Electronic Colloquium on Computational Complexity, Report No. 57 (2006)

Cryptographic Hardness Results for Learning Intersections of Halfspaces

Adam R. Klivans and Alexander A. Sherstov

The University of Texas at Austin
Department of Computer Sciences
Austin, TX 78712 USA
{klivans,sherstov}@cs.utexas.edu

Abstract

We give the first representation-independent hardness results for PAC learning intersections of halfspaces, a central concept class in computational learning theory. Our hardness results are derived from two public-key cryptosystems due to Regev, which are based on the worst-case hardness of well-studied lattice problems. Specifically, we prove that a polynomial-time algorithm for PAC learning intersections of n^ϵ halfspaces (for a constant $\epsilon > 0$) in n dimensions would yield a polynomial-time solution to $\tilde{O}(n^{1.5})$-uSVP (unique shortest vector problem). We also prove that PAC learning intersections of n^ϵ low-weight halfspaces would yield a polynomial-time quantum solution to $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP (shortest vector problem and shortest independent vector problem, respectively). By making stronger assumptions about the hardness of uSVP, SVP, and SIVP, we show that there is no polynomial-time algorithm for learning intersections of $\log^c n$ halfspaces in n dimensions, for $c > 0$ sufficiently large. Our approach also yields the first representation-independent hardness results for learning polynomial-size depth-2 neural networks and polynomial-size depth-3 arithmetic circuits.

ISSN 1433-8092

1 Introduction

A halfspace in n dimensions is a Boolean function of the form $a_1x_1 + a_2x_2 + \cdots + a_nx_n \geqslant \theta$, where $a_1, \ldots, a_n, \theta$ are integers. Halfspace-based learning methods have important applications in almost every area of computer science, including data mining, artificial intelligence, and computer vision. A natural and important extension of the concept class of halfspaces is the concept class of *intersections* of halfspaces. While many efficient algorithms exist for PAC learning a *single* halfspace, the problem of learning the intersection of even two halfspaces remains a central challenge in computational learning theory, and a variety of efficient algorithms have been developed for natural restrictions of the problem [14, 15, 18, 25]. Attempts to prove that the problem is hard have been met with limited success: all known hardness results for the general problem of PAC learning intersections of halfspaces apply only to the case of *proper* learning, where the output hypothesis must be of the same form as the unknown concept.

1.1 Our Results

We obtain the first *representation-independent* hardness results for PAC learning intersections of halfspaces. Assuming the intractability of the lattice problems uSVP (unique shortest vector problem), SVP (shortest vector problem), or SIVP (shortest independent vector problem), we prove that there is no polynomial-time PAC learning algorithm for intersections of n^ϵ halfspaces (for any $\epsilon > 0$). The above lattice problems are widely believed to be hard [21].

Our hardness results apply even to intersections of *light* halfspaces, i.e., halfspaces whose weight $|\theta| + \sum_{i=1}^{n} |a_i|$ is bounded by a polynomial in n (we say a halfspace is *heavy* otherwise). We first state our hardness results for intersections of *heavy* halfspaces. Throughout this paper, "PAC learnable" stands for "learnable in the PAC model in polynomial time."

Theorem 1.1. *Assume that intersections of n^ϵ heavy halfspaces in n dimensions are PAC-learnable for some constant $\epsilon > 0$. Then there is a polynomial-time solution to $\tilde{O}(n^{1.5})$-uSVP.*

With a different (incomparable) hardness assumption, we obtain an intractability result for learning intersections of *light* halfspaces, a less powerful concept class:

Theorem 1.2. *Assume that intersections of n^ϵ light halfspaces in n dimensions are PAC-learnable for some constant $\epsilon > 0$. Then there is a polynomial-time quantum solution to $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP.*

These hardness results extend to polynomial-size depth-2 neural networks as follows:

Theorem 1.3. *Assume that depth-2 polynomial-size circuits of majority gates are PAC learnable. Then there is a polynomial-time solution to $\tilde{O}(n^{1.5})$-uSVP and polynomial-time quantum solutions to $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP.*

Finally, we prove a hardness result for learning depth-3 arithmetic circuits:

Theorem 1.4. *Assume that depth-3 polynomial-size arithmetic circuits are PAC-learnable in polynomial time. Then there is a polynomial-time quantum solution to $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP.*

We are not aware of any previous representation-independent hardness results for learning small-depth arithmetic circuits.

By making a stronger quantitative assumption about the hardness of uSVP and the quantum hardness of SVP and SIVP, we can sharpen Theorems 1.1–1.4. Specifically, we show that there is no polynomial-time algorithm for PAC learning the intersection of $\log^c n$ halfspaces in n dimensions, depth-2 majority circuits with $\log^c n$ gates, or depth-3 arithmetic circuits with $\log^c n$ gates (here $c > 0$ is a sufficiently large absolute constant). See Theorems 6.3–6.5 for precise statements. We note here that the algorithm due to Klivans *et al.* [14] PAC learns the intersection of k weight-w halfspaces in time $n^{O(k \log k \log w)}$. Taking $k = \log^c n$ and $w = n^{O(1)}$ we see that there is a *quasi*-polynomial time algorithm for PAC learning the intersection of polylogarithmically many light halfspaces.

A natural question to ask is whether our approach can yield hardness results for other classes such as AC^0 or, more ambitiously, polynomial-size DNF formulas. In Section 7 we show that the decryption functions of the cryptosystems we use contain parity as a subfunction, so we cannot directly apply this approach.

Note: Subhash Khot has recently informed us that he has independently obtained Theorem 1.3.

1.2 Previous Results

In his fundamental paper on learning, Valiant [24] established a cryptographic hardness result for learning polynomial-size circuits. Kearns and Valiant [10] used number-theoretic problems (inverting the RSA function, deciding quadratic residuosity, and factoring Blum integers) to obtain hardness results for NC^1 circuits, constant-depth threshold circuits TC_0, and deterministic finite automata. Kharitonov [11] obtained hardness results for AC^1 and NC^1 circuits based on the conjectured hardness of the subset sum problem. Kharitonov [12] later used the Blum-Blum-Shub pseudorandom generator [5] to obtain a hardness result for learning AC^0 and TC_0 that holds even under the uniform distribution and if membership queries are allowed.

Hardness results of any kind for learning intersections of halfspaces, by contrast, have seen quite limited progress. Until recently, the problem was known to be hard only for *proper* learning: if the learner's output hypothesis must be from a restricted class of functions (e.g., intersections of halfspaces), then the learning problem is NP-hard with respect to randomized reductions [4, 2]. Klivans and Sherstov [17] have since obtained a $2^{\Omega(\sqrt{n})}$ lower bound on the sample complexity of learning intersections of $\sqrt{n}$ halfspaces in the *statistical query* (SQ) model, an important restriction of the PAC model. Since the SQ model is stronger than PAC, the lower bounds in [17] do not not imply hardness in the PAC model, the subject of this paper. We are not aware of any other results on the difficulty of learning intersections of halfspaces.

We are also not aware of any representation-independent hardness results for PAC learning small-depth arithmetic circuits. There is a long line of research establishing lower bounds on the query complexity of polynomial *interpolation* algorithms over various fields, but these do not imply hardness results for the problem of PAC learning polynomials with small representations as arithmetic circuits (see Section 5.1 for more details).

1.3 Our Techniques

Our results exploit recent cryptosystems due to Regev [20, 21], which improve on the security of the Ajtai-Dwork cryptosystem [1]. These cryptosystems are based on the hardness of the well-studied lattice problems uSVP, SVP, and SIVP. As pointed out in [21], an advantage of these problems is the equivalence of the worst-case and average-case complexity. In other words, an efficient algorithm for solving these problems on a nonnegligible (inverse-polynomial) fraction of instances yields an efficient algorithm for solving *every* instance. This contrasts with common number-theoretic problems such as factoring or deciding quadratic residuosity. Furthermore, lattice-based cryptosystems feature decryption functions that are completely different from modular exponentiation $d(Y) = Y^D \bmod N$, the decryption function that is at the heart of virtually every number-theoretic cryptosystem. As a result, lattice-based cryptosystems imply hardness results for learning intersections of halfspaces that number-theoretic cryptosystems have not yielded.

An established method [10] for obtaining hardness results for a concept class $\mathcal{C}$ is to demonstrate that $\mathcal{C}$ can compute the decryption function of a public-key cryptosystem. Intersections of a polynomial number of halfspaces, however, cannot compute the decryption functions of the cryptosystems we use. In fact, the decryption functions in question contain PARITY as a subfunction (see Section 7), which cannot be computed by intersections of a polynomial number of any unate functions [17]. Furthermore, the decryption functions for Regev's cryptosystems perform a division or an iterated addition, which require threshold depth 3 and 2, respectively [26, 23]. Threshold circuits of depth 2 and higher are known to be more powerful than intersections of halfspaces.

To overcome these difficulties, we use non-uniform distributions on $\{0,1\}^n$ to help us with the computation. This technique allows us to use intersections of *degree-*2 *polynomial threshold functions* to compute the decryption function while still obtaining a hardness result for intersections of *halfspaces*. In addition, we construct particularly compact intersections of polynomial threshold functions by treating the private key as a constant rather than an input to the decryption function.

2 Preliminaries

A halfspace in n dimensions is a Boolean function $f = \mathrm{sign}(a_1x_1 + a_2x_2 + \cdots + a_nx_n - \theta)$, where $a_1, \ldots, a_n, \theta$ are integers. It is well known that, without loss of generality, we may assume that the magnitude of each integer is at most $2^{O(n \log n)}$. The intersection of k halfspaces is the Boolean function $g = \bigwedge_{i=1}^k h_i$, where each h_i is a halfspace. A *polynomial threshold function* (PTF) of degree d is a Boolean function $f = \mathrm{sign}(p(x))$ for some real-valued degree-d polynomial p in $x_1, x_2, \ldots, x_n$.

We adopt the *probably approximately correct* (PAC) model of learning, due to Valiant [24]. An overview of this model is as follows. A *concept class* $\mathcal{C}$ is any subset of Boolean functions mapping $\{0,1\}^n \to \{0,1\}$ with polynomial description length (e.g., polynomial-size circuits). Fix a *target function* $f \in \mathcal{C}$ and a distribution μ on $\{0,1\}^n$. The learner, who does not know f, receives labeled examples $(x^1, f(x^1)), (x^2, f(x^1)), \ldots$. Here $x^1, x^2, \cdots \in \{0,1\}^n$ are chosen independently at random according to μ. An algorithm is said to learn $\mathcal{C}$ if, on input $\epsilon \in (0,1)$ and $\mathsf{poly}(n, \frac{1}{\epsilon})$ labeled examples, it outputs a hypothesis h that with high probability has $\Pr_{x\sim\mu}[f(x) \neq h(x)] < \epsilon$. We will be using a looser requirement called *weak learning*, which relaxes the success criterion to $\Pr_{x\sim\mu}[f(x) \neq h(x)] \geqslant \frac{1}{2} + \frac{1}{n^c}$ for a constant c.

2.1 Lattice-based Cryptography

This subsection describes lattice-based cryptography and presents two relevant lattice-based cryptosystems due to Regev [20, 21]. A *lattice* in n dimensions is the set $\{a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n : a_1, \ldots, a_n \in \mathbb{Z}\}$ of all integral linear combinations of a given basis $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$. The primary problems on lattices are the *unique shortest vector problem* $f(n)$-uSVP, *shortest vector problem* $f(n)$-SVP, and *shortest independent vector problem* $f(n)$-SIVP. In $f(n)$-uSVP, the goal is to find the shortest nonzero vector in the lattice, provided that it is shorter by a factor of at least $f(n)$ than any other non-parallel vector. In $f(n)$-SVP, the goal is to approximate the length of the shortest vector within a factor of $f(n)$. Thus, uSVP is a special case of SVP, distinguished by the "uniqueness" condition. Finally, in $f(n)$-SIVP the goal is to approximate (within a factor of $f(n)$) the length of the shortest basis for the lattice, where the length of a basis is the length of its longest vector. Note that all three problems become harder as the approximation factor $1 \leqslant f(n) \leqslant \mathsf{poly}(n)$ decreases.

We will be working with $f(n) = \tilde{O}(n^{1.5})$, an approximation factor for which these three problems are believed to be hard.

The cryptosystems below encrypt one-bit messages (0 and 1). Encryption is randomized; decryption is deterministic. Let $e_{K,r} : \{0,1\} \to \{0,1\}^{\mathsf{poly}(n)}$ denote the encryption function corresponding to a choice of private and public keys $K = (K_{\text{priv}}, K_{\text{pub}})$ and a random string r. In discussing security, we will need the following notion.

Definition 2.1 (Distinguisher). An algorithm $\mathcal{A}$ is said to distinguish between the encryptions of 0 and 1 if for some universal constant c,

$$\left| \Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(1)) = 1] - \Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(0)) = 1] \right| \geqslant \frac{1}{n^c}.$$

We focus on those aspects of the cryptosystems that are relevant to the hardness proofs in this paper. For example, we state the numeric ranges of public and private keys without describing the key generation procedure. We follow the established convention of denoting polynomially-bounded quantities (in n) by lowercase letters, and superpolynomial ones by capital letters.

2.2 The uSVP-based Cryptosystem

We start with a cryptosystem, due to Regev [20], whose security is based on the worst-case hardness of uSVP. Let n be the security parameter. Denote $N = 2^{8n^2}$ and $m = cn^2$, where c is a universal constant. Let $\gamma(n)$ be any function with $\gamma(n) = \omega(n\sqrt{\log n})$, where faster-growing functions γ correspond to improved security but also higher probability of decryption error.

Private key: A real number H with $\sqrt{N} \leqslant H < 2\sqrt{N}$.

Public key: A vector $(A_1, \ldots, A_m, i_0)$, where $i_0 \in \{1, \ldots, m\}$ and each $A_i \in \{0, \ldots, N-1\}$.

Encryption: To encrypt 0, pick a random set $S \subseteq [m]$ and output $\sum_{i \in S} A_i \bmod N$. To encrypt 1, pick a random set $S \subseteq [m]$ and output $\lfloor A_{i_0}/2 \rfloor + \sum_{i \in S} A_i \bmod N$.

Decryption: On receipt of $W \in \{0, \ldots, N-1\}$, decrypt 0 if $\text{frac}(WH/N) < 1/4$, and 1 otherwise. Here $\text{frac}(a) \rightleftharpoons \min\{\lceil a \rceil - a, a - \lfloor a \rfloor\}$ denotes the distance from $a \in \mathbb{R}$ to the closest integer. By a standard argument, the security and correctness of the cryptosystem are unaffected if we change the decryption function to $\text{frac}(AW) < 1/4$, where A is a representation of H/N to within $\mathsf{poly}(n)$ fractional bits.

Correctness: The probability of decryption error (over the choice of private and public keys and the randomness in the encryption) is $2^{-\Omega(\gamma(n)^2/m)}$.

Regev [20] showed that breaking the above cryptosystem would yield a polynomial-time algorithm for uSVP. A more detailed statement follows (see Theorem 4.5 and Lemma 5.4 of [20]):

Theorem 2.2 (Regev [20]). *Assume that one can efficiently distinguish between the encryptions of* 0 *and* 1*. Then there is a polynomial-time solution to every instance of* $(\sqrt{n} \cdot \gamma(n))$-uSVP.

We will set $\gamma(n) = n \log n$ to make the probability of decryption error negligible (inverse-superpolynomial) while guaranteeing $\tilde{O}(n^{1.5})$-uSVP security. Regev's cryptosystem thus improves on the public-key cryptosystem of Ajtai and Dwork [1] whose security is based on the worst-case hardness of $O(n^8)$-uSVP, an easier problem than $\tilde{O}(n^{1.5})$-uSVP.

2.3 The SVP- and SIVP-based Cryptosystem

This second cryptosystem [21] is based on the worst-case *quantum* hardness of SVP and SIVP. Let n be the security parameter. Denote by p a prime with $n^2 < p < 2n^2$, and let $m = 5(n+1)(1+2\log n)$. Let $\gamma(n)$ be any function with $\gamma(n) = \omega(\sqrt{n}\log n)$, where faster-growing functions γ correspond to improved security but also higher probability of decryption error.

Private key: A vector $\mathbf{s} \in \mathbb{Z}_p^n$.

Public key: A sequence of pairs $(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_m, b_m)$, where each $\mathbf{a}_i \in \mathbb{Z}_p^n$ and $b_i \in \mathbb{Z}_p$.

Encryption: To encrypt 0, pick $S \subseteq [m]$ randomly and output $(\sum_{i\in S} \mathbf{a}_i, \sum_{i\in S} b)$. To encrypt 1, pick $S \subseteq [m]$ randomly and output $(\lfloor p/2 \rfloor + \sum_{i\in S} \mathbf{a}_i, \sum_{i\in S} b_i)$. (All arithmetic is modulo p.)

Decryption: On receipt of $(\mathbf{a}, b) \in \mathbb{Z}_p^n \times \mathbb{Z}_p$, decrypt 0 if $b - \langle \mathbf{a}, \mathbf{s} \rangle$ is closer to 0 than to $\lfloor p/2 \rfloor$ modulo p. Decrypt 1 otherwise. (All arithmetic is modulo p.)

Correctness: The probability of decryption error (over the choice of private and public keys and the randomness in the encryption) is $2^{-\Omega(\gamma(n)^2/m)}$.

Regev [21] showed that breaking the above cryptosystem would imply a polynomial-time quantum algorithm for solving SVP and SIVP. A more precise statement is as follows (see Theorem 3.1 and Lemmas 4.4, 5.4 of [21]):

Theorem 2.3 (Regev [21]). *Assume that there is a polynomial-time (possibly quantum) algorithm for distinguishing between the encryptions of* 0 *and* 1. *Then there is a polynomial-time quantum solution to* $\tilde{O}(n \cdot \gamma(n))$-$\mathsf{SVP}$ *and* $\tilde{O}(n \cdot \gamma(n))$-$\mathsf{SIVP}$.

We adopt the setting $\gamma(n) = \sqrt{n}\log^2 n$ to make the probability of decryption error negligible while guaranteeing $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP security. Observe that this second cryptosystem due to Regev [21] is preferable to his first cryptosystem (Regev [20]) in that it is based on the worst-case hardness of a more general lattice problem (SVP vs. uSVP). The disadvantage of the second cryptosystem is that breaking it would only yield a *quantum* algorithm for SVP, as opposed to the first cryptosystem which would yield a standard algorithm for uSVP.

3 Learning Decryption Functions vs. Breaking Cryptosystems

In their seminal paper [10], Kearns and Valiant established a key relationship between the security of a public-key cryptosystem and the hardness of learning an associated concept class. We rederive it below for completeness and extend it to allow for error in the decryption process. This link is a natural consequence of the ease of encrypting messages with the public key. A large pool of such encryptions can be viewed as a set of training examples for learning the decryption function. But learning the decryption function to a nonnegligible advantage would mean breaking the cryptosystem. Assuming the cryptosystem is secure, we can thus conclude that it is not feasible to learn the decryption function. We formalize this observation in the following lemma:

Lemma 3.1 (Cryptography and learning; cf. Kearns & Valiant [10]). *Consider a public-key cryptosystem for encrypting individual bits by n-bit strings. Let $\mathcal{C}$ be a concept class that contains all the decryption functions $\{0,1\}^n \to \{0,1\}$ of the cryptosystem. Let $\varepsilon(n) = \Pr_{K,r}[d_K(e_{K,r}(0)) \neq 0$ or $d_K(e_{K,r}(1)) \neq 1]$ be the probability of decryption error (over the choice of keys and randomization in the encryption). If $\mathcal{C}$ is weakly PAC-learnable in time $t(n)$ with $t(n)\varepsilon(n) = 1/n^{\omega(1)}$, then there is a distinguisher between the encryptions of* 0 *and* 1 *that runs in time $t(n)$.*

Proof. For a pair of keys $K = (K_{\text{priv}}, K_{\text{pub}})$, let $e_{K,r} : \{0,1\} \to \{0,1\}^n$ be the randomized encryption function (indexed by the choice of random string r). Let $d_K : \{0,1\}^n \to \{0,1\}$ denote the matching decryption function. We will use the assumed learnability of $\mathcal{C}$ to exhibit an algorithm $\mathcal{A}$ that runs in time $O(t(n))$ and has

$$\Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(1)) = 1] - \Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(0)) = 1] \geqslant \frac{1}{n^c}$$

for some universal constant c, as long as $t(n)\varepsilon(n) = 1/n^{\omega(1)}$. The probability is taken over the choice of keys, randomness in the encryption, and any internal randomization in $\mathcal{A}$. It follows that $\mathcal{A}$ is the desired distinguisher.

Algorithm $\mathcal{A}$ takes as input a pair (K_{pub}, w), where $w \in \{0,1\}^n$ is the encryption of an unknown bit. First, $\mathcal{A}$ draws $t(n)$ independent training examples, choosing each as follows:

1. Pick $b = 0$ or $b = 1$, with equal probability.
2. Pick r, an unbiased random string.
3. Create training example $\langle e_{K,r}(b), b\rangle$.

Next, $\mathcal{A}$ passes the training examples to the assumed algorithm for learning $\mathcal{C}$. Assume no decryption error has occurred, i.e., the decryption function d_K is consistent with the examples. Then the learning algorithm outputs, with high probability, a hypothesis h that approximates d_K with a nonnegligible advantage:

$$\Pr_{b,r}[h(e_{K,r}(b)) = d_K(e_{K,r}(b))] \geqslant \frac{1}{2} + \frac{1}{n^c},$$

for some constant c. With this hypothesis in hand, algorithm $\mathcal{A}$ outputs $h(w)$ and exits.

It remains to show that $\mathcal{A}$ is indeed distinguisher. We will first handle the case in which no decryption error occurs; call this event $\overline{\mathcal{E}}$. Then:

$$\begin{aligned}
&\Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(1)) = 1 \mid \overline{\mathcal{E}}] - \Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(0)) = 1 \mid \overline{\mathcal{E}}] \\
&\quad = \Pr_{K,r}[h(e_{K,r}(1)) = 1] - \Pr_{K,r}[h(e_{K,r}(0)) = 1] \\
&\quad = \Pr_{K,r}[h(e_{K,r}(1)) = 1] - \left(1 - \Pr_{K,r}[h(e_{K,r}(0)) = 0]\right) \\
&\quad = \Pr_{K,r}[h(e_{K,r}(1)) = 1] + \Pr_{K,r}[h(e_{K,r}(0)) = 0] - 1 \\
&\quad = 2\left(\frac{1}{2}\Pr_{K,r}[h(e_{K,r}(1)) = 1] + \frac{1}{2}\Pr_{K,r}[h(e_{K,r}(0)) = 0]\right) - 1 \\
&\quad = 2\Pr_{K,b,r}[h(e_{K,r}(b)) = b] - 1 \\
&\quad \geqslant 2\left(\Pr_{K,b,r}[h(e_{K,r}(b)) = d_K(e_{K,r}(b))] - \Pr_{K,b,r}[d_K(e_{K,r}(b)) \neq b]\right) - 1 \\
&\quad \geqslant 1 + \frac{2}{n^c} - 2\varepsilon(n) - 1 \\
&\quad = \frac{2}{n^c} - 2\varepsilon(n).
\end{aligned}$$

We now extend the analysis to account for possible decryption errors. Observe that the likelihood of a decryption error on a run of $\mathcal{A}$ is small:

$$\begin{aligned}
\Pr[\mathcal{E}] &= \mathbf{E}_K\left[\Pr_r[\mathcal{E} \mid K]\right] \\
&\leqslant \mathbf{E}_K\left[t(n)\cdot \Pr_{b,r}[d_K(e_{K,r}(b)) \neq b \mid K]\right] && \text{by union bound} \\
&= t(n)\cdot \mathbf{E}_K\left[\Pr_{b,r}[d_K(e_{K,r}(b)) \neq b \mid K]\right] \\
&= t(n)\cdot \Pr_{K,b,r}[d_K(e_{K,r}(b)) \neq b] \\
&\leqslant t(n)\varepsilon(n).
\end{aligned}$$

This upper bound on $\Pr[\mathcal{E}]$, along with the above analysis of the error-free case, allows us to complete the proof of the desired claim:

$$\begin{aligned}
&\Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(1)) = 1] - \Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(0)) = 1] \\
&\quad = \left(\Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(1)) = 1 \mid \overline{\mathcal{E}}] - \Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(0)) = 1 \mid \overline{\mathcal{E}}]\right)\cdot \Pr[\overline{\mathcal{E}}] \\
&\qquad + \left(\Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(1)) = 1 \mid \mathcal{E}] - \Pr_{K,r}[\mathcal{A}(K_{\text{pub}}, e_{K,r}(0)) = 1 \mid \mathcal{E}]\right)\cdot \Pr[\mathcal{E}] \\
&\quad \geqslant \left(\frac{2}{n^c} - 2\varepsilon(n)\right)\cdot \Pr[\overline{\mathcal{E}}] - 1\cdot \Pr[\mathcal{E}] \\
&\quad \geqslant \frac{1}{n^c}.
\end{aligned}$$

□

4 Implementing the Decryption Functions

Section 3 demonstrated that if a public-key cryptosystem is secure, then no concept class that can implement its decryption function is efficiently PAC-learnable. In what follows, we obtain implementations of the decryption functions from Section 2 by *intersections of degree-*2 *PTFs.* This will lead to a hardness result for learning intersections of degree-2 PTFs. We will obtain the main result of the paper by noting that intersections of degree-2 PTFs are no harder to learn than are intersections of halfspaces, a claim we formalize next.

Lemma 4.1. *Assume that intersections of n^ϵ heavy (respectively, light) halfspaces are weakly PAC-learnable. Then for any constant $c > 0$, intersections of n^c heavy (respectively, light) degree-*2 *PTFs are weakly PAC-learnable.*

Proof sketch. We will prove the "light" case only; the "heavy" case is analogous. Consider the following concept classes:

$\mathcal{C}$:	intersections of n^ϵ light halfspaces;
$\mathcal{C}'$:	intersections of n^ϵ light degree-2 PTFs;
$\mathcal{C}''$:	intersections of n^c light degree-2 PTFs.

First observe that a polynomial-time PAC-learning algorithm for $\mathcal{C}$ implies one for $\mathcal{C}'$. This is because a degree-2 PTF in the n variables $x_1, \ldots, x_n$ is a halfspace in the $n + \binom{n}{2}$ variables

$x_1, \ldots, x_n,\ x_1x_2,\ \ldots,\ x_{n-1}x_n$, which yields a polynomial-time map from training/testing examples for a degree-2 PTF to those for a halfspace.

Finally, a polynomial-time PAC-learning algorithm for $\mathcal{C}'$ implies one for $\mathcal{C}''$ via a "padding argument." This is because we can "project" the problem of learning $\mathcal{C}''$ from n into $n^{c/\epsilon}$ dimensions. This projection preserves the polynomial running time of the learning algorithm for $\mathcal{C}'$ (since c and ϵ are constants), while making it so that the number of PTFs is the ϵth power of the dimension. $\square$

4.1 The uSVP-based Cryptosystem

Recall that $\mathrm{frac}(a) \rightleftharpoons \min\{\lceil a \rceil - a, a - \lfloor a \rfloor\}$ denotes the distance from $a \in \mathbb{R}$ to the closest integer. Throughout this section, $\{a\}$ stands for the fractional part of $a \in \mathbb{R}$. Define the Boolean predicate $\text{CLOSE-TO-INT}(a) = 1 \iff \mathrm{frac}(a) < 1/4$. This predicate ignores the integral part of a, meaning that $\text{CLOSE-TO-INT}(a) = \text{CLOSE-TO-INT}(\{a\})$.

The decryption function in the uSVP-based cryptosystem (Section 2) is $d_A(W) = \text{CLOSE-TO-INT}(AW)$, where A is a fixed real number and W is an integer input, both with a polynomial number of bits. We will demonstrate how to implement $\text{CLOSE-TO-INT}(AW)$ with intersections of degree-2 PTFs. A critical ingredient of our implementation is the "interval trick" of Siu and Roychowdhury [23], an insightful idea that the was used in [23] to obtain a depth-2 light-weight threshold circuit for iterated addition.

Lemma 4.2 (Implementing the uSVP-based decryption function). *Let $A > 0$ be a real number with k fractional bits. Then function $f(x) = \text{CLOSE-TO-INT}(A \sum_{j=0}^{n-1} x_j 2^j)$ can be computed by the intersection of k PTFs with degree 2 and weight $O(k^4 4^k)$.*

Proof. Let $\{A\} = .b_1 b_2 \ldots b_k$ be the fractional part of A; the integral part of A is irrelevant. Then

$$\left\{ A \sum_{j=0}^{n-1} x_j 2^j \right\} = \left\{ \sum_{i=1}^{k} \sum_{j=0}^{n-1} b_i x_j 2^{j-i} \right\} \qquad \text{(by definition of } A\text{)}$$

$$= \left\{ \sum_{i=1}^{k} \sum_{j=0}^{\min\{n-1, i-1\}} b_i x_j 2^{j-i} \right\} \quad \text{(drop terms } b_i x_j 2^{j-i} \text{ that are whole numbers).}$$

Denote $S(x) = \sum_{i=1}^{k} \sum_{j=0}^{\min\{n-1,i-1\}} b_i x_j 2^{j-i}$ so that $\{A \sum_{j=0}^{n-1} x_j 2^j\} = \{S(x)\}$. Observe that $S(x)$ is a multiple of $1/2^k$ and ranges between 0 and k. We will use degree-2 PTFs to identify intervals in $[0, k]$ on which $\text{CLOSE-TO-INT}(S(x)) = 1$. A listing of the first few such intervals is as follows:

Values of $S(x)$	Output of CLOSE-TO-INT($S(x)$)
. 0 0 0 0 … 0 0	
⋮	1
. 0 0 1 1 … 1 1	
. 0 1 0 0 … 0 0	
⋮	0
. 1 1 0 0 … 0 0	
. 1 1 0 0 … 0 1	
⋮	1
1 . 0 0 1 1 … 1 1	
1 . 0 1 0 0 … 0 0	
⋮	0
1 . 1 1 0 0 … 0 0	

Each interval $[a,b]$ can be computed by the PTF $(S(x)-\frac{a+b}{2})^2 \leqslant (\frac{b-a}{2})^2$; an integral representation of this PTF has weight $O(k^4 4^k)$. To compute the negation of an interval, we replace the inequality sign by ">". Finally, there are at most $2k+1$ intervals because every two consecutive intervals, starting at the second, cover a distance of 1 on the interval $[0,k]$. By AND'ing the negations of the k intervals on which CLOSE-TO-INT$(S(x)) = 0$, we obtain the desired f as an AND of k weight-$O(k^4 4^k)$ degree-2 PTFs. □

4.2 The SVP- and SIVP-based Cryptosystem

For an integer $a \in \{0,\ldots,p-1\}$, define the Boolean predicate CLOSE-TO-MIDDLE$_p(a) \iff |a - \lfloor p/2 \rfloor| \leqslant \min\{a, p-a\}$. Recall that the decryption function in the SVP- and SIVP-based cryptosystem (Section 2) is $d_{s_1,\ldots,s_n}(a_1,\ldots,a_n,b) = \text{CLOSE-TO-MIDDLE}_p(b - \sum a_i s_i)$, where all s_i, a_i, and b are integers in $\{0,\ldots,p-1\} = \mathbb{Z}_p$. We will show how to compute $d_{s_1,\ldots,s_n}$ with intersections of degree-2 PTFs.

Lemma 4.3 (Implementing the SVP- and SIVP-based decryption function). *Let $d_{s_1,\ldots,s_n} : \left(\{0,1\}^{\log p}\right)^{n+1} \to \{0,1\}$ be the Boolean function defined by*

$$d_{s_1,\ldots,s_n}(x) = \text{CLOSE-TO-MIDDLE}_p\left(\textstyle\sum_{i=0}^{\log p-1} 2^i x_{0,i} - \sum_{j=1}^{n} s_j \sum_{i=0}^{\log p-1} 2^i x_{j,i}\right),$$

where all s_i are integers in $\{0,\ldots,p-1\}$. Then $d_{s_1,\ldots,s_n}$ can be computed by the intersection of $n\log p$ PTFs with degree 2 and weight $O((pn\log p)^2)$.

Proof. Denote

$$S(x) \rightleftharpoons \sum_{i=0}^{\log p-1} 2^i x_{0,i} - \sum_{j=1}^{n}\sum_{i=0}^{\log p-1} (2^i s_j \bmod p) x_{j,i}.$$

Observe that $S(x)$ is just the original weighted sum $\left(\sum_{i=0}^{\log p-1} 2^i x_{0,i} - \sum_{j=1}^{n} s_j \sum_{i=0}^{\log p-1} 2^i x_{j,i}\right)$ with the coefficients reduced modulo p.

Using the definition of CLOSE-TO-MIDDLE$_p$, we have $d_{s_1,\ldots,s_n}(x) = \text{CLOSE-TO-MIDDLE}_p(S(x))$. The integer $S(x)$ ranges between $-(p-1)n\log p$ and $p-1$, a total range of length $< pn\log p$. As in the proof of Lemma 4.2, this range can be divided into consecutive intervals on which $d_{s_1,\ldots,s_n}(x)$ is constant (i.e., does not change value within an interval).

Every two consecutive intervals cover a length of p units. Thus, there are a total of $\leqslant 2(pn\log p)/p = 2n\log p$ consecutive intervals. By picking out the $n\log p$ intervals on which $d_{s_1,\ldots,s_n}(x) = 0$ and AND'ing their negations, we can compute $d_{s_1,\ldots,s_n}$ exactly. It remains to note that the negation of an interval $[a,b]$ can be computed by a degree-2 weight-$O((pn\log n)^2)$ PTF of the form $(S(x) - \frac{a+b}{2})^2 > (\frac{b-a}{2})^2$. □

We additionally observe that the decryption function in the SVP- and SIVP-based cryptosystem can be computed by a depth-3 arithmetic circuit.

Lemma 4.4 (Extension to arithmetic circuits). *Let $d_{s_1,\ldots,s_n} : \left(\{0,1\}^{\log p}\right)^{n+1} \to \{0,1\}$ be the Boolean function defined by*

$$d_{s_1,\ldots,s_n}(x) = \text{CLOSE-TO-MIDDLE}_p\left(\textstyle\sum_{i=0}^{\log p-1} 2^i x_{0,i} - \sum_{j=1}^{n} s_j \sum_{i=0}^{\log p-1} 2^i x_{j,i}\right),$$

where all s_i are integers in $\{0,\ldots,p-1\}$. Then $d_{s_1,\ldots,s_n}$ can be computed by a depth-3 arithmetic circuit of size $\mathsf{poly}(p,n)$.

Proof. Set $S(x)$ as in the proof of Lemma 4.3. Recall that $S(x)$ is an integer in the range $R \rightleftharpoons [-(p-1)n\log p,\ p-1] \cap \mathbb{Z}$ and completely determines the target function: $d_{s_1,\dots,s_n}(x) = \text{CLOSE-TO-MIDDLE}_p(S(x))$.

Let g be a polynomial such that $g(S(x)) = d_{s_1,\dots,s_n}(x)$ for all Boolean inputs x. It can be constructed by interpolating $d_{s_1,\dots,s_n}$ on the range of $S(x)$ via the Lagrange formula:

$$g(y) = \sum_{r \in R} \text{CLOSE-TO-MIDDLE}_p(r) \cdot \prod_{r' \in R, r \neq r} \frac{y - r'}{r - r'}.$$

Since the range R contains $\mathsf{poly}(p,n)$ integers, $g(S(x))$ can be computed by a depth-3 arithmetic circuit of size $\mathsf{poly}(p,n)$ with input $S(x)$ and summation gates at the bottom. But $S(x)$ is a sum of $\mathsf{poly}(p,n)$ terms, each a singleton variable x_i or a constant. Thus, $d_{s_1,\dots,s_n}$ can be computed directly by a depth-3 arithmetic circuit of size $\mathsf{poly}(p,n)$ with inputs x. □

5 Main Results

Based on the assumed hardness of the cryptosystems in Section 2 and the learning-to-cryptography reductions of Sections 3 and 4, we are in a position to prove the desired hardness results for learning intersections of halfspaces.

Theorem 1.1. (Restated from page 2.) *Assume that intersections of n^ϵ heavy halfspaces in n dimensions are PAC-learnable for some constant $\epsilon > 0$. Then there is a polynomial-time solution to $\tilde{O}(n^{1.5})$-*uSVP.

Proof. Let $\mathcal{C}$ denote the concept class of intersections of n^ϵ heavy halfspaces, and let $\mathcal{C}'$ denote the concept class of intersections of n^c heavy degree-2 PTFs (for a large enough constant $c > 0$). By Lemma 4.1, the assumed PAC-learnability of $\mathcal{C}$ implies the PAC-learnability of $\mathcal{C}'$. By Lemma 4.2, the decryption function in the uSVP-based cryptosystem is in $\mathcal{C}'$. A PAC-learning algorithm for $\mathcal{C}'$ would thus yield a distinguisher between the encryptions of 0 and 1 (by Lemma 3.1) and, as a result, an efficient solution to $O(\sqrt{n} \cdot \gamma(n))$-$\mathsf{uSVP}$ for $\gamma(n) = n \log n$ (by Theorem 2.2). □

Theorem 1.2. (Restated from page 2.) *Assume that intersections of n^ϵ light halfspaces in n dimensions are PAC-learnable for some constant $\epsilon > 0$. Then there is a polynomial-time quantum solution to $\tilde{O}(n^{1.5})$-*SVP *and $\tilde{O}(n^{1.5})$-*SIVP.

Proof. Let $\mathcal{C}$ denote the concept class of intersections of n^ϵ light halfspaces, and let $\mathcal{C}'$ denote the concept class of intersections of n^c light degree-2 PTFs (for a large enough constant $c > 0$). By Lemma 4.1, the assumed PAC-learnability of $\mathcal{C}$ implies the PAC-learnability of $\mathcal{C}'$. By Lemma 4.3, the decryption function in the uSVP-based cryptosystem is in $\mathcal{C}'$. A PAC-learning algorithm for $\mathcal{C}'$ would thus yield a distinguisher between the encryptions of 0 and 1 (by Lemma 3.1) and, as a result, an efficient quantum solution to $\tilde{O}(n \cdot \gamma(n))$-$\mathsf{SVP}$ and $\tilde{O}(n \cdot \gamma(n))$-$\mathsf{SIVP}$ for $\gamma(n) = \sqrt{n} \log^2 n$ (by Theorem 2.3). □

Theorems 1.1 and 1.2 both imply a hardness result for learning polynomial-size depth-2 circuits of majority gates, a concept class commonly denoted by $\widehat{\mathsf{LT}}_2$. To prove this, we will need a result regarding heavy vs. light threshold circuits, due to Goldmann, Håstad, and Razborov [7] and Goldmann and Karpinski [8]. Let $\widehat{\mathsf{LT}}_d$ denote the class of depth-d polynomial-size threshold circuits with polynomially-bounded weights. Let $\widetilde{\mathsf{LT}}_d$ denote the class of depth-d polynomial-size threshold circuits in which only the output gate is required to have polynomially-bounded weights.

Theorem 5.1. [7, 8] *For any fixed integer d, $\widehat{\mathsf{LT}}_d = \widetilde{\mathsf{LT}}_d$.*

We are now in a position to prove the desired hardness result for depth-2 neural networks.

Theorem 1.3. (Restated from page 2.) *Assume that depth-2 polynomial-size circuits of majority gates are PAC learnable. Then there is a polynomial-time solution to $\tilde{O}(n^{1.5})$-uSVP and polynomial-time quantum solutions to $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP.*

Proof. Let $\wedge\widehat{\mathsf{LT}}_1$ (respectively, $\wedge\mathsf{LT}_1$) denote the concept classes of intersections of polynomially many light (respectively, heavy) halfspaces. By Theorems 1.1 and 1.2, it suffices to show that $\wedge\widehat{\mathsf{LT}}_1 \subseteq \widehat{\mathsf{LT}}_2$ and $\wedge\mathsf{LT}_1 \subseteq \widehat{\mathsf{LT}}_2$. The first statement is obvious: each halfspace is already a majority gate (with the inputs suitably negated/replicated), and the top gate $\mathrm{AND}(f_1, f_2, \ldots, f_t)$ can be replaced by a majority gate $\mathrm{MAJ}(-t, f_1, f_2, \ldots, f_t)$. To prove that $\wedge\mathsf{LT}_1 \subseteq \widehat{\mathsf{LT}}_2$, observe that $\wedge\mathsf{LT}_1 \subseteq \widetilde{\mathsf{LT}}_2$ (by an argument similar to the first case) and $\widetilde{\mathsf{LT}}_2 = \widehat{\mathsf{LT}}_2$ (by Theorem 5.1). ☐

5.1 Hardness for PAC Learning Arithmetic Circuits

Here we give a hardness result for PAC learning depth-3 arithmetic circuits over the integers. Many researchers have constructed efficient, sparse polynomial interpolation algorithms where the learner has *query* access to the unknown polynomial [16, 19, 22]. If, in addition to membership queries, the learner can make equivalence queries, Klivans and Shpilka [13] have shown how to exactly learn restricted types of depth-3 arithmetic circuits via multiplicity automata techniques [3]. We show that if the learner receives random examples only, then learning depth-3 polynomial-size arithmetic circuits is as hard as solving $\tilde{O}(n^{1.5})$-SVP in quantum polynomial-time (the proof is similar to the proofs of Theorems 1.1 and 1.2):

Theorem 1.4. (Restated from page 2.) *Assume that depth-3 polynomial-size arithmetic circuits are PAC-learnable in polynomial time. Then there is a polynomial-time quantum solution to $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP.*

6 Extensions

We can obtain stronger hardness results by making explicit quantitative assumptions about the hardness of uSVP, SVP, and SIVP.

Assumption 6.1 (Hardness of uSVP). There is a constant $\epsilon > 0$ such that every algorithm for solving $\tilde{O}(n^{1.5})$-uSVP requires time 2^{n^ϵ}.

Assumption 6.2 (Quantum hardness of SVP and SIVP). There is a constant $\epsilon > 0$ such that every quantum algorithm for solving $\tilde{O}(n^{1.5})$-SVP and $\tilde{O}(n^{1.5})$-SIVP requires time 2^{n^ϵ}.

We believe the above assumptions are reasonable. We can use them to sharpen the conclusions of Theorems 1.1–1.4 as follows:

Theorem 6.3. *Let $c > 0$ be a large enough absolute constant. Then Assumption 6.1 implies that there is no polynomial-time algorithm for PAC learning the intersection of $\log^c n$ heavy halfspaces in n dimensions.*

Proof. We first show that under Assumption 6.1, there is a constant $\gamma > 0$ such that every algorithm for learning the intersection of n halfspaces in n dimensions requires time 2^{n^γ}. Indeed, suppose that for every $\gamma > 0$ there were such a learning algorithm with running time less than 2^{n^γ}. By Theorem 1.1, that would yield an algorithm for $\tilde{O}(n^{1.5})$-uSVP with running time $\mathsf{poly}(n, t(\mathsf{poly}(n)))$, which for $\gamma > 0$ sufficiently small violates the 2^{n^ϵ} time lower bound of Assumption 6.1.

We have shown that there is a constant $\gamma > 0$ such that every algorithm for learning the intersection of n halfspaces in n dimensions requires time 2^{n^γ}. Let $c = 2/\gamma$. Then every algorithm for learning the intersection of $\log^c n$ halfspaces in $\log^c n$ dimensions requires time $2^{(\log^c n)^\gamma} = n^{\omega(1)}$. $\square$

Applying an analogous argument to Theorem 1.2 and Assumption 6.2 establishes stronger results for intersections of light halfspaces and for arithmetic circuits:

Theorem 6.4. *Let $c > 0$ be a large enough absolute constant. Then Assumption 6.2 implies that there is no polynomial-time algorithm for PAC learning the intersection of $\log^c n$ light halfspaces in n dimensions.*

Theorem 6.5. *Let $c > 0$ be a large enough absolute constant. Then Assumption 6.2 implies that there is no polynomial-time algorithm for PAC learning a depth-3 arithmetic circuit with $\log^c n$ gates and inputs $x_1, \dots, x_n \in \{0, 1\}$.*

Finally, we obtain a sharper hardness result for learning depth-2 majority circuits.

Theorem 6.6. *Let $c > 0$ be a large enough absolute constant. Then both Assumption 6.1 and Assumption 6.2 imply that there is no polynomial-time algorithm for PAC learning depth-2 circuits with $\log^c n$ majority gates.*

Proof. Combine Theorems 6.3 and 6.4 in the same way Theorem 1.3 combines Theorems 1.1 and 1.2. $\square$

7 Hardness for AC^0?

A natural question to ask is whether our approach could yield hardness results for other concept classes. Particularly interesting candidates are AC^0 and, more ambitiously, polynomial-size DNF formulas. Here we prove that the decryption functions of the Regev cryptosystems contain PARITY as a subfunction and thus are not computable in AC^0.

We start with the easier proof. Recall that the decryption function of the SVP-based cryptosystem is $f_{s_1,\dots,s_n}(a_1, \dots, a_n, b) = \text{CLOSE-TO-MIDDLE}_p(b - \sum a_i s_i)$, where all s_i, a_i, and b are integers in $\{0, \dots, p-1\} = \mathbb{Z}_p$ with $n^2 < p < 2n^2$.

Proposition 7.1 (SVP-based cryptosystem and AC^0). *The decryption function of the SVP-based cryptosystem, $f_{s_1,\dots,s_n}(a_1, \dots, a_n, b) = \text{CLOSE-TO-MIDDLE}_p(b - \sum a_i s_i)$, is not in AC^0.*

Proof. Note that

$$\text{CLOSE-TO-MIDDLE}_p(\tfrac{p-1}{2} \textstyle\sum x_i) = \text{CLOSE-TO-MIDDLE}_p(\tfrac{p}{2} \textstyle\sum x_i) = \text{PARITY}(x_1, \dots, x_n).$$

The first equality holds because $\frac{1}{2} \sum x_i \leqslant \frac{n}{2} \ll p$. Thus, $\text{PARITY}(x_1, \dots, x_n)$ is a subfunction of $\text{CLOSE-TO-MIDDLE}_p(b - \sum a_i s_i)$. Since AC^0 cannot compute PARITY [6, 9], the claim follows. $\square$

Recall now that the decryption function in the uSVP-based cryptosystem is $d_A(X) = \text{CLOSE-TO-INT}(AX)$, where A is a fixed real number and X is an integer input. For convenience, we assume that X has $n + 1$ bits rather than n.

Proposition 7.2 (uSVP-based cryptosystem and AC^0). *The decryption function of the* uSVP*-based cryptosystem, $d_A(X) = \text{CLOSE-TO-INT}(AX)$, is not in* AC^0.

Proof. We will show that $d_A(X)$ computes PARITY on a subset of $\Theta(n/\log n)$ bits from among $x_1, \ldots, x_n$ (when the other bits are set to 0). The claim will follow.

Let $\Delta \rightleftharpoons 3 + \log n$ and $A \rightleftharpoons \sum_{i=0}^{n/\Delta} 2^{-i\Delta-1}$. In what follows, we show that $d_A(X) = \text{PARITY}(x_0, x_\Delta, x_{2\Delta}, \ldots, x_n)$ when $x_i = 0$ for all $i \notin \{0, \Delta, 2\Delta, \ldots, n\}$. Namely,

$$d_A(X) = \text{CLOSE-TO-INT}(AX) = \text{CLOSE-TO-INT}\left(\left(\sum_{i=0}^{n/\Delta} \frac{1}{2^{i\Delta+1}}\right)\left(\sum_{j=0}^{n/\Delta} x_{j\Delta} 2^{j\Delta}\right)\right)$$
$$= \text{CLOSE-TO-INT}\left(\sum_i \sum_{j>i} \frac{x_j 2^{j\Delta}}{2^{i\Delta+1}} + \sum_i \frac{x_i 2^{i\Delta}}{2^{i\Delta+1}} + \sum_i \sum_{j<i} \frac{x_j 2^{j\Delta}}{2^{i\Delta+1}}\right).$$

The first summation features only whole numbers and can thus be dropped. The second summation is precisely $\frac{1}{2}(x_0 + x_\Delta + x_{2\Delta} + \cdots + x_n)$, a multiple of $\frac{1}{2}$. The third summation does not exceed 1/8 (by the choice of Δ and the geometric series) and thus does not affect the result. We obtain:

$$d_A(X) = \text{CLOSE-TO-INT}\left(\frac{x_0 + x_\Delta + x_{2\Delta} + \cdots + x_n}{2}\right).$$

It remains to note that the latter expression is exactly $\text{PARITY}(x_0, x_\Delta, x_{2\Delta}, \ldots, x_n)$. □

References

[1] M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 284–293, New York, NY, USA, 1997. ACM Press.

[2] M. Alekhnovich, M. Braverman, V. Feldman, A. Klivans, and T. Pitassi. Learnability and automatizability. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science (FOCS)*, 2004.

[3] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. On the applications of multiplicity automata in learning. In *FOCS*, pages 349–358, 1996.

[4] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117–127, 1992.

[5] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[6] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[7] M. Goldmann, J. Håstad, and A. A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[8] M. Goldmann and M. Karpinski. Simulating threshold circuits by majority circuits. *SIAM J. Comput.*, 27(1):230–246, 1998.

[9] J. Håstad. *Computational limitations of small-depth circuits.* MIT Press, Cambridge, MA, USA, 1987.

[10] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[11] M. Kharitonov. Cryptographic lower bounds for learnability of boolean functions on the uniform distribution. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 29–36, New York, NY, USA, 1992. ACM Press.

[12] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on theory of computing*, pages 372–381, New York, NY, USA, 1993. ACM Press.

[13] Klivans and Shpilka. Learning arithmetic circuits via partial derivatives. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 2003.

[14] A. Klivans, R. O'Donnell, and R. Servedio. Learning intersections and thresholds of halfspaces. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science*, pages 177–186, 2002.

[15] A. Klivans and R. Servedio. Learning intersections of halfspaces with a margin. In *Proceedings of the 17th Annual Conference on Learning Theory,*, pages 348–362, 2004.

[16] A. Klivans and D. A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *STOC*, pages 216–223, 2001.

[17] A. R. Klivans and A. A. Sherstov. Improved lower bounds for learning intersections of halfspaces. In *Proceedings of the 19th Annual Conference on Learning Theory*, Pittsburg, USA, 2006. To appear.

[18] S. Kwek and L. Pitt. PAC learning intersections of halfspaces with membership queries. *Algorithmica*, 22(1/2):53–75, 1998.

[19] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM J. Comput.*, 24(2):357–368, 1995.

[20] O. Regev. New lattice based cryptographic constructions. In *STOC '03: Proceedings of the thirty-fifth annual ACM Symposium on theory of computing*, pages 407–416, New York, NY, USA, 2003. ACM Press.

[21] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC '05: Proceedings of the thirty-seventh annual ACM Symposium on theory of computing*, pages 84–93, New York, NY, USA, 2005. ACM Press.

[22] Schapire and Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. *JCSS: Journal of Computer and System Sciences*, 52, 1996.

[23] K.-Y. Siu and V. P. Roychowdhury. On optimal depth threshold circuits for multiplication and related problems. *SIAM J. Discrete Math.*, 7(2):284–292, 1994.

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

[25] S. Vempala. A random sampling based algorithm for learning the intersection of halfspaces. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 508–513, 1997.

[26] I. Wegener. Optimal lower bounds on the depth of polynomial-size threshold circuits for some arithmetic functions. *Inf. Process. Lett.*, 46(2):85–87, 1993.