} + \frac{1}{2}\sqrt{\mathrm{NoiseStab}_\delta(f)}$.

The first class of functions we consider are the Talagrand functions. Talagrand's function is a randomly constructed CNF formula on n inputs [38]. Specifically, it has $2^{\sqrt{n}}$ clauses of size $\sqrt{n}$, where the $\sqrt{n}$ variables of each clause are selected independently and uniformly at random from $[n]$. We write $\mathcal{T}_n$ to denote this probability distribution over n-variable CNFs. Mossel and O'Donnell showed that these functions are somewhat sensitive to very small amounts of noise. They prove:

Lemma 5.5 ([26]) *For infinitely many values of n, there is a function T_n in the support of $\mathcal{T}_n$ such that* $\mathrm{NoiseStab}_{1/\sqrt{n}}(T_n) \leq 1 - \Omega(1)$.

The functions of Lemma 5.5 are not necessarily balanced. We show that there is a balanced version of T_n that can be computed by a slightly larger CNF formula while preserving its noise stability.

Lemma 5.6 *Let f be a monotone function over n variables represented by a size-s CNF, and let $0 < \eta < 1/2$ be such that $\Pr[f(x) \neq f(N_\eta(x))] \geq \delta$. Then there exists a* balanced *monotone function g over $n+2$ variables represented by a size-$(s+n)$ CNF formula such that $\Pr[g(x) \neq g(N_\eta(x))] \geq \delta/16$.*

Proof: Our proof is a simple refinement of the proof from [26] that shows the existence of a balanced version of T_n (not necessarily computable by a "small" CNF).

The new function g will be defined on x and on two extra variables z_1 and z_2. Let $m_a(x)$ be the monotone function that checks if the number represented by x in binary is at least the number represented by a. Let $g_a(zx) = (f(x) \vee z_1) \wedge (m_a(x) \vee z_2)$. Note that $g_a(01x) = f(x)$, $g_a(10x) = m_a(x)$, and $g(11x) = \mathrm{TRUE}$. The function g_{0^n} is biased towards TRUE and g_{1^n} is biased towards FALSE and for consecutive a and a', g_a differs from $g_{a'}$ on exactly one point. Therefore there exists a value of a such that g_a is perfectly balanced. The function $m_a(x)$ can be represented as a short CNF as follows $\wedge_{i:a_i=1}(x_i \vee_{j:j<i,a_j=0} x_j)$. Thus $g_a(x)$ has at most n more clauses than f.

Finally, we have

$$\Pr[g(zx) \neq g(N_\eta(zx))] \geq \Pr[z_1 = 0 \wedge z_2 = 1 \wedge N_\eta(z) = 01] \cdot \Pr[f(x) \neq f(N_\eta(x))] \geq \delta/16,$$

and thus the noise stability of the function is preserved. □

Applying Lemma 5.6 to the appropriate Talagrand function chosen from T_{n-2} gives us the following corollary.

Corollary 5.7 *There exists an infinite family of balanced monotone functions $\mathrm{Tal}_n : \{0,1\}^n \to \{+1,-1\}$ with $\mathrm{NoiseStab}_{n^{-1/2}}(\mathrm{Tal}_n) \leq 1 - \Omega(1)$, that can be represented by a $(2^{\sqrt{n}} + n)$-size monotone CNF.*

The second function we consider is the "tribes function", which is sensitive to moderate noise. Given a positive integer b, let $n = n_b$ be the smallest positive integral multiple of b such that $(1 - 2^{-b})^{n/b} \leq 1/2$, so n is roughly $(\ln 2)b2^b$ and b equals $\log n - \log\ln n + o(1)$. The "tribes" function on n variables is the the read-once n/b-term monotone b-DNF

$$(x_1 \wedge \cdots \wedge x_b) \vee (x_{b+1} \wedge \cdots \wedge x_{2b}) \vee \cdots \vee (x_{n-b+1} \wedge \cdots \wedge x_n).$$

Lemma 5.8 ([26, 28]) *Let* $\mathrm{Tribes}_n : \{0,1\}^n \to \{+1, -1\}$ *denote the n-variable tribes function. For every constant* $0 < \delta < 1$*, we have*

$$\mathrm{NoiseStab}_\delta[\mathrm{Tribes}_n] = O(n^{-c_\delta})$$

where $c_\delta > 0$ *is a constant depending only on* δ*.*

We will use the function $\mathrm{Tribes}^\dagger_n$ which is the monotone complement of Tribes (*i.e.,* $\neg\,\mathrm{Tribes}(\neg x)$) which is given by a n/b-clause b-CNF and has the same noise stability as Tribes_n by Boolean duality.

Now we will combine the tribes function with the balanced Talagrand function to obtain a function that is sensitive to very small amounts of noise, using the following lemma to bound its noise stability.

Lemma 5.9 *For infinitely many values of* k, m *with* $m < k$*, the function* $C_{k,m} := \mathrm{Tribes}^\dagger_{k/m} \otimes \mathrm{Tal}_m$ *on* k *variables has:*

$$\mathrm{NoiseStab}_{1/\sqrt{m}}(C_{k,m}) \leq O\left(\left(\frac{m}{k}\right)^c\right)$$

where $c > 0$ *is some absolute constant.*

Proof: By Corollary 5.7 we know that Tal_m is balanced so we can use Fact 5.3 to combine the noise stabilities of Tal_m and $\mathrm{Tribes}^\dagger_{k/m}$ (Lemma 5.8). □

The function $C_{k,m}$ can in fact be computed by a small monotone CNF formula.

Lemma 5.10 *Let* $C_{k,m} := \mathrm{Tribes}^\dagger_{k/m} \otimes \mathrm{Tal}_m$*. The function* $C_{k,m}$ *can be computed by a monotone CNF formula of size* $\frac{k}{mb} \cdot (2^{\sqrt{m}} + m)^b)$*, where* $b = \log(k/m) - \log\ln(k/m) + o(1)$*.*

Proof: $C_{k,m}$ can be represented as a depth-4 monotone formula by combining the $k/(mb)$-clause monotone b-CNF formula for the $\mathrm{Tribes}^\dagger_{k/m}$ function with the $(2^{\sqrt{m}} + m)$-clause monotone CNF for the Tal_m function. Using the distributive rule we can rewrite the b-wise ORs at the bottom of the $\mathrm{Tribes}^\dagger_{k/m}$ function with the $(2^{\sqrt{m}} + m)$-wise ANDs of the Tal_m functions to get $(2^{\sqrt{m}} + m)^b$-wise ANDs followed by b-wise ORs. Now the two adjacent levels of ANDs and the two adjacent levels of ORs can be collapsed to obtain a size-$(\frac{k}{mb} \cdot (2^{\sqrt{m}} + m)^b)$, monotone CNF formula. □

We are now ready to prove Lemma 5.1.

Proof: Let $C_{k,m}$ be the function from Lemma 5.9. Combining Lemmas 5.9 and 5.4, we can bound the expected bias by

$$\frac{1}{2} + O\left(\left(\frac{m}{k}\right)^c\right) = \frac{1}{2} + O\left(\log^{c(2-\beta)+\alpha}(n) \cdot 2^{-c(\log n)^\alpha}\right) = \frac{1}{2} + O(2^{-(\log n)^\gamma}),$$

recalling that $\gamma < \alpha < \beta/2 < 1/2$.

By Lemma 5.10, we know that $C_{k,m}$ can be computed by a monotone CNF formula of size $\frac{k}{mb} \cdot (2^{\sqrt{m}} + m)^b$, where $b = \log(k/m) - \log\ln(k/m) + o(1)$. Therefore, for our setting of k and m, the CNF formula will have size at most $2^{(\log n)^{1-\beta/2+\alpha} + (\log n)^\alpha} = n^{o(1)}$. □

5.1 Proof of Hardness of SQ Learning Depth-4 Monotone Formulas

Coupled with our hardness result for depth-3 monotone formulas, Lemma 5.1 gives the claimed lower bound for depth-4 monotone formulas.

Theorem 5.11 *For every $0 < \gamma < 1/2$, the class of $n^{o(1)}$-size, depth-4 monotone formulas is not SQ-learnable to accuracy $\frac{1}{2} + 2^{-(\log n)^{\gamma}}$ in* poly(n) *time.*

Proof: Let $\mathcal{M}$ be the class from Theorem 3.1 that is defined over $\log^{2-\beta}(n)$ variables, let $C_{k,m}$ be the function from Lemma 5.9 defined over $k = 2^{(\log n)^{\alpha}}$ for $m = \log^{2-\beta}(n)$, and let $\gamma < \alpha < \beta/2 < 1/2$, $c < \beta/2$.

Let $\delta = 1/\log(n)^{1-c}$. As $\delta = o(1/\sqrt{m}) = o(1/\log^{1-\beta/2}(n))$, by Theorem 2.4 and Theorem 3.1, there is no poly(n)-time algorithm that can SQ-learn the class $\mathcal{M}$ (of m-variable functions) to accuracy $1 - \delta$. Every function in $\mathcal{M}$ has bias $1/2$. Thus, by Theorem 4.5 and Lemma 5.1, there is no poly(n)-time algorithm that can SQ-learn the class $\mathcal{M}^{C_{k,m}}$ to accuracy

$$\mathrm{ExpBias}_{\delta}(C_{k,m}) \leq \frac{1}{2} + O(2^{-(\log n)^{\gamma}}).$$

Combining the size-$n^{o(1)}$, depth-2 monotone formulas constructed in Lemma 5.1 with ORs on the bottom layer with k copies of the size-$n^{o(1)}$, depth-3 monotone formulas constructed in Theorem 3.2 with ORs on the top layer gives us a $n^{o(1)}$-size, depth-4 monotone formula as required. □

References

[1] K. Amano and A. Maruoka. On learning monotone boolean functions under the uniform distribution. In *Proceedings of the 13th International Conference on Algorithmic Learning Theory (ALT)*, pages 57–68, 2002.

[2] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *FOCS*, pages 238–247, 2002.

[3] Avrim Blum, Carl Burch, and John Langford. On learning monotone boolean functions. In *Proc. 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 408–415. IEEE Computer Society Press, 1998.

[4] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *PODS*, pages 128–138, 2005.

[5] Avrim Blum, Alan M. Frieze, Ravi Kannan, and Santosh Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1/2):35–52, 1998.

[6] Avrim Blum, Merrick L. Furst, Jeffrey Jackson, Michael J. Kearns, Yishay Mansour, and Steven Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 253–262. ACM Press, 1994.

[7] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, July 2003. Prelim. ver. in *Proc. of STOC'00*.

[8] D. Boneh and R. Lipton. Amplification of weak learning over the uniform distribution. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 347–351, 1993.

[9] Nader H. Bshouty and Vitaly Feldman. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research*, 2:359–395, 2002. Prelim. ver. in *Proc. of COLT'01*.

[10] Nader H. Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 2006.

[11] T. Bylander. Learning noisy linear threshold functions. Available at: `http://ringer.cs.utsa.edu/research/AI/bylander/pubs/pubs.html`, 1998.

[12] Tom Bylander. Learning linear threshold functions in the presence of classification noise. In *Proc. of the 7th Annual Conference on Computational Learning Theory (COLT)*, pages 340–347, 1994.

[13] D. Dachman-Soled, H. Lee, T. Malkin, R. Servedio, A. Wan, and H. Wee. Optimal cryptographic hardness of learning monotone functions. In *Proc. 35th International Colloquium on Algorithms, Languages and Programming (ICALP)*, pages 36–47, 2008.

[14] John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. In *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 315–320. ACM Press, 2004.

[15] V. Feldman. A complete characterization of statistical query learning with applications to evolvability. *CoRR*, abs/1002.3183, 2010. Prelim. ver. in *Proc. of FOCS'09*.

[16] Vitaly Feldman. Evolvability from learning algorithms. In *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 2008.

[17] T. Hancock and Y. Mansour. Learning monotone k-μ DNF formulas on product distributions. In *Proceedings of the Fourth Annual Conference on Computational Learning Theory*, pages 179–193, 1991.

[18] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the Thirty-Sixth Annual Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[19] Jeffrey C. Jackson, Homin K. Lee, Rocco A. Servedio, and Andrew Wan. Learning random monotone DNF. In *11th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems and 12th International Workshop on Randomization and Computation (RANDOM-APPROX)*, pages 483–497. Springer-Verlag, 2008.

[20] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540, 2008.

[21] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998. Prelim. ver. in *Proc. of STOC'93*.

[22] Michael J. Kearns, Ming Li, and Leslie G. Valiant. Learning Boolean formulas. *Journal of the ACM*, 41(6):1298–1328, 1994. Prelim. ver. in *Proc. of STOC'87*.

[23] Maria M. Klawe, Wolfgang J. Paul, Nicholas Pippenger, and Mihalis Yannakakis. On monotone formulae with restricted depth. In *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*, pages 480–487. ACM Press, 1984.

[24] Adam Klivans and Rocco A. Servedio. Boosting and hard-core sets. *Machine Learning*, 53(3):217–238, 2003. Prelim. ver. in *Proc. of FOCS'99*.

[25] Adam R. Klivans and Alexander A. Sherstov. Unconditional lower bounds for learning intersections of halfspaces. In *Proc. of the 19th Annual Conference on Computational Learning Theory (COLT)*, 2006.

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

[27] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. *SIAM J. Comput.*, 37(3):827–844, 2007.

[28] Ryan O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, 2004. Prelim. ver. in *Proc. of STOC'02*.

[29] Ryan O'Donnell and Karl Wimmer. KKL, Kruskal-Katona, and monotone nets. In *Proc. 50th IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, 2009.

[30] Y. Sakai and A. Maruoka. Learning monotone log-term DNF formulas under the uniform distribution. *Theory of Computing Systems*, 33:17–33, 2000.

[31] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. Prelim. ver. in *Proc. of FOCS'1989*.

[32] Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2001.

[33] R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

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

[35] Alexander A. Sherstov. Communication complexity under product and nonproduct distributions. *Computational Complexity, Annual IEEE Conference on*, 0:64–70, 2008.

[36] Hans Ulrich Simon. A characterization of strong learnability in the statistical query model. In *Proc. 24th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 393–404, 2007.

[37] B. Szörényi. Characterizing statistical query learning:simplified notions and proofs. In *ALT*, pages 186–200, 2009.

[38] M. Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.

[39] Ke Yang. On learning correlated Boolean functions using statistical query. Technical Report 98, Electronic Colloquium on Computational Complexity (ECCC), 2001. Prelim. ver. in *Proc. of ALT'01*.

[40] Ke Yang. New lower bounds for statistical query learning. *Journal of Computer and System Sciences*, 70(4):485–509, 2005. Prelim. ver. in *Proc. of COLT'02*.

[41] Ke Yang and Avrim Blum. On statistical query sampling and nmr quantum computing. In *IEEE Conference on Computational Complexity*, pages 194–, 2003.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de