On the Cryptographic Hardness of Learning Single Periodic Neurons

Anonymous Author(s)
Affiliation
Address
email

Abstract

We show a simple reduction which demonstrates the cryptographic hardness of learning a single periodic neuron over isotropic Gaussian distributions in the presence of noise. More precisely, our reduction shows that *any* polynomial-time algorithm (not necessarily gradient-based) for learning such functions under small noise implies a polynomial-time quantum algorithm for solving worst-case lattice problems which form the foundation of lattice-based cryptography. Our core hard family of functions, which are well-approximated by one-layer neural networks, take the general form of a univariate periodic function applied to an affine projection of the data. These functions have appeared in previous seminal works which demonstrate their hardness against gradient-based (Shamir, 2018), and Statistical Query (SQ) algorithms (Song et al., 2017). We show that if (polynomially) small noise is added to the labels, the intractability of learning these functions apply to all polynomial-time algorithms, beyond gradient-based and SQ algorithms, under the aforementioned cryptographic assumptions. Moreover, we demonstrate the necessity of noise in our hardness result by designing a polynomial-time algorithm for learning certain families of such functions under exponentially small adversarial noise. Our proposed algorithm is not a gradient-based or an SQ algorithm, but is rather based on the celebrated Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm.

1 Introduction

The empirical success of Deep Learning has given an impetus to provide theoretical foundations explaining when and why it is possible to efficiently learn from high-dimensional data with neural networks. Currently, there are large gaps between positive and negative results for learning, even for the simplest neural network architectures [44, 22, 11, 20]. These gaps offer a large ground for debate, discussing the extent up to which improved learning algorithms can be designed, or whether a fundamental computational barrier has been reached.

One particular challenge in closing these gaps is establishing negative results for improper learning in the distribution-specific setting, in which the learner can exploit the peculiarities of a known input distribution, and is not limited to outputting hypotheses from the target function class. Over the last few years, authors have successfully developed distribution-specific hardness results in the context of learning neural networks, providing different flavors. On one hand, there have been several results proving the failure of a restricted class of algorithms, such as gradient-based algorithms [37, 36], or more generally Statistical Query (SQ) algorithms [25, 16, 39, 22, 14]. Notably, such results apply to the simplest cases, such as learning one-hidden-layer neural networks over the standard Gaussian input distribution [22, 14]. On the other hand, a different line of work has shown the hardness of learning *two*-hidden-layer neural networks for *any* polynomial-time algorithm by leveraging cryptographic

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

assumptions, such as the existence of local pseudorandom generators (PRGs) with polynomial stretch [12]. Despite such significant advances, important open questions remain, such as whether the simpler case of learning one hidden-layer neural network over standard Gaussian input remains hard for algorithms not captured by the SQ framework. For instance, are non-SQ polynomial-time algorithms, which may inspect individual samples – such as stochastic gradient descent (SGD) [1] – able to learn one-hidden layer neural networks over Gaussian input? Understanding the answer to this question is a partial motivation of the present work.

A key technique for constructing hard-to-learn functions is leveraging "high-frequency" oscillations in high-dimensions. The simplest instance of such functions is given by pure cosines of the form $f(x) = \cos(2\pi\gamma\langle w, x\rangle)$, where we refer to $w \in S^{d-1}$ as its hidden direction, and γ as its frequency. Such functions have already been investigated by previous works [39, 37, 36] in the context of lower bounds for learning neural networks. For these hard constructions, the frequency γ is taken to scale polynomially with the dimension d. Since as a univariate function $\cos(2\pi\gamma t)$ is $O(\gamma)$-Lipschitz, the function f is well-approximated by one-hidden-layer ReLU network of $\mathsf{poly}(\gamma)$-width on any compact set (see e.g. Appendix E), so understanding the hardness of learning such functions is an unavoidable step towards understanding the hardness of learning one-hidden-layer ReLU networks.

In this work, we pursue this line of inquiry, focusing on weakly learning the cosine neuron class over the standard Gaussian input distribution in the presence of noise. Our main result is a proof, via a reduction from a fundamental problem in lattice-based cryptography called the Shortest Vector Problem (SVP), that such learning task is hard for *any* polynomial-time algorithm, based on the widely-believed cryptographic assumption that (approximate) SVP is computationally intractable against quantum algorithms (See e.g., [34, 33, 3, 15, 28, 23] and references therein). Our result therefore extends the hardness of learning such functions from a restricted family of algorithms, such as SQ, to *all* polynomial-time algorithms by leveraging cryptographic assumptions.

The problem of learning cosine neurons with noise can be understood in the broader context of detecting hidden structures in noisy high-dimensional data. Multiple such inference settings, including, for example, the well-known planted clique model [24, 5], exhibit so-called *computational-to-statistical gaps*. These gaps refer to intervals of signal-to-noise ratio (SNR) values where inference of the hidden structure is possible by exponential-time estimators but appears out of reach for any polynomial-time estimator. Following this line of work, we define the SNR of our cosine neuron learning problem to be the inverse of the noise level, and analyze its hardness landscape. As it turns out, weakly learning the cosine neuron class provides a rich landscape, yielding a computational-to-statistical gap based on a worst-case hardness guarantee. We note that this in contrast with the usual study of such gaps where such worst-case hardness guarantees are usually missing and they are mostly based on the refutation of restricted computational classes, such as Sums-of-Squares [8], low-degree polynomial estimators [30], Belief Propagation [7] or local search methods [18]. Additionally, we establish an upper bound for the computational threshold, thanks to a polynomial-time algorithm based on the LLL lattice basis reduction algorithm (See details in Section 4.3), which bypasses the SQ and gradient-based hardness established by previous works (building upon and generalizing a similar story for linear regression [43], [30, Sec. 4.2.1.]).

Related work

Hardness of learning from cryptographic assumptions. Among several previous works [26, 27] which leverage cryptographic assumptions to establish hardness of improper learning, most relevant to our results is the seminal work of Klivans and Sherstov [29] whose hardness results are also based on SVP. To elaborate, they show that learning intersections of halfspaces, which can be seen as neural networks with the threshold activation, is hard based on the worst-case hardness of approximate SVP. Our work differs, though, in several important aspects from theirs. First, and perhaps most importantly, our result holds over the well-behaved Gaussian input distribution over $\mathbb{R}^d$, whereas their hardness utilizes a non-uniform distribution over the Boolean hypercube $\{0, 1\}^d$. Second, at a technical level and in agreement with our continuous input domain and their discrete input domain, we take a different reduction route from SVP. Their link to SVP is the Learning with Errors (LWE) Problem [34], whereas our link in the reduction is the recently developed Continuous Learning with Errors (CLWE) Problem [10]. On another front, very recently, [12] presented an abundance of novel hardness results in the context of improper learning by assuming the mere *existence* of Local Pseudorandom Generators (LPRGs) with polynomial stretch. While the LPRG and SVP assumptions are not directly comparable, we emphasize that we rely on the worst-case hardness of SVP, whereas LPRG assumes average-case hardness. A worst-case hardness assumption is arguably weaker as

it requires only *one* instance to be hard, whereas an average-case hardness assumption requires instances to be hard on average.

Upper and lower bounds against restricted class of algorithms. As mentioned previously, a widely adapted method for proving hardness of learning is through SQ lower bounds [25, 9, 41, 16]. Among previous work, most closely related to our work is [39] and [37], who consider learning linear-periodic function classes which contain cosine neurons. By constructing a different class of hard one-hidden-layer networks, stronger SQ lower bounds over the Gaussian distribution, in terms of both query complexity and noise rate, have been established [22, 14]. Yet, for technical reasons, the SQ model cannot rule out algorithms such as stochastic gradient descent (SGD), since these algorithms can in principle can inspect each sample individually. In fact, [1] carry this advantage of SGD to the extreme and show that SGD is *poly-time universal*. [6] establishes sharp bounds using SGD for weakly learning a single planted neuron, and reveals a fundamental dependency between the regularity of their (dimension-independent) activation function, which they name the "information exponent", and the sample complexity. The regularity of the activation function has been leveraged in several works to yield positive learning results [44, 20, 17, 13, 4, 21, 38].

2 Contributions: the Hardness Landscape of Learning Cosine Neurons

In this work, we thoroughly study the hardness of improperly learning single cosine neurons over isotropic d-dimensional Gaussian data. We study them under the existence of a small amount of adversarial noise per sample, call it $\beta \geq 0$, which we prove is necessary for the hardness to take place. Specifically we study improperly (weakly) learning in the squared loss sense, the function $f(x) = \cos(2\pi\gamma\langle w, x\rangle)$, for some hidden direction $w \in S^{d-1}$, from m samples of the form $z_i = f(x_i) + \xi_i, i = 1, \ldots, m$ where $x_i \overset{\text{i.i.d.}}{\sim} N(0, I_d)$ and arbitrary $|\xi_i| \leq \beta$.

Information-theoretic bounds: We first address the statistical, or also known as information-theoretic, question of understanding for which noise level β one can hope to learn $f(x)$ from polynomially in d many samples, by using computationally unconstrained estimators. Since the range of the functions $f = f_w$ is the interval $[-1, 1]$ it is a trivial observation that for any $\beta \geq 1$ learning is impossible. This follows because the (adversarial) noise could then produce always the uninformative case where $z_i = 0$ for all $i = 1, \ldots, m$.

Our first result (see Section 4.1 for details), is a design and analysis of an algorithm which runs in $O(\exp(d \log(\gamma/\beta)))$ time and satisfies the following property. For any β smaller than a *sufficiently small constant*, the output hypothesis of the algorithm learns the function f with access $O(d \log(\gamma/\beta))$ samples, with high probability. To the best of our knowledge, such an information-theoretic result has not appeared before in the literature of learning a single cosine neuron. We consider this result essential and reassuring as it implies that the learning task is statistically achievable if β is less than a small constant. Therefore, any hardness claim in terms of polynomial-time algorithms aiming to learn this function class is meaningful and implies an essential computational barrier.

Cryptographic hardness under moderate noise: Our second and main result, presented in Section 4.2, is a reduction establishing that (weakly) learning this function class under any β which scales at least inverse polynomially with d, i.e. $\beta \geq d^{-C}$ for some constant $C > 0$, is as hard as a worst-case lattice problem on which the security of lattice-based cryptography is based on.

Theorem 2.1 (Informal)**.** *Consider the function class $\mathcal{F}_\gamma = \{f_{\gamma,w}(x) = \cos(2\pi\gamma\langle x, w\rangle) \mid w \in \mathcal{S}^{d-1}\}$. Weakly learning $\mathcal{F}_\gamma$ over Gaussian inputs $x \sim N(0, I_d)$ under any inverse-polynomial adversarial noise when $\gamma \geq 2\sqrt{d}$ and $\beta = 1/\mathsf{poly}(d)$, is hard, assuming worst-case lattice problems are secure against quantum attacks.*

The exact sense of cryptographic hardness used is that weakly learning the single cosine neuron under the described assumptions, reduces to solving a *worst-case* lattice problem, known as the approximate Shortest Vector Problem (SVP). The approximation factor of SVP obtained in our reduction, is not known to be NP-hard [2], but it is widely believed to be computationally hard against any polynomial-time algorithm, including quantum algorithms. The reduction makes use of a recently developed average-case detection problem, called Continuous Learning with Errors (CLWE) [10] which has been established to be hard under the same hardness assumption on SVP.

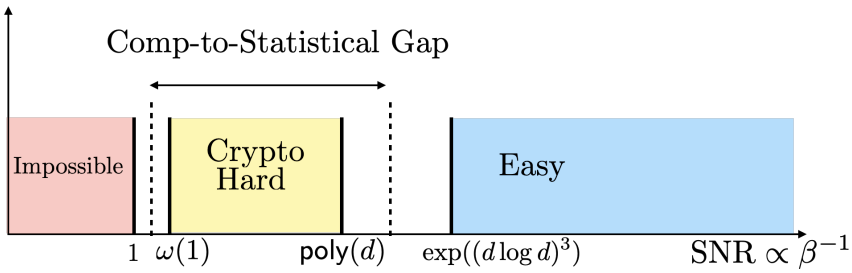

Figure 1: Our results at a glance for weakly learning the class $\mathcal{F}_\gamma$. Section 4.1 describes information-theoretical limits, Section 4.2 presents the reduction from CLWE, while Section 4.3 introduces an efficient algorithm based on LLL.

Our reduction shows that weakly learning the single cosine neuron in polynomial time, implies a polynomial-time algorithm for solving the CLWE problem (see Section 3 for the definition). The link here between the two settings comes from the periodicity of the cosine function, and the fact that the CLWE has an appropriate $\bmod 1$ structure. Interestingly, our reduction works for any class of function $g(x) = \phi(\gamma\langle w, x\rangle)$ where ϕ is a 1-periodic and $O(1)$-Lipschitz function and under $\gamma \geq 2\sqrt{d}$, generalizing the hardness claim much beyond the single cosine neuron. As mentioned also above, note that this is a (conditional) lower bound against *any* polynomial-time estimator, not just SQ or gradient based methods.

Polynomial-time algorithm under small noise: We finally address the question of whether there is some polynomial-time algorithm that can weakly learn the single cosine neuron, in the presence of potentially exponentially small noise. Notably, the current lower bound with respect to SQ [39] or gradient based methods [37] apply without any noise assumption *per-sample*, raising the suspicion that no "standard" learning method works even in the case $\beta = 0$.

We design and analyze a novel learning algorithm for the single cosine neuron which provably succeeds in learning the function $f = f_w$ when $\beta \leq \exp(-\tilde{O}(d^3))$, and with access to only $d+1$ samples. Note that this sample complexity is perhaps surprising: one needs only one more sample than just receiving the samples in the "pure" linear system form $\langle w, x_i\rangle$ instead of $\cos(2\pi\gamma\langle w, x_i\rangle) + \xi_i$. The algorithm comes from reducing the problem to an integer relation detection question and then make a careful use of the powerful Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [31] to solve it in polynomial time. The integer relation detection allows us to recover the (unknown) integer periods naturally occuring because of the periodicity of the cosine, which then allows us to provably learn the hidden direction w simply by solving a linear system and then the function.

The LLL algorithm is a widely celebrated algorithm in computer science and mathematics, which has rarely been used in the learning literature (with the notable recent exceptions [43, 19]) We consider our connection between learning the single cosine neuron, integer relation detection and the LLL algorithm, a potentially interesting algorithmic novelty of the present work.

3 Definitions and Notations

Distribution-specific PAC-learning We consider the problem of learning a sequence of real-valued function classes $\{\mathcal{F}_d\}_{d\in\mathbb{N}}$, each over the standard Gaussian input distribution on $\mathbb{R}^d$, an instance of what is called distribution-specific PAC learning [27, 37]. The input is a multiset of i.i.d. labeled examples $(x, y) \in \mathbb{R}^d \times \mathbb{R}$, where $x \sim N(0, I_d)$, $y = f(x) + \xi$, $f \in \mathcal{F}_d$, and $\xi \in \mathbb{R}$ is some type of observation noise. We denote by $D = D_f$ the resulting data distribution. The goal of the learner is to output an hypothesis $h : \mathbb{R}^d \to \mathbb{R}$ that is close to the target function f in the squared loss sense over the Gaussian input distribution. We say a learning algorithm is *proper* if it outputs an hypothesis $h \in \mathcal{F}_d$. On the other hand, we say a learning algorithm is *improper* if h is not necessarily in $\mathcal{F}_d$ [35]. We omit the index d, when the input dimension is clear from context.

We denote by $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ the squared loss function defined by $\ell(y, z) = (y - z)^2$. For a given hypothesis h and a data distribution D on pairs $(x, z) \in \mathbb{R}^d \times \mathbb{R}$, we define its *population loss* $L_D(h)$

over a data distribution D by

$$L_D(h) = \mathop{\mathbb{E}}_{(x,y)\sim D}[\ell(h(x), y)] \ . \tag{1}$$

Definition 3.1 (Weak learning)**.** *Let $\epsilon = \epsilon(d) > 0$ be a sequence of numbers, $\delta \in (0,1)$ a fixed constant, and let $\{\mathcal{F}_d\}_{d\in\mathbb{N}}$ be a sequence of function classes defined on input space $\mathbb{R}^d$. We say that a (randomized) learning algorithm $\mathcal{A}$ ϵ-weakly learns $\{\mathcal{F}_d\}_{d\in\mathbb{N}}$ over the standard Gaussian distribution if for every $f \in \mathcal{F}_d$ the algorithm outputs a hypothesis h_d such that for large values of d with probability at least $1-\delta$*

$$L_{D_f}(h_d) \le L_{D_f}(\mathbb{E}[f(x)]) - \epsilon \ .$$

Note that $\mathbb{E}[f(x)]$ is the best constant predictor for the data distribution $D = D_f$. Hence, we refer to $L_D(\mathbb{E}[f(x)]) = \mathrm{Var}_{Z\sim N(0,I_d)}(f(Z))$, as the trivial loss, and ϵ as the edge of the learning algorithm.

From simplicity, we refer to an hypothesis as *weakly learning* a function class if it can achieve edge ϵ which is depending inverse polynomially in d.

Periodic Neurons. Let $\gamma = \gamma(d) > 1$ be a sequence of numbers indexed by the input dimension $d \in \mathbb{N}$, and let $\phi : \mathbb{R} \to [-1, 1]$ be an 1-periodic function. We denote by $\mathcal{F}_\gamma^\phi$ the function class

$$\mathcal{F}_\gamma^\phi = \{f : \mathbb{R}^d \to [-1,1] \mid f(x) = \phi(\gamma\langle w, x\rangle), w \in S^{d-1}\} \tag{2}$$

Note that each member of the function class $\mathcal{F}_\gamma^\phi$ is fully characterized by a unit vector $w \in S^{d-1}$. We refer such function classes as *periodic neurons*.

Cosine Learning. We define the *cosine distribution* on dimension d with frequency $\gamma = \gamma(d)$, adversarial noise rate $\beta = \beta(d)$, and hidden direction $w \in S^{d-1}$ to be the distribution of samples of the form $(x, z) \in \mathbb{R}^d \times \mathbb{R}$, where $x \overset{\text{i.i.d.}}{\sim} N(0, I_d)$, some bounded adversarial noise $|\xi| \le \beta$, and

$$z = \cos(2\pi\gamma\langle w, x\rangle) + \xi. \tag{3}$$

The cosine learning problem consists of weakly learning the cosine distribution, per Definition 3.1. This learning problem is the central subject of our analysis. Hence, we slightly abuse notation and denote the corresponding cosine function class by

$$\mathcal{F}_\gamma = \{\cos(2\pi\gamma\langle w, x\rangle) \mid w \in S^{d-1}\}. \tag{4}$$

Continuous Learning with Errors (CLWE) [10] We define the CLWE distribution $A_{w,\beta,\gamma}$ on dimension d with frequency $\gamma = \gamma(d) \ge 0$, and noise rate $\beta = \beta(d) \ge 0$ to be the distribution of samples of the form $(x, z) \in \mathbb{R}^d \times [-1/2, 1/2)$ where $x \overset{\text{i.i.d.}}{\sim} N(0, I_d), \xi \overset{\text{i.i.d.}}{\sim} N(0, \beta)$ and

$$z = \gamma\langle x, w\rangle + \xi \mod 1 \tag{5}$$

Note that for the $\mod 1$ operation, we take the representatives in $[-1/2, 1/2)$. The CLWE problem consists of detecting between i.i.d. samples from the CLWE distribution or the null distribution. In the context of CLWE, we refer to the distribution $N(0, I_d) \times U([-1/2, 1/2))$ as the null distribution and denote it by Q. Given $\gamma = \gamma(d)$ and $\beta = \beta(d)$, we consider a sequence of decision problems $\{\mathrm{CLWE}_{\beta,\gamma}\}_{d\in\mathbb{N}}$, indexed by the input dimension d, in which the learner is given samples from an unknown distribution D such that $D \in \{A_{w,\beta,\gamma} \mid w \in S^{d-1}\}$ with probability $1/2$, and $D = Q$ with probability $1/2$. The algorithm is asked to decide whether $D \in \{A_{w,\beta,\gamma} \mid w \in S^{d-1}\}$ or $D = Q$ in polynomial-time. Under this setup, we define the *advantage* of an algorithm as the difference between its success probability, i.e., correctly guessing D, and $1/2$, which is trivially achieved a random coin flip. We call the advantage *negligible* if it decays superpolynomially. For a more detailed setup of this problem, we refer the reader to Appendix A.

Bruna et al. [10] showed that the CLWE problem is computationally hard even with inverse polynomial noise rate β if $\gamma \ge 2\sqrt{d}$, despite its seemingly mild requirement of non-negligible advantage. In fact, their evidence of computational intractability is based on *worst-case* lattice problems called the Shortest Vector Problem (SVP) [32]. In particular, they showed that distinguishing a *typical* CLWE distribution, where the randomness is over the uniform choice of hidden direction $w \in S^{d-1}$, from the null distribution is as hard as, for instance, solving the *worst* instance of approximate SVP. For a formal definition of the lattice problems, we refer the reader to Appendix A, but note that the (quantum) worst-case hardness of these problems are widely-believed by the community [3].

Theorem 3.2 ([10, Corollary 3.2]). *Let $\beta = \beta(d) \in (0,1)$ and $\gamma = \gamma(d) \geq 2\sqrt{d}$ such that γ/β is polynomially bounded. Then, there is a polynomial-time quantum reduction from $\tilde{O}(d/\beta)$-approximate*[1] SVP *to* $\mathrm{CLWE}_{\beta,\gamma}$.

Weak learning and parameter recovery Note that every element of the function class $\mathcal{F}_\gamma$ is fully characterized by the hidden unit vector $w \in S^{d-1}$. Hence, one possible strategy towards achieving weak learning of the cosine distribution, could be to recover the vector w from samples of the form (3). The following lemma (proven in Appendix G) shows that for any w' sufficiently close to w one can construct an hypothesis that weakly learn the function $f(x) = \cos(2\pi\gamma\langle w, x\rangle)$.

Proposition 3.3. *Suppose $\gamma = \omega(1)$. For any $w' \in S^{d-1}$ with $\min\{\|w - w'\|_2^2, \|w + w'\|_2^2\} \leq 1/(16\pi^2\gamma^2)$, the functions $h_A(x) = \cos(2\pi\gamma\langle A, x\rangle)$, $A \in \{w', w\}$ satisfy for large values of d*

$$\mathbb{E}_{x\sim N(0,I_d)}[\ell((h_w(x), h_{w'}(x))] \leq \mathrm{Var}_{x\sim N(0,I_d)}[(h_w(x))^2] - 1/12.$$

The LLL algorithm and integer relation detection In our algorithmic result, we make use of an appropriate integer relation detection application of the celebrated lattice basis reduction LLL algorithm [31]. We say that for some $b \in \mathbb{R}^n$ the vector $m \in \mathbb{Z}^n \setminus \{0\}$ is an *integer relation* for b if $\langle m, b\rangle = 0$. We make use of the following theorem, and we refer the interested reader to the Appendix D for a complete proof and intuition behind the result.

Theorem 3.4. *Let $n, N \in \mathbb{Z}_{\geq 1}$. Suppose $b \in (2^{-N}\mathbb{Z})^n$. Let also $m \in \mathbb{Z}^n$ be an integer relation of b. Then an appropriate application of the LLL algorithm with input b outputs an integer relation $m' \in \mathbb{Z}^n$ of b with $\|m'\|_2 = O(2^{n/2}\|m\|_2\|b\|_2)$ in time polynomial in n, N and $\log(\|m\|_2\|b\|_\infty)$.*

4 Main Results

In this section we present our main results towards understanding the fundamental hardness of (weakly) learning the single cosine neuron class given by (4). We present our results in terms of signal to noise ratio (SNR) equal to $1/\beta$, where recall that $\beta > 0$ is an upper bound on the level of adversarial noise ξ one may introduce at the samples given by (3). All proofs of the statements are deferred to the appendices of each subsection.

A key correspondence At the heart of our main results are the following simple elementary equalities that hold for all $v \in \mathbb{R}$, and may help the intuition of the reader.

$$\cos(2\pi(v \mod 1)) = \cos(2\pi v) \tag{6}$$

$$\arccos(\cos(2\pi v)) = 2\pi|v \mod 1|\ , \tag{7}$$

where in Eq (7), we recall that our $\mathrm{mod}\ 1$ operation takes representatives in $[-1/2, 1/2)$.

An immediate outcome of these equalities, is a key correspondence between the labels of cosine samples and "phaseless" CLWE samples, per (5). By (6), applying the cosine function to CLWE labels results in the cosine distribution with the same frequency, and hidden direction. Conversely, by (7), applying $\arccos$ to cosine labels results in an arguably harder variant of CLWE, in which the ($\mathrm{mod}\ 1$)-signs of the labels are dropped, with again the same frequency and hidden direction. We say this "phaseless" variant of CLWE is harder than CLWE as we can trivially take the absolute value of CLWE labels to obtain these phaseless CLWE samples, and so an algorithm for solving phaseless CLWE automatically implies an algorithm for CLWE.

We have ignored the issue of additive noise for the sake of simplicity in the above discussion. Indeed, the amount of noise in the samples is a key quantity for characterizing the difficulty of these learning problems and the main technical challenge in carrying the reduction between learning single cosine neurons and CLWE. In subsequent sections, we carefully analyze the interplay between the noise level and the computational difficulty of these learning problems.

4.1 The Possible Regime: A Constant-Noise Tolerant Exponential Algorithm

Before discussing the topic of computational hardness, we address the important first question of identifying the noise levels β under which *some estimator*, running in potentially exponential time,

[1] The notation $\tilde{O}(\cdot)$ ignores logarithmic factors.

Algorithm 1: Information-theoretic recovery algorithm for learning cosine neurons

Input: Real numbers $\gamma = \gamma(d) > 1$, $\beta = \beta(d)$, and a sampling oracle for the cosine distribution (3) with frequency γ, β-bounded noise, and hidden direction w.
Output: Unit vector $\hat{w} \in S^{d-1}$ s.t. $\min\{\|\hat{w} - w\|_2, \|\hat{w} + w\|_2\} = O(\arccos(1-\beta)/\gamma)$.

Let $\tau = \arccos(1-\beta)/(2\pi)$, $\epsilon = 2\tau/\gamma$, $m = 64d\log(1/\epsilon)$, and let $\mathcal{C}$ be an ϵ-cover of the unit sphere S^{d-1}. Draw m samples $\{(x_i, y_i)\}_{i=1}^m$ from the cosine distribution (3).
for $i = 1$ **to** m **do**
  $z_i = \arccos(y_i)/(2\pi)$
for $v \in \mathcal{C}$ **do**
  Compute
  $T_v = \frac{1}{m}\sum_{i=1}^m \mathbb{1}\left[|\gamma\langle v, x_i\rangle - z_i \mod 1| \le 3\tau\right] + \mathbb{1}\left[|\gamma\langle v, x_i\rangle + z_i \mod 1| \le 3\tau\right]$
return $\hat{w} = \arg\max_{v\in\mathcal{C}} T_v$.

can weakly learn the class of interest. Note that any constant level of noise above 1, that is $\beta \ge 1$, would make learning impossible for trivial reasons. Indeed, as the cosine takes values in $[-1, 1]$ if $\beta \ge 1$ all the labels z_i can be transformed to the uninformative 0 value because of the adversarial noise. One can naturally wonder whether any estimator can succeed at the presence of some constant noise level $\beta \in (0, 1)$. In this section, we establish that for sufficiently small but constant $\beta > 0$ weak learning is indeed possible by running an appropriate exponential-time estimator.

Towards establishing this result, we leverage Proposition 3.3, according to which to achieve weak lernability it suffices to construct an estimator that achieves ℓ_2 recovery of w or $-w$ with an ℓ_2 error $O(1/\gamma)$. For this reason, we build an exponential-time algorithm that can provably obtain this ℓ_2 guarantee. Corollary 4.2, then, is immediate from Theorem 4.1, Proposition 3.3 and that $\arccos(1-\beta) \approx \sqrt{\beta}$ for small β.

Theorem 4.1 (Information-theoretic upper bound)**.** *For some constants $c_0, C_0 > 0$ (e.g. $c_0 = 1 - \cos(\frac{1}{200}), C_0 = 2000$) the following holds. Let $d \in \mathbb{N}$ and let $\gamma = \gamma(d) > 1$, $\beta(d) \le c_0$, and $\tau = \arccos(1-\beta)/(2\pi)$. Moreover, let P be data distribution given by (3) with frequency γ, hidden direction w, and noise level β. Then, there exists an $\exp(O(d\log(\gamma/\tau)))$-time algorithm using $O(d\log(\gamma/\tau))$ i.i.d. samples from P that outputs a direction $\hat{w} \in S^{d-1}$ satisfying $\min\{\|\hat{w} - w\|_2^2, \|\hat{w} + w\|_2^2\} \le C_0\tau^2/\gamma^2$ with probability $1 - \exp(-\Omega(d))$.*

Corollary 4.2. *Under the assumptions of Theorem 4.1 there exists some sufficiently small $c_1 > 0$ if $\beta \le c_1$ there exist a $\exp(O(d\log(\gamma/\beta)))$-time algorithm using $O(d\log(\gamma/\beta))$ i.i.d. samples from P that weakly learns the function class $\mathcal{F}_\gamma$.*

The exponential-time algorithm achieving the guarantees of Theorem 4.1 is described in Algorithm 1 and proceeds as following. First, it needs to construct an ϵ-cover $\mathcal{C}$ of the sphere S^{d-1} where $\epsilon = \tau/\gamma$. Note that this step already requires at least exponential-time as any such cover needs to be of exponential size. Furthermore, note that such a construction is indeed possible in exponential time by just drawing $O(N\log N)$ uniform random points on the sphere where $N \approx \epsilon^{-d}$ is the covering number of the sphere. Following that it assigns a score to each point in the cover, call it v, which simply counts the number of samples that could have been possibly produced under the assumption that the vector v was the true hidden vector. The algorithm then outputs the element of maximum score. The analysis then proceeds by a careful probabilistic reasoning to claim that the maximizer needs to land $O(1/\gamma)$-close to the true hidden vector (or its antipode), something true for all β less than a sufficiently small constant. Finally, notice that to properly choose the appropriate quantification of the score assignment the algorithm uses the "key correspondence" (7) to transform the samples into "phaseless CLWE samples", which allowed for a cleaner presentation of the algorithm and an easier analysis. We refer the reader to Appendix B for full details.

4.2 The Cryptographically-Hard Regime: Polynomially Small Noise

Given the results in the previous subsection, we discuss now whether a polynomial-time algorithm can achieve weak learnability of the class $\mathcal{F}_\gamma$ for some noise level β smaller than a constant. For inverse polynomial noise, we answer this in the negative by showing a reduction from CLWE to the problem of weakly learning $\mathcal{F}_\gamma$ to any inverse-polynomial edge. This implies that a polynomial-time

algorithm for weakly learning $\mathcal{F}_\gamma$ would yield polynomial-time quantum attacks against worst-case lattice problems, which are widely believed to be hard against quantum computers. As mentioned in the introduction, our reduction applies with any 1-periodic and $O(1)$-Lipschitz activation ϕ.

Theorem 4.3 (Formal)**.** *Let $d \in \mathbb{N}$, $\gamma = \omega(\sqrt{\log d}), \beta = \beta(d) \in (0,1)$. Moreover, let $L > 0$, let $\phi : \mathbb{R} \to [-1,1]$ be an L-Lipschitz 1-periodic univariate function, and $\tau = \tau(d)$ be such that $\beta/(L\tau) = \omega(\sqrt{\log d})$. Then, a polynomial-time (improper) algorithm that weakly learns the function class $\mathcal{F}_\gamma^\phi = \{f_{\gamma,w}(x) = \phi(\gamma\langle w, x\rangle) \mid w \in \mathcal{S}^{d-1}\}$ over Gaussian inputs $x \overset{i.i.d.}{\sim} N(0, I_d)$ under β-bounded adversarial noise implies a polynomial-time algorithm for* $\mathrm{CLWE}_{\tau,\gamma}$.

By the hardness of CLWE (Theorem 3.2) and our Theorem 4.3, we can immediately deduce the cryptographic hardness of learning the single cosine neuron under inverse polynomial noise.

Corollary 4.4. *Let $d \in \mathbb{N}$, $\gamma = \gamma(d) \geq 2\sqrt{d}$ and $\tau = \tau(d) \in (0,1)$ be such that $\gamma/\tau = \mathsf{poly}(d)$, and $\beta = \beta(d)$ be such that $\beta/\tau = \omega(\sqrt{\log d})$. Then, a polynomial-time algorithm that weakly learns the function class $\mathcal{F}_\gamma$ under β-bounded adversarial noise implies a polynomial-time quantum algorithm for $\tilde{O}(d/\tau)$-approximate* SVP.

Proof sketch of Theorem 4.3. Recall that the goal is to reduce $\mathrm{CLWE}_{\beta,\gamma}$ to the problem of weakly learning the function class $\mathcal{F}_\gamma^\phi$. Now, $\mathrm{CLWE}_{\beta,\gamma}$ is the problem of distinguishing the distribution $A_{w,\tau,\gamma}$ which outputs samples of the form (x, z) where $z = \gamma\langle w, x\rangle + \xi, x \sim N(0, I_d), \xi \sim N(0, \tau)$, for some hidden direction $w \in S^{d-1}$, from the null distribution Q which outputs (x, z) where $x \sim N(0, I_d)$ but $z \sim U[-1/2, 1/2]$ independent from x. Notice that (similar to Eq (6)) the 1-periodicity and the Lipschitzness of ϕ implies that for any $\gamma \geq 0$, $w \in S^{d-1}$, $x \in \mathbb{R}^d$, and $\xi \in \mathbb{R}$,

$$\phi(z_i) = \phi(\gamma\langle w, x\rangle + \xi \mod 1) = \phi(\gamma\langle w, x\rangle + \xi) = \phi(\gamma\langle w, x\rangle) + \tilde{\xi}' , \tag{8}$$

for some $\tilde{\xi} \in [-L|\xi|, L|\xi|]$. Using Eq. (8) one can then directly use m CLWE samples with Gaussian random noise, say, (x_i, z_i), and transform them into m samples from D_w^ϕ with bounded adversarial noise by $L\tau \leq \beta$, by simply considering the pairs $(x_i, \phi(z_i)), i = 1, 2, \ldots, m$.

Let us suppose now we have a learning algorithm that weakly learns the function class $\mathcal{F}_\gamma^\phi$ with β-bounded adversarial noise. Then we can draw m samples from $A_{w,\tau,\gamma}$, transform them as above into samples from D_w^ϕ, run the (robust) learning algorithm on D_w^ϕ, and finally obtain an hypothesis $h = h(x_i, \phi(z_i))$ that weakly learns the function class $\mathcal{F}_\gamma^\phi$. On the other hand, samples from Q have labels z_i independent with x_i and therefore are completely uninformative for the learning problem of interest. In particular, one can never hope to achieve weak learning of the function class $\mathcal{F}_\gamma^\phi$, using the hypothesis function $h = h(x_i, \phi(z_i))$ on the samples (x_i, z_i) now generated from Q. This difference is quantified by the loss of the hypothesis $\mathcal{L}_D(h)$ which in case $D = A_{w,\tau,\gamma}$, is smaller by an inverse polynomial additive factor from the trivial error, while in the case in case $D = Q$ it is lower bounded by the trivial error. This property is what allows indeed to detect between $A_{w,\tau,\gamma}$ and Q, and therefore solve $\mathrm{CLWE}_{\beta,\gamma}$ and complete the reduction. □

4.3 The Easy Regime: Exponentially Small Noise

In this section, in sharp contrast with the previous section, we design and analyze a novel polynomial-time algorithm which provably weakly learns the single cosine neuron with only $d+1$ samples, when the noise is exponentially small. The algorithm is based on the celebrated lattice basis reduction LLL algorithm and its specific application obtaining the integer relation detection guarantee described in Theorem 3.4. Let us also mention that in this subsection, for a real number x and a natural number N, we denote by $(x)_N := \mathrm{sgn}(x)\lfloor 2^N x\rfloor/2^N$. We establish the following result.

Theorem 4.5. *Suppose that $1 \leq \gamma \leq d^K$ for some fixed $K > 0$, and $\beta \leq \exp(-(d\log d)^3)$. The Algorithm 2 with input $(x_i, z_i)_{i=1,\ldots,d+1}$ i.i.d. samples from (3) with frequency γ, hidden direction w and noise level β, outputs $w' \in \mathbb{R}^d$ with $\min\{\|w' - w\|_2, \|w' + w\|_2\} = \exp(-\Omega((d\log d)^3))$ and terminates in $\mathsf{poly}(d)$ steps, with probability $1 - \exp(-\Omega(d))$.*

In particular, by combining our result with Proposition 3.3, one concludes the following result.

Corollary 4.6. *Suppose that $\omega(1) = \gamma = \mathrm{poly}(d)$ and $\beta \leq \exp(-(d\log d)^3)$. Then there exists a polynomial-in-d time algorithm using $d+1$ samples from a single cosine neuron distribution (3), with frequency γ and noise level β, that weakly learns the function class $\mathcal{F}_\gamma$.*

Algorithm 2: LLL-based algorithm for learning the single cosine neuron.

Input: Real numbers $\gamma > 0$, and i.i.d. noisy γ-single cosine neuron samples $\{(x_i, z_i)\}_{i=1}^{d+1}$.
Output: Unit vector $\hat{w} \in S^{d-1}$ such that $\min(\|\hat{w} - w\|, \|\hat{w} + w\|) = \exp(-\Omega((d \log d)^3))$.

for $i = 1$ **to** $d + 1$ **do**
  $z_i \leftarrow \mathrm{sgn}(z_i) \cdot \min(|z_i|, 1)$
  $\tilde{z}_i = \arccos(z_i)/(2\pi) \mod 1$

Construct a $d \times d$ matrix X with columns $x_2, \ldots, x_{d+1}$, and let $N = d^3(\log d)^2$.
if $\det(X) = 0$ **then**
  return $\hat{w} = 0$ and output FAIL

Compute $\lambda_1 = 1$ and $\lambda_i = \lambda_i(x_1, \ldots, x_{d+1})$ given by $(\lambda_2, \ldots, \lambda_{d+1})^\top = X^{-1}x_1$.
Set $M = 2^{3d}$ and $\tilde{v} = \left((\lambda_2)_N, \ldots, (\lambda_{d+1})_N, (\lambda_1 z_1)_N, \ldots, (\lambda_{d+1} z_{d+1})_N, 2^{-N}\right) \in \mathbb{R}^{2d+2}$
Output $(t_1, t_2, t) \in \mathbb{Z}^{d+1} \times \mathbb{Z}^{d+1} \times \mathbb{Z}$ from running the LLL basis reduction algorithm on the lattice generated by the columns of the following $(2d + 3) \times (2d + 3)$ integer-valued matrix,

$$\left(\begin{array}{c|c} M2^N(\lambda_1)_N & M2^N\tilde{v} \\ \hline 0_{(2d+2)\times 1} & I_{(2d+2)\times(2d+2)} \end{array}\right)$$

where $(t_1, t_2, t) \in \mathbb{Z}^{d+1} \times \mathbb{Z}^{d+1} \times \mathbb{Z}$.
Compute $g = \gcd(t_2)$, by running Euclid's algorithm.
if $g = 0 \vee (t_2/g) \notin \{-1, 1\}^{d+1}$ **then**
  return $\hat{w} = 0$ and output FAIL

$\hat{w} \leftarrow \mathrm{SolveLinearEquation}(w', \gamma X^\top w' = (t_2/g)z + (t_1/g))$
return $\hat{w}$ and output SUCCESS.

We now describe that key idea behind the algorithm and its analysis. For the purposes of the sketch let us focus on the noiseless case, explaining at the end how an exponentially small tolerance is possible. In this setting, we receive m samples of the form $z_i = \cos(2\pi\langle w, x_i\rangle), i \in [m]$. The algorithm then uses the arcosine and obtains the "phaseless" CLWE values $\tilde{z}_i$ which according to (7) satisfy for some *unknown* $\epsilon_i \in \{-1, 1\}, K_i \in \mathbb{Z}$ $\langle w, x_i\rangle = \epsilon_i \tilde{z}_i + K_i$. Notice that if we knew the integer values of ϵ_i, K_i, since we know $\tilde{z}_i$, the problems becomes simply solving a linear system for w. The algorithm then leverages the application of the powerful LLL algorithm to perform integer relation detection and identify the values of ϵ_i, K_i, as stated in Theorem 3.4. The way it does it is as follows. It first finds coefficients $\lambda_i, i = 1, 2, \ldots, d + 1$ such that $\sum_{i=1}^{d+1} \lambda_i x_i = 0$ which can be easily computed because we have $d + 1$ vectors in $\mathbb{R}^d$. Then using the definition of $\tilde{z}_i$ the relation between the coefficient implies the identity

$$\sum_{i=1}^{d+1} \epsilon_i \lambda_i \tilde{z}_i + \sum_{i=1}^{d+1} K_i \lambda_i = \sum_{i=1}^{d+1} \lambda_i \langle x_i, w\rangle = 0. \tag{9}$$

In particular, the ϵ_i, K_i are coefficients in an integer relation connecting the *known* numbers $\lambda_i z_i, \lambda_i, i = 1, 2, \ldots, d + 1$. Now, an issue is that as one cannot enter the real numbers as input for the lattice-based LLL, the algorithm truncate the numbers to the first N bits and then hope that post-truncation the the near-minimal integer relation connecting these truncated numbers remain a (small multiple of) ϵ_i, K_i, a sufficient condition so that LLL can identify them based on Theorem 3.4. We establish that indeed this the case and this is the most challenging part of the argument. The argument is based on some careful application of the anticoncentration properties of low-degree polynomials (notice that the λ_i are rational functions of x_i by Cramer's rule), to deduce that the numbers $\lambda_i, \lambda_i z_i$ are in "sufficient general position", in terms of rational independence, for the argument to work. The final step is to prove that the algorithm is able to tolerate some noise level. We establish that indeed if $N = \tilde{\Theta}(d^3)$ then indeed the argument can still work and tolerate $\exp(-\tilde{\Theta}(d^3))$-noise by showing that the near-minimal integer relations remain unchanged under this level of exponentially small noise.

References

[1] Emmanuel Abbe and Colin Sandon. Poly-time universality and limitations of deep learning, 2020.

[2] Dorit Aharonov and Oded Regev. Lattice problems in np ∩ conp. *J. ACM*, 52(5):749–765, September 2005. ISSN 0004-5411. doi: 10.1145/1089023.1089025. URL https://doi.org/10.1145/1089023.1089025.

[3] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. Technical Report NIST Internal or Interagency Report (NISTIR) 8309, National Institute of Standards and Technology, July 2020.

[4] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*, 2018.

[5] Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. *Random Structures & Algorithms*, 13(3-4):457–466, 1998. doi: 10.1002/(SICI)1098-2418(199810/12)13:3/4<457::AID-RSA14>3.0.CO;2-W. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291098-2418%28199810/12%2913%3A3/4%3C457%3A%3AAID-RSA14%3E3.0.CO%3B2-W.

[6] Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. Online stochastic gradient descent on non-convex losses from high-dimensional inference, 2021.

[7] Afonso S. Bandeira, Amelia Perry, and Alexander S. Wein. Notes on computational-to-statistical gaps: predictions using statistical physics, 2018.

[8] B. Barak, S. B. Hopkins, J. Kelner, P. Kothari, A. Moitra, and A. Potechin. A nearly tight sum-of-squares lower bound for the planted clique problem. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 428–437, 2016. doi: 10.1109/FOCS.2016.53.

[9] Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *STOC*, STOC '94, pages 253–262, 1994. ISBN 0-89791-663-8. doi: 10.1145/195058.195147.

[10] Joan Bruna, Oded Regev, Min Jae Song, and Yi Tang. Continuous lwe. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021.

[11] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

[12] Amit Daniely and Gal Vardi. From local pseudorandom generators to hardness of learning. *arXiv preprint arXiv:2101.08303*, 2021.

[13] Ilias Diakonikolas, Surbhi Goel, Sushrut Karmalkar, Adam R Klivans, and Mahdi Soltanolkotabi. Approximation schemes for relu regression. In *Conference on Learning Theory*, pages 1452–1485. PMLR, 2020.

[14] Ilias Diakonikolas, Daniel M Kane, Vasilis Kontonis, and Nikos Zarifis. Algorithms and sq lower bounds for pac learning one-hidden-layer relu networks. In *Conference on Learning Theory*, pages 1514–1539. PMLR, 2020.

[15] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals – dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017. https://eprint.iacr.org/2017/633.

[16] Vitaly Feldman, Elena Grigorescu, Lev Reyzin, Santosh S. Vempala, and Ying Xiao. Statistical algorithms and a lower bound for detecting planted cliques. *J. ACM*, 64(2), 2017.

[17] Spencer Frei, Yuan Cao, and Quanquan Gu. Agnostic learning of a single neuron with gradient descent. *arXiv preprint arXiv:2005.14426*, 2020.

[18] David Gamarnik and Ilias Zadik. The landscape of the planted clique problem: Dense subgraphs and the overlap gap property. *arXiv preprint arXiv:1904.07174*, 2019.

[19] David Gamarnik, Eren C. Kızıldağ, and Ilias Zadik. Inference in high-dimensional linear regression via lattice basis reduction and integer relation detection, 2019.

[20] Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*, 2017.

[21] Surbhi Goel, Sushrut Karmalkar, and Adam Klivans. Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[22] Surbhi Goel, Aravind Gollakota, Zhihan Jin, Sushrut Karmalkar, and Adam Klivans. Superpolynomial lower bounds for learning one-layer neural networks using gradient descent. In *International Conference on Machine Learning*, pages 3587–3596. PMLR, 2020.

[23] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 469–477, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936318. doi: 10.1145/1250790.1250859. URL https://doi.org/10.1145/1250790.1250859.

[24] Mark Jerrum. Large cliques elude the metropolis process. *Random Structures & Algorithms*, 3 (4):347–359, 1992.

[25] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6): 983–1006, November 1998. ISSN 0004-5411.

[26] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, January 1994. ISSN 0004-5411. doi: 10.1145/174644.174647. URL https://doi.org/10.1145/174644.174647.

[27] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 372–381, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897915917. doi: 10.1145/167088.167197. URL https://doi.org/10.1145/167088.167197.

[28] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, September 2005. ISSN 0004-5411. doi: 10.1145/1089023.1089027. URL https://doi.org/10.1145/1089023.1089027.

[29] Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009.

[30] Dmitriy Kunisky, Alexander S. Wein, and Afonso S. Bandeira. Notes on computational hardness of hypothesis testing: Predictions using the low-degree likelihood ratio, 2019.

[31] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[32] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*. The Springer International Series in Engineering and Computer Science. Springer US, 2002. ISBN 978-0-7923-7688-0. doi: 10.1007/978-1-4615-0897-7. URL https://www.springer.com/gp/book/9780792376880.

[33] Daniele Micciancio and Oded Regev. *Lattice-based Cryptography*, pages 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-88702-7. doi: 10.1007/978-3-540-88702-7_5. URL https://doi.org/10.1007/978-3-540-88702-7_5.

[34] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, STOC '05, pages 84–93, 2005. ISBN 1-58113-960-8. doi: 10.1145/1060590.1060603.

[35] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014. ISBN 1107057132.

[36] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *International Conference on Machine Learning*, pages 3067–3075. PMLR, 2017.

[37] Ohad Shamir. Distribution-specific hardness of learning neural networks. *J. Mach. Learn. Res.*, 19(1):1135–1163, 2018.

[38] Mahdi Soltanolkotabi. Learning relus via gradient descent. *arXiv preprint arXiv:1705.04591*, 2017.

[39] Le Song, Santosh Vempala, John Wilmes, and Bo Xie. On the complexity of learning neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/a78482ce76496fcf49085f2190e675b4-Paper.pdf.

[40] Noah Stephens-Davidowitz. *On the Gaussian measure over lattices*. Phd thesis, New York University, 2017.

[41] Balázs Szörényi. Characterizing statistical query learning: Simplified notions and proofs. In Ricard Gavaldà, Gábor Lugosi, Thomas Zeugmann, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 186–200, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04414-4.

[42] Roman Vershynin. *High-dimensional probability: an introduction with applications in data science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. ISBN 9781108415194.

[43] Ilias Zadik and David Gamarnik. High dimensional linear regression using lattice basis reduction. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/ccc0aa1b81bf81e16c676ddb977c5881-Paper.pdf.

[44] Kai Zhong, Zhao Song, Prateek Jain, Peter L Bartlett, and Inderjit S Dhillon. Recovery guarantees for one-hidden-layer neural networks. *arXiv preprint arXiv:1706.03175*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] They are described in detail in Section 1
   (b) Did you describe the limitations of your work? [Yes] Figure 1 highlights where our understanding falls short.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This is a complexity-theoretic work which does not have negative societal impact to the best of our knowledge.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] We provide essential proof sketches in the main text, and delegate the full detailed proof to the appendix due to space constraints.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A] theoretical work
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A] theoretical work
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] theoretical work
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] theoretical work
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]