Hardness of Learning a Single Neuron with Adversarial Label Noise

Ilias Diakonikolas
UW Madison

Daniel Kane
UC San-Diego

Pasin Manurangsi
Google Research

Lisheng Ren
UW Madison

Abstract

We study the problem of distribution-free PAC learning a single neuron under adversarial label noise with respect to the squared loss. For a range of activation functions, including ReLUs and sigmoids, we prove strong computational hardness of learning results in the Statistical Query model and under a well-studied assumption on the complexity of refuting XOR formulas. Specifically, we establish that no polynomial-time learning algorithm, even improper, can approximate the optimal loss value within any constant factor.

1 INTRODUCTION

1.1 Background

The recent success of deep learning has served as a practical motivation for the development of provable efficient learning algorithms for various natural classes of neural networks. Despite extensive investigation, our theoretical understanding of the assumptions under which neural networks are efficiently learnable remains somewhat limited.

In this work, we study arguably the simplest possible setting of learning a *single* neuron, i.e., a real-valued function of the form $\mathbf{x} \mapsto f(\langle \mathbf{w}, \mathbf{x} \rangle)$, where $\mathbf{w}$ is the weight vector of parameters and $f : \mathbb{R} \to \mathbb{R}$ is a fixed non-linear activation function. Concretely, the learning problem is the following: Given i.i.d. samples from a distribution D on $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathbb{R}^n$ is the feature vector and $y \in \mathbb{R}$ is the corresponding label, our goal is to learn the underlying function in L_2^2-norm. That is, the learner's objective is to output a hypothesis $h : \mathbb{R}^n \to \mathbb{R}$ such that $\mathbf{E}_{(\mathbf{x},y)\sim D}[(h(\mathbf{x}) - y)^2]$ is as small as possible, compared to the minimum possible loss $\mathrm{OPT} := \min_{\mathbf{w}\in\mathbb{R}^n} \mathbf{E}_{(\mathbf{x},y)\sim D}[(f(\langle \mathbf{w}, \mathbf{x} \rangle) - y)^2]$. Settings of particular interest for the activation f include the ReLU and sigmoid functions, corresponding to $f(u) = \mathrm{ReLU}(u) \stackrel{\mathrm{def}}{=} \max\{0, u\}$ and $f(u) = \sigma(u) \stackrel{\mathrm{def}}{=} 1/(1 + \exp(-u))$ respectively.

Recall that a learning algorithm is called *proper* if the hypothesis h is restricted to be of the form $h_{\widehat{\mathbf{w}}}(\mathbf{x}) = f(\langle \widehat{\mathbf{w}}, \mathbf{x} \rangle)$. Here we are interested in the complexity of *improper* learning, where no such assumption is imposed on the hypothesis h (beyond the fact that it is efficiently computable).

Let $\mathcal{L}_2(h; D) := \mathbf{E}_{(\mathbf{x},y)\sim D}[(h(x) - y)^2]$ be the L_2^2-loss of a function $h : \mathbb{R}^n \to \mathbb{R}$ with respect to a distribution D on $\mathbb{R}^n \times \mathbb{R}$. For an activation function $f : \mathbb{R} \to \mathbb{R}$, we will denote by $\mathcal{C}_f \stackrel{\mathrm{def}}{=} \{c_{\mathbf{w}} : \mathbb{R}^n \to \mathbb{R} \mid c_{\mathbf{w}}(\mathbf{x}) = f(\langle \mathbf{w}, \mathbf{x} \rangle)\}$ the concept class of a single neuron with respect to f. We can now formally define our learning problem.

Definition 1.1 (α-approximate Learner for $\mathcal{C}_f$)**.** An algorithm $\mathcal{A}$ is an α-approximate learner for $\mathcal{C}_f$ if given sample access to an unknown distribution D on $\mathbb{R}^n \times \mathbb{R}$ and $\epsilon \geq 1/\mathrm{poly}(n)$, $\mathcal{A}$ outputs a hypothesis $h : \mathbb{R}^n \to \mathbb{R}$ that with high probability satisfies $\mathcal{L}_2(h; D) \leq \alpha(n) \cdot \mathrm{OPT} + \epsilon$, where $\mathrm{OPT} := \min_{c\in\mathcal{C}_f} \mathcal{L}_2(c; D)$.

In the realizable case, i.e., when the labels y are consistent with a function in the class, the above learning problem is known to be solvable in polynomial time for a range of activation functions. A line of work, (see Kalai and Sastry 2009; Soltanolkotabi 2017; Yehudai and Shamir 2020) and references therein, has in fact shown that simple algorithms like gradient-descent efficiently converge to an optimal solution under additional assumptions on the marginal distribution $D_{\mathbf{x}}$ on examples.

In this work, we focus on the *agnostic* learning model, where no realizability assumptions are made on the distribution D. Roughly speaking, the agnostic model corresponds to learning in the presence of adversarial label noise.

Proceedings of the 25[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

In the distribution-specific setting, when the marginal distribution on examples $D_{\mathbf{x}}$ is assumed to be "well-behaved" (e.g., log-concave), recent work (Diakonikolas et al., 2020a) developed efficient constant factor approximate learners for ReLUs and other activations; their algorithm runs in time $\mathrm{poly}(n/\epsilon)$ and achieves error of $O(\mathrm{OPT}) + \epsilon$. For the special case of ReLU activations, Goel et al. (2017) gave an algorithm with error $\mathrm{OPT} + \epsilon$ and runtime $\mathrm{poly}(n)2^{\mathrm{poly}(1/\epsilon)}$ that succeeds as long as the distribution on examples is supported on the unit sphere. On the lower bound side, a sequence of recent works (Goel et al., 2019, 2020; Diakonikolas et al., 2020b, 2021) have shown that, even under the Gaussian distribution on examples, achieving error of $\mathrm{OPT} + \epsilon$ requires time $n^{F(1/\epsilon)}$, for some function F with $\lim_{u\to\infty} F(u) = \infty$.

The complexity of learning single neurons in the *distribution-free* (agnostic) model – the focus of this paper – is poorly understood, even for the case of ReLU and sigmoid activations. Known NP-hardness results (Síma, 2002; Manurangsi and Reichman, 2018) only rule out efficient constant factor approximations for *proper* learning algorithms. On the other hand, these results do not preclude the existence of an efficient constant factor *improper* learner. This discussion prompts the following question:

Is there an efficient improper $O(1)$-approximate learner in the distribution-free setting?

The results of this paper rule out this possibility.

1.2 Our Results

In this paper, we provide strong evidence that no efficient constant factor approximation exists for distribution-free agnostic learning of a single neuron. Importantly, our hardness results apply to improper learning, where the learning algorithm is allowed to output any polynomially evaluatable hypothesis.

Concretely, our hardness results apply for any activation function with a finite limit on either side.

Definition 1.2 (Convergent Activation Functions)**.** An activation function $f : \mathbb{R} \to \mathbb{R}$ is *convergent* if $\lim_{u\to\infty} f(u)$ or $\lim_{u\to-\infty} f(u)$ exists, and f is not a constant function.

A wide range of commonly used activation functions in deep learning (see, e.g., (Dubey et al., 2021)) are convergent. This includes the Logistic, Sigmoid, Tanh, ReLU, ELU, Swish, and Mish activations.

Prior to our work, the only known activation for which similar hardness results were known is the sign function, corresponding to the class of linear threshold functions or halfspaces (Daniely, 2016). Our techniques generalize and strengthen this prior work (Daniely, 2016) to a very broad class of activations.

We provide two complementary hardness results: (1) An unconditional hardness result in the Statistical Query (SQ) model, and (2) a reduction-based hardness, starting from a well-known assumption about the hardness of refuting XOR formulas.

SQ Lower Bounds. SQ algorithms are a class of algorithms that are allowed to query expectations of bounded functions on the underlying distribution rather than directly access samples (Definition 3.1). The SQ model was introduced by Kearns (1998) in the context of supervised learning as a natural restriction of the PAC model (Valiant, 1984). Subsequently, the SQ model has been extensively studied in a range of settings (see, e.g., (Feldman, 2016) and references therein). The class of SQ algorithms is broad and captures a range of known supervised learning algorithms. Indeed, several known algorithmic techniques in machine learning are known to be implementable using SQs. These include spectral techniques, moment and tensor methods, local search (e.g., Expectation Maximization), and many others (see, e.g., (Feldman et al., 2017a,b)).

We prove a super-polynomial SQ lower bound ruling out any constant factor approximate learner (see Theorem 3.3 for a detailed formal statement).

Theorem 1.3 (SQ Lower Bound, Informal version)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be any convergent activation and $\alpha > 0$ be any universal constant. Any SQ algorithm that is an α-approximate learner for a single neuron defined by f on $\mathbb{R}^n$ with ℓ_2-weight $O(n)$ requires either $n^{\omega(1)}$ many queries or at least one query with $n^{-\omega(1)}$ accuracy.*

Computational Hardness. Our SQ lower bound serves as an inspiration for our second result: a reduction-based hardness result, ruling out efficient constant-factor approximations. Our result relies on a widely believed assumption on the hardness of strongly refuting random K-XOR formulae. See Assumption 4.3 for the precise statement.

Under this assumption, we establish the following.

Theorem 1.4. *Let $f : \mathbb{R} \to \mathbb{R}$ be any convergent activation and $\alpha > 0$ be any universal constant. Under Assumption 4.3, there is no polynomial-time α-approximate learner for a single neuron defined by f on $\mathbb{R}^n$ with ℓ_2-weight* $\mathrm{poly}(n)$.

We note that under a plausible strengthening of Assumption 4.3, we can even rule out (slightly) super-constant approximation ratios (as a function of the dimension). See the supplementary material.

Applebaum et al. (2008) proved several barriers in proving hardness for improper learning based on worst-case hardness assumption. For example, NP-hardness (via Karp reductions) would imply a collapse of the polynomial hierarchy. As such, for the hardness results proved in our paper, either average-case hardness assumptions (e.g. Assumption 4.3) or a restriction of the learning algorithms (e.g. the SQ model) is likely required.

2 PRELIMINARY

Notation. We will use small boldface characters to denote vectors. For a vector $\mathbf{x} \in \mathbb{R}^n$ and $i \in [n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$, x_i denotes the i-th coordinate of $\mathbf{x}$. For $r \geq 1$, we will use $\|\mathbf{x}\|_r$ for the ℓ_r-norm of the vector $\mathbf{x}$ and $\langle \mathbf{w}, \mathbf{x} \rangle$ for the standard inner product between vectors $\mathbf{w}, \mathbf{x} \in \mathbb{R}^n$.

For a discrete set S, we will use $x \sim_U S$ to denote that x is sampled uniformly at random (u.a.r.) from S. We define the degree-t Veronese mapping as the function $V_t : \mathbb{R}^n \to \mathbb{R}^{(n+1)^t}$ that takes as input n real values and outputs all $(n+1)^t$ monomials of degree at most t (including the degree-0 constant 1).

Learning a Single Neuron. We focus on Convergent Activation functions (Definition 1.2). Throughout this paper, we will assume w.l.o.g. that $\lim_{u \to -\infty} f(u)$ exists, as otherwise one can always treat $-\mathbf{w}$ as $\mathbf{w}$ to make $\lim_{u \to -\infty} f(u)$ finite. We will use $f_{\lim}$ to denote $\lim_{u \to -\infty} f(u)$ and $c_+ \in \mathbb{R}$ to denote a fixed value such that $f(c_+) \neq f_{\lim}$.

Our concept classes of interest are those of single neurons, defined below.

Definition 2.1 (Neuron with Activation Function f)**.** Let $f : \mathbb{R} \to \mathbb{R}$. The function class $\mathcal{C}_f$ of a single neuron with activation function f contain all functions $c_{\mathbf{w}} : \mathbb{R}^n \to \mathbb{R}$ of the form $c_{\mathbf{w}}(\mathbf{x}) = f(\langle \mathbf{w}, \mathbf{x} \rangle)$.

As specified below, all of our hardness results apply even when $\|\mathbf{w}\|_1 \leq \text{poly}(n)$. This is a regime where the underlying learning problem can be solved (inefficiently) using $\text{poly}(n)$ samples. Our hardness results give evidence that no efficient algorithm exists.

Predicates. Our proofs will make essential use of certain Boolean functions that we define here. We will use K-XOR: $\{\pm 1\}^K \to \{\pm 1\}$ to denote the parity function, i.e., $K\text{-XOR}(x_1, \cdots, x_K) := \prod_{i=1}^K x_i$. Additionally, it will be convenient to define a truncated version of the parity function, denoted by (K,t)-XOR, which is defined as follows:

$$(K,t)\text{-XOR}(x_1, \cdots, x_K) := \begin{cases} K\text{-XOR}(x_1, \cdots, x_K) & \text{, if } \sum_{j=1}^K x_j \in [-2t, +2t] \\ -1 & \text{, otherwise.} \end{cases}$$

We will also need the majority function over a predicate. We define $\text{L-MAJ}_P : \{\pm 1\}^{LK} \to \{\pm 1\}$ for a predicate $P : \{\pm 1\}^K \to \{\pm 1\}$ as

$$\text{L-MAJ}_P(\mathbf{x}) := \text{sign}\left(\sum_{i=0}^{L-1} P(x_{iK+1}, \cdots, x_{iK+K}) \right) .$$

Polynomial Approximation. Our reductions and their analysis will make essential use of the following polynomial approximation lemma (see supplementary material for the proof.) For a polynomial p, we write $\|p\|_1$ to denote the sum of the absolute values of its coefficients.

Lemma 2.2. *Let $f : \mathbb{R} \to \mathbb{R}$ be a convergent activation function. Consider a function $g : \mathbb{R} \to \{f_{\lim}, f(c_+)\}$ and $a, b \in \mathbb{Z}$ with $a < b$. Then, for any constant $\nu > 0$, there exist a degree-$O(b-a)$ polynomial $p : \mathbb{R} \to \mathbb{R}$ such that*

(i) $|g(u) - f(p(u))| \leq \nu$ *for* $u \in \mathbb{Z}$ *and* $u \in (a, b)$,

(ii) $|f_{\lim} - f(p(u))| \leq \nu$ *for* $u \notin (a, b)$, *and*

(iii) $\|p\|_1 \leq \max(|a|, |b|)^{O(b-a)}$.

3 SQ HARDNESS

3.1 Basics on SQ Algorithms

A Statistical Query (SQ) algorithm has access to the oracle defined below.

Definition 3.1. Let D be a distribution on labeled examples supported on $X \times \mathbb{R}$, for some domain X. A statistical query is a function $q : X \times \mathbb{R} \to [-1, +1]$. We define STAT to be the oracle that given any such query $q(\cdot, \cdot)$ outputs a value v such that $|v - \mathbf{E}_{(\mathbf{x},y) \sim D} q[(\mathbf{x}, y)]| \leq \tau$, where $\tau > 0$ is the tolerance parameter of the query.

It turns out one can prove unconditional lower bounds on the complexity of SQ algorithms via an appropriate notion of SQ dimension. A lower bound on the SQ dimension of a learning problem provides an unconditional lower bound on the complexity of any SQ algorithm for the problem. Interestingly, in our SQ lower bound construction, we bound the SQ complexity of our problem indirectly, by reducing from a known problem with a known SQ lower bound.

For our SQ hardness result, we reduce from the well-known SQ lower bound for learning K-parities, stated below:

Fact 3.2 (Kearns 1993). *For any constant $c \in (0,1)$ and $K \le n^{1-c}$, any SQ algorithm that can distinguish the following two cases of joint distributions $(\mathbf{x}, y)$ on $\{\pm 1\}^n \times \{\pm 1\}$ requires either $n^{\Omega(K)}$ many queries or at least one query with $n^{-\Omega(K)}$ accuracy.*

- *(YES Case)* $\mathbf{x} \sim_U \{\pm 1\}^n$ *and* $y = K\text{-XOR}(x_{i_1}, \cdots, x_{i_K})$ *for some fixed unknown* $i_1, \cdots, i_K \in [n]$.
- *(NO Case)* $(\mathbf{x}, y) \sim_U \{\pm 1\}^n \times \{\pm 1\}$.

3.2 Formal Theorem and Proof Overview

With the aforementioned background, we can now formally state the main result of this section.

Theorem 3.3. *For any convergent activation function $f : \mathbb{R} \to \mathbb{R}$ the following holds: For any universal constant $\zeta > 0$, any SQ algorithm that can distinguish between the following two cases of joint distribution $(\mathbf{x}, y)$ on $\{\pm 1\}^N \times \{f_{\lim}, f(c_+)\}$ requires either $N^{\omega(1)}$ queries or at least one query with $N^{-\omega(1)}$ accuracy.*

- *(YES Case)* $\mathbf{E}[(y - f(\langle \mathbf{w}, \mathbf{x} \rangle))^2] \le \zeta$ *for some* $\mathbf{w} \in \mathbb{R}^N$ *such that* $\|\mathbf{w}\|_1 = O(N)$.
- *(NO Case)* $y \sim_U \{f_{\lim}, f(c_+)\}$ *independently of* $\mathbf{x}$.

Notice that Theorem 3.3 immediately implies Theorem 1.3, because the NO case of the former clearly implies that any output hypothesis must incur error at least $(f_{\lim} - f(c_+))^2/4$.

Proof Overview. Here we provide an intuitive overview of our SQ lower bound proof. Our proof proceeds in three steps:

- **Truncating XOR.** First, observe that $K\text{-XOR}(x_{i_1}, \cdots, x_{i_K})$ is a function of $x_{i_1} + \cdots + x_{i_K}$. When $x_{i_1}, \cdots, x_{i_K}$ are sampled i.i.d. from $\{\pm 1\}$, the value of $x_{i_1} + \cdots + x_{i_K}$ concentrates within $[-o(K), o(K)]$ with high probability. This allows us to replace K-XOR with its truncated variant (K,t)-XOR (defined in Section 2) that evaluates to a constant value outside of the range $[-2t, +2t]$.

  Notice that this step means that in the YES case we have that $y = (K,t)\text{-XOR}(x_{i_1}, \cdots, x_{i_K})$ with high probability – as opposed to with probability one. This property – and to a less extent the last step – is the source of the final adversarial label noise.
- **Polynomial Approximation.** Since the (K,t)-XOR predicate evaluates to a constant everywhere outside $[-2t, +2t]$, we can replace (K,t)-XOR with a degree-$O(t)$ polynomial that coincides with (K,t)-XOR for $x_{i_1} + \cdots + x_{i_K}$ everywhere.
- **From Polynomial to Learning a Neuron.** Viewing a polynomial as a linear function on the monomials, the hardness result of a polynomial (after the activation function is applied) can be transformed to the desired hardness result of neurons. The only tricky part is how to put the activation function f outside. The idea here is to change the $\{\pm 1\}$ labeled classification problem to a regression problem that only has two different real values, namely $\{f_{\lim}, f(c_+)\}$ for labels. We can then show if a learner can achieve small squared error, it can be used to correctly distinguish the two cases in the original binary classification problem.

In the following subsections, we elaborate on each of the above steps.

3.3 Step I: Truncating XOR

We start by noting that Fact 3.2 yields a similar SQ hardness result for the truncated version of XOR. The only difference is that the YES case only satisfies the predicate with high probability (instead of with probability one). Specifically, we have:

Lemma 3.4. *For any constant $c \in (0,1)$ and $K \le n^{1-c}$, any SQ algorithm that can distinguish the following two cases of joint distributions $(\mathbf{x}, y)$ supported on $\{\pm 1\}^n \times \{\pm 1\}$ either requires at least $n^{\Omega(K)}$ many queries or at least one query with $n^{-\Omega(K)}$ accuracy.*

- *(YES Case) With probability* $1 - \exp(-\Theta(t^2/K))$, *we have that* $y = (K,t)\text{-XOR}(x_{i_1}, \cdots, x_{i_K})$ *for some fixed unknown* $i_1, \cdots, i_K \in [n]$.
- *(NO Case)* $(\mathbf{x}, y) \sim_U \{\pm 1\}^n \times \{\pm 1\}$.

Proof. We claim that the two cases in Fact 3.2 satisfy the conditions in the corresponding cases here. The NO case is immediate. As for the YES case, since $x_{i_1}, \cdots, x_{i_K}$ are picked independently and uniformly at random from $\{\pm 1\}$, standard concentration inequalities imply that with probability $1 - \exp(-\Theta(t^2/K))$ we have that $x_{i_1} + \cdots + x_{i_K} \in [-2t, +2t]$, which indeed gives that $y = (K,t)\text{-XOR}(x_{i_1}, \cdots, x_{i_K}) = K\text{-XOR}(x_{i_1}, \cdots, x_{i_K})$. Thus, if any SQ algorithm solves this (K,t)-XOR problem, then it solves the original K-XOR problem. □

3.4 Step II: Polynomial Approximation

Then by approximating (K,t)-XOR functions by polynomials with the activation function outside, we get the following lemma.

Lemma 3.5. *For any convergent activation function $f : \mathbb{R} \to \mathbb{R}$ the following holds: For any constant $c \in (0,1)$, $t \le K \le n^{1-c}$, and $\beta > 0$, any SQ algorithm that can distinguish the following two cases of joint distributions $(\mathbf{x}, y')$ supported on $\{\pm 1\}^n \times \{f_{\lim}, f(c_+)\}$ either requires $n^{\Omega(K)}$ many queries or at least one query with $n^{-\Omega(K)}$ accuracy.*

- *(YES Case)* $\mathbf{E}[(y - f(p(\mathbf{x})))^2] \le \beta + \exp(-\Theta(t^2/K))$, *where p is a degree-$O(t)$ and $\|p\|_1 = t^{O(t)}$.*
- *(NO Case) $y \sim_U \{f_{\lim}, f(c_+)\}$ independently of $\mathbf{x}$.*

Proof. We reduce from the problem in Lemma 3.4 to this problem. Given an instance of the problem in Lemma 3.4, we replace the labels $-1, +1$ with $f_{\lim}, f(c_+)$ respectively.

For the YES case, first observe that (K,t)-XOR is a function that only depends on the sum of its inputs, i.e., $(K,t)\text{-XOR}(x_{i_1}, \dots, x_{i_K}) = q(x_{i_1} + \cdots + x_{i_K})$, for some function q. Applying Lemma 2.2 with $\nu = \min\{\sqrt{\beta}, |f_{\lim} - f(c_+)|\}, g = q$ and $(a,b) = (-2t - 1, 2t + 1)$, we obtain a degree-$O(t)$ polynomial p with $\|p\|_1 = t^{O(t)}$ satisfying the properties of the lemma.

To calculate $\mathbf{E}[(y - f(p(\mathbf{x})))^2]$, we consider two cases based on whether the original label is equal to (K,t)-$\text{XOR}(x_{i_1}, \dots, x_{i_K})$ or not. If it is indeed equal, then Lemma 2.2 ensures that $|y - f(p(\mathbf{x}))| \le \sqrt{\beta}$, and therefore this contributes to at most β. On the other hand, Lemma 3.4 implies that the unequal case can happen with probability only $\exp(-\Theta(t^2/K))$. Our choice of p also ensures that $|y - f(p(\mathbf{x}))| \le |f_{\lim} - f(c_+)| + \nu \le 2|f_{\lim} - f(c_+)|$, which is a constant (depending only on f). Thus, in total, $\mathbf{E}[(y - f(p(\mathbf{x})))^2] \le \beta + \exp(-\Theta(t^2/K))$.

For the NO case, since the original labels are from $U(\{\pm 1\})$ uniform at random and independent of $\mathbf{x}$, the new labels are from $U(\{f_{\lim}, f(c_+)\})$ uniform at random and independent of $\mathbf{x}$. This completes the proof. □

3.5 Step III: From Polynomial to Learning a Neuron

We have already established an SQ hardness result on agnostic learning a polynomial with an activation function outside. Since a polynomial is linear in the Veronese mapping of the inputs, the above lemma implies the main theorem of the SQ hardness result on agnostic learning a single neuron, as formalized below.

Proof for Theorem 3.3. We reduce from Lemma 3.5 with $\beta = \zeta/2$, $K = C \log n$, where C is a sufficiently large constant, $t = \lceil K^{3/4} \rceil$ so that the $\exp(-\Theta(t^2/K))$ term in the YES case is at most β. Our reduction keeps the label the same while applying the Veronese mapping of degree-$O(t)$ on the sample $\mathbf{x}$ to map it to $\mathbf{x}' \in \{0,1\}^N$, where $N = n^{O(t)}$.

The NO case is immediate. The YES case of Lemma 3.5 together with our choice of parameters implies that $\mathbf{E}[(f(\langle \mathbf{w}, \mathbf{x}' \rangle) - y)^2] \le \zeta$, where $\mathbf{w}$ is the coefficient vector of p and $\|\mathbf{w}\|_1 = \|p\|_1 = t^{O(t)} = \log^{O(t)} n \le O(n^t) \le O(N)$. This completes the proof. □

4 COMPUTATIONAL HARDNESS FROM REFUTING RANDOM XORS

In this section, we prove Theorem 1.4. In Section 4.1, we introduce the required background on the hardness assumption we rely on. In Section 4.2, we present our reduction-based hardness result and prove its correctness.

4.1 Additional Background

To describe our hypothesis, we require some terminology. Here we follow the notation used by Daniely (2016). For a predicate $P : \{\pm 1\}^K \to \{\pm 1\}$, a P-clause C consisting of K literals $\ell_1, \dots, \ell_K$ is evaluated as $P(\ell_1, \dots, \ell_K)$. A P-formula J consists of P-clauses $C_1, \dots, C_m$ over n variables and values $v_1, \dots, v_m \in \{\pm 1\}$. Let $\text{Val}_{\mathbf{z}}(J)$ denote the *value* of assignment $\mathbf{z} : \{\pm 1\}^n$, i.e., the fraction of clauses satisfied by the assignment[1], and let $\text{Val}(J)$ denote the maximum value across all assignments, i.e., $\text{Val}(J) = \max_{\mathbf{z}:\{\pm 1\}^n} \text{Val}_{\psi}(J)$. In a random formula, the literals of $C_1, \dots, C_m$ are picked independently and uniformly at random, i.e., for each clause C consisting of $l_1, \cdots, l_K$, each literal is chosen independently u.a.r. from the set of all literals, and $v_1, \dots, v_m$ are also picked independently and u.a.r. from $\{\pm 1\}$.

We are now ready to define the relevant decision problem.

Definition 4.1. For a predicate P, let $\text{CSP}^{\text{rand},1-\eta}_m(P)$ be the distinguishing problem

[1] A clause (C_i, v_i) is satisfied iff C_i evaluated on $\mathbf{z}$ equals to v_i.

between the following cases of the input P-formula J with m clauses.

- (YES Case) $\mathrm{Val}(J) \geq 1 - \eta$.
- (NO Case) J is a random formula of m P-clauses.

An algorithm is said to solve $\mathrm{CSP}_m^{\mathrm{rand},1-\eta}(P)$ if in the YES case (resp. in the NO case), it outputs YES (resp. NO) with probability at least $3/4$.

We use $\vec{0}_q$ to denote length-q all-zero vector. Here it is convenient to use the vector representation for a clause and an assignment.

Definition 4.2. For a clause C consisting of literals $l_1, \cdots, l_K$, let $\vec{C} \in \{-1, 0, +1\}^{Kn}$ be the concatenation of indicator vectors for literals.

For a given position $j \in [K]$, we represent an assignment $\mathbf{z} \in \{\pm 1\}^n$ by $\mathbf{w}_j^{\mathbf{z}} := \vec{0}_{n(j-1)}\mathbf{z}\vec{0}_{n(K-j)}$.

For example, for $n = 4$, the clause

$$C(\mathbf{z}) = 2\text{-MAJ}((3,t)\text{-XOR}(z_1, \bar{z_3}, z_2), \\ (3,t)\text{-XOR}(\bar{z_4}, \bar{z_1}, z_3))$$

has the vector representation

$$\vec{C} = ((1,0,0,0|0,0,-1,0|0,1,0,0), \\ (0,0,0,-1|-1,0,0,0|0,0,1,0)) \ .$$

Furthermore, if $\mathbf{z} = (-1,1,-1,-1)$, then we have

$$\mathbf{w}_2^{\mathbf{z}} = ((0,0,0,0|-1,1,-1,-1|0,0,0,0) \\ (0,0,0,0|0,0,0,0|0,0,0,0)) \ .$$

An important observation here is that $\langle \mathbf{w}_j^{\mathbf{z}}, \vec{C} \rangle$ is equal to the value of the j-th literal of C.

We now describe our computational assumption, which can be interpreted as noisy versions of Theorem 3.2 generalized to arbitrary (i.e., even non-SQ) algorithms. We remark that we cannot directly use the noiseless version as a computational hardness assumption; if we take non-SQ algorithms into considerations, then Gaussian elimination can solve the problem efficiently. This extra noise introduced here ensures that known algebraic techniques do not work.

Assumption 4.3. *There exist constants $b > 0$ and $\eta \in (0, 1/2)$ such that for any sufficiently large constant $K \in \mathbb{N}$, with $m = O(n^{b\sqrt{K}\log K})$, $\mathrm{CSP}_m^{\mathrm{rand},1-\eta}(K\text{−XOR})$ cannot be solved in $\mathrm{poly}(n,m)$ time.*

Assumption 4.3 is widely believed and is the same as that used by Daniely (2016). The best known efficient algorithm for $\mathrm{CSP}_m^{\mathrm{rand},1-\eta}(K\text{−XOR})$ requires $m \geq \tilde{O}(n^{K/2})$ (Allen et al., 2015), and when $m < O(n^{k/2-\epsilon})$ it is not efficiently solvable via Sum-of-Square relaxations (Schoenebeck, 2008), a strong algorithmic technique that encompasses various convex relaxations.

The above assumption only enables us to rule out $\alpha \cdot \mathrm{OPT}$ for any constant α. In the supplementary material, we also prove hardness for super-constant α, under a stronger (but still plausible) assumption.

4.2 Ruling Out Efficient Constant-Factor Approximate Learners

Here we give the proof for Theorem 1.4 which rules out (even non-SQ) $O(1)$-approximate learners for a single neuron with a convergent activation function.

Proof Overview. While our high-level approach is similar to that of the SQ proof from the previous section, there are several subtle differences, which we outline below.

- **Boosting Completeness via Majority.** While in the YES case of SQ there is no noise at all, our starting assumption (Assumption 4.3) has a fixed noise rate of $\eta > 0$ even in the YES case. If we apply the reduction directly as before, we will only be able to rule out α-approximate agnostic learners for *some* $\alpha > 0$. To prove such a hardness for *all* constant $\alpha > 0$, we "boost" the completeness in the YES case by taking the majority of L clauses, similar to what was done by Daniely (2016). Essentially, this gives us the hardness of $\mathrm{CSP}_m^{\mathrm{rand},1-\zeta}(L\text{-MAJ}_{(K,t)\text{-XOR}})$, where $\zeta > 0$ can now be *any* positive constant.

- **From Clause to Samples & Polynomial Approximation.** So far we have discussed about CSPs, which are not learning problems yet. To arrive at a learning problem, we view each clause (C, v) as a sample, where the sample is its vector representation $\vec{C}$ (Definition 4.2) and v gets mapped to a label. Then, we find a polynomial approximation for the hypothesis (corresponding to an assignment to CSP). We remark that our polynomial approximation is very different from that of Daniely (2016), because halfspaces already have the sign activation function, which nicely represents the majority function. In our setting, we need to come up with a more careful way to approximately represent a $L\text{-MAJ}_{(K,t)\text{-XOR}}$-clause for any convergent activation function.

4.2.1 Step I: Boosting Completeness via Majority

Similar to Daniely (2016), we can prove that $\mathrm{CSP}^{\mathrm{rand},1-\zeta}_m(L\text{-MAJ}_{(K,t)\text{-XOR}})$ is hard for all $\zeta > 0$. In fact, we need a slightly stronger condition in the YES case, namely that the inner (K,t)-XOR-clauses must have the sum of their literals belonging to $[-2t,+2t]$, as stated below.

Lemma 4.4. *Given Assumption 4.3, there exists a constant $b > 0$ such that for arbitrary constant $\zeta \in (0,1)$, sufficiently large constant $L \in \mathbb{N}$ which only depends on ζ, and sufficient large constant $K \in \mathbb{N}$, with $m = O(n^{b\sqrt{K}\log K})$ and $t = \sqrt{K}\log^{2/3} K$, if we are given an input of m (C,y) tuples, where C is a $L\text{-MAJ}_{(K,t)\text{-XOR}}$-clause and $y \in \{\pm 1\}$, the following two cases cannot be distinguished in $\mathrm{poly}(n,m)$ time:*

- *(YES Case) There exists an input assignment $\mathbf{z} \in \{\pm 1\}^n$ such that $1-\zeta$ fraction of the (C,y) tuples satisfies the following properties.*
  - *(i) (C,y) is satisfied by $\mathbf{z}$, and,*
  - *(ii) Every (K,t)-XOR clause in C has the sum of inputs inside the interval $[-2t,2t]$.*
- *(NO Case) For each tuple (C,y), C is sampled u.a.r. from all possible clauses and the label $y \sim_U \{\pm 1\}$.*

4.2.2 Step II: Polynomial Approximation

Next, we view each C as a vector and use polynomial approximation to arrive at the following hardness.

Lemma 4.5. *Let f be a convergent activation function. Given Assumption 4.3, there exists a constant $b > 0$ such that, for any constants $\zeta \in (0,1)$, $\beta > 0$, sufficiently large constant $L \in \mathbb{N}$ which only depends on ζ, and sufficient large constant $K \in \mathbb{N}$, with $m = O(n^{b\sqrt{K}\log K})$ and $t = \sqrt{K}\log^{2/3} K$, if we are given an input of m $(\mathbf{x},y) \in \{-1,0,+1\}^{LKn} \times \{f_{\lim}, f(c_+)\}$ tuples, then the following two cases cannot be distinguished in $\mathrm{poly}(n,m)$ time:*

- *(YES Case) There exists a degree-$O(t+L)$ polynomial p satisfying the following properties.*
  - *(i) $\|p\|_1 \le (Kn)^{O_\zeta(t)}$, and,*
  - *(ii) $1-\zeta$ fraction of tuples satisfy $|f(p(\mathbf{x})) - y| \le \beta$ and the remaining ζ fraction satisfies $|f(p(\mathbf{x})) - y| \le \beta + |f_{\lim} - f(c_+)|$.*
- *(NO Case) $y \sim_U \{f_{\lim}, f(c_+)\}$ independently of $\mathbf{x}$*

Proof. We reduce the problem in Lemma 4.4 to the problem in Lemma 4.5. Given an instance of the problem in Lemma 4.4, we change the $-1,+1$ labels to $f_{\lim}, f(c_+)$ respectively and rewrite each clause C as its vector representation $\vec{C}$ on $\{-1,0,+1\}^{LKn}$. Then the new instance would be of the form

$$(\vec{C}_1, y_1), (\vec{C}_2, y_2), \cdots, (\vec{C}_m, y_m) .$$

For the NO case, it is clear that $y \sim_U \{f_{\lim}, f(c_+)\}$ independently of $\mathbf{x}$. For the YES case, let $\mathbf{z}$ be the assignment that satisfies the condition in Lemma 4.4. Note that the value of a clause C evaluated with $L\text{-MAJ}_{(K,t)\text{-XOR}}$ predicate is then the following.

$$\begin{aligned} C(\mathbf{z}) = \text{L-MAJ}((K,t)\text{-XOR}(\langle \mathbf{w}^{\mathbf{z}}_1, \vec{C}\rangle, \cdots, \langle \mathbf{w}^{\mathbf{z}}_K, \vec{C}\rangle), \\ \cdots, \\ (K,t)\text{-XOR}(\cdots, \langle \mathbf{w}^{\mathbf{z}}_{LK}, \vec{C}\rangle)) . \end{aligned}$$

We select polynomials p_1, p_2 as follows:

- Recall that (K,t)-XOR can be written as $q(s)$, where s is the sum of its inputs. Let p_1 be a degree-$O(t)$ polynomial such that $p_1(u) = q(u)$ for all $u \in \{-2t, -2t+2, \ldots, 2t\}$ and $p_1(u) < -3L$ for all $u \notin [-2t,+2t]$. Note that such polynomial p_1 exists according to Lemma A.1.
- Let $g : \mathbb{Z} \to \{f_{\lim}, f(c_+)\}$ be such that

  $$g(u) = \begin{cases} f(c_+) & \text{if } u \in [0,L], \\ f_{\lim} & \text{otherwise.} \end{cases}$$

  We then apply Lemma 2.2 to construct a degree-$O(L)$ polynomial p_2.

From the YES condition in Lemma 4.4, for $1-\zeta$ fraction of clauses (C,v), its i-th (K,t)-XOR-clause C'_i has its input sum s_i lie in $[-2t,+2t]$ for all $i \in [L]$, and the clause C itself is satisfied. The condition (ii) means that $p_1(s_i) = C'_i(\mathbf{z})$ for all $i \in [L]$, while the condition (i) implies that $|f(p_2(C'_1(\mathbf{z}) + \cdots + C'_L(\mathbf{z})) - y| < \nu$, where $y \in \{f_{\lim}, f(c_+)\}$ is the new label. Combining these, we have that

$$|f(p_2(p_1(s_1) + \cdots + p_1(s_L))) - y| < \nu .$$

For the remaining ζ fraction of clauses, it is simple to verify that $p_1(s_1) + \cdots + p_1(s_L)$ is either an integer in $[-2t,+2t]$ or is less than $-L$. In both cases, Lemma 2.2 guarantees that

$$|f(p_2(p_1(s_1) + \cdots + p_1(s_L))) - y| < |f_{\lim} - f_{c_+}| + \nu .$$

Finally, notice that

$$\begin{aligned} &p_2(p_1(s_1) + \cdots + p_1(s_L)) \\ &= p_2\left(\sum_{i=1}^{L} p_1\left(\sum_{j=(i-1)K+1}^{iK} \langle \mathbf{w}^{\mathbf{z}}_j, \vec{C}\rangle\right)\right) , \end{aligned}$$

which can be viewed as a degree-$O(t+L)$ polynomial p in $\vec{C}$ because p_1 is a degree-$O(t)$ polynomial and p_2 is a degree-$O(L)$ polynomial.

Next we will show $\|p\|_1 \leq (Kn)^{O_\zeta(t)}$. Note that $\langle \mathbf{w}_i^{\mathbf{z}}, \vec{C} \rangle$ here has $\|\mathbf{w}_i^{\mathbf{z}}\|_1 = n$ for all i. Note that p_1's degree is $O(t)$ and $\|p_1\|_1 = t^{O(t)}$ and is independent of n. Moreover, p_2 only depends on L, thus p_2 only depends on ζ and is fixed with respect to K and n. Also note that p_2's degree is $O(L)$. Therefore, $\|p\|_1 \leq \|p_2\|_1 \cdot \left(L \cdot \|p_1\|_1 \cdot (Kn)^{O(t)}\right)^{O(L)} \leq \left(t^{O(t)} \cdot (Kn)^{O(t)}\right)^{O(L)} \leq (Kn)^{O_\zeta(t)}$. □

4.2.3 Step III: From Polynomial to Learning a Neuron

The above is a computational hardness result on agnostic learning of a polynomial with an activation function outside. Since a polynomial is linear in the Veronese mapping of the inputs, the above lemma implies the main theorem of the computational hardness result on agnostic learning a single neuron.

Theorem 4.6. *Let f be a convergent activation function. Given Assumption 4.3, for any constant $\zeta' > 0$ and $m = \mathrm{poly}(N)$, if we are given an input of m $(\mathbf{x}, y) \in \{-1, 0, +1\}^N \times \{f_{\mathrm{lim}}, f(c_+)\}$ tuples, then the following two cases cannot be distinguished in $\mathrm{poly}(N)$ time:*

- *(YES Case) There exists $\mathbf{w}$ such that*
  - *(i) $\|w\|_1 \leq N^{O_{\zeta'}(1)}$, and,*
  - *(ii) $\mathbf{E}[(y - f(\langle \mathbf{w}, \mathbf{x} \rangle))^2] \leq \zeta'$.*
- *(NO Case) $y \sim_U \{f_{\mathrm{lim}}, f(c_+)\}$ independently of $\mathbf{x}$.*

Proof. Suppose we have a $\mathrm{poly}(N)$-time algorithm A for solving the problem here with $m = N^u$ tuples. We let $\zeta = \frac{\zeta'}{2(f_{\mathrm{lim}} - f(c_+))^2}$ and take β to be a sufficiently small constant such that $\beta^2 \leq \zeta'/4$ and $\frac{(|f_{\mathrm{lim}} - f(c_+)| + \beta)^2}{(f_{\mathrm{lim}} - f(c_+))^2} \leq 3/2$. Then select L (depending on ζ) as specified in Lemma 4.5. Let $d = O(t+L) = O_{\zeta'}(\sqrt{K} \log^{2/3} K)$ be the degree of the polynomial p in Lemma 4.5, $N = (LKn)^{O(d)}$ be the dimension after applying degree-d Veronese mapping, and select K to be a sufficiently large constant so that $n^{b\sqrt{K} \log K} \geq N^u$.

Given an instance of the problem in Lemma 4.5, we apply the Veronese mapping of degree-d on $\mathbf{x}$. Note that A runs in $\mathrm{poly}(N)$ time, which is also $\mathrm{poly}(n)$ time, since $N = (LKn)^{O(t+L)}$. Furthermore, our choice of parameters ensures that there are sufficiently many samples for A.

We claim that the two cases in Lemma 4.5 satisfy the conditions in the corresponding cases here. For both cases, there are at least N^u tuples. The NO case here is immediate. For the YES case, $\|w\|_1 = \|p\|_1 \leq (Kn)^{O_\zeta(t)} \leq (LKn)^{O_\zeta(t)} \leq N^{O_{\zeta'}(1)}$, since $N = (LKn)^{O(t+L)}$. For the expected error, we can write $\mathbf{E}[(y - h(\mathbf{x}))^2] \leq \zeta(|f_{\mathrm{lim}} - f(c_+)| + \beta)^2 + (1-\zeta) \cdot \beta^2 \leq \zeta'$. This completes the proof. □

Theorem 1.4 then follows from Theorem 4.6.

Proof of Theorem 1.4. Suppose we have an algorithm A that is an α-approximate agnostic learner and runs in time $O(N^t)$. Then, we claim that we can solve the problem in Theorem 4.6 for $\zeta' \leq \frac{(f_{\mathrm{lim}} - f(c_+))^2}{16\alpha}$ and $m = N^{t+1}$. The inputs distribution for A will be the uniform distribution on the set of m tuples in Theorem 4.6 with $\epsilon = \frac{(f_{\mathrm{lim}} - f(c_+))^2}{16}$.

In the YES case, w.h.p. A will return a hypothesis of squared error at most $\frac{(f_{\mathrm{lim}} - f(c_+))^2}{8}$. In the NO case, no hypothesis can have a nontrivial advantage over the constant hypothesis $h(\mathbf{x}) = \frac{f_{\mathrm{lim}} + f(c_+)}{2}$ on the support it has not seen. Since A can see at most $O(N^t)$ samples among all the N^{t+1} samples, with sufficiently large N, A cannot return a hypothesis with error smaller than $\frac{(f_{\mathrm{lim}} - f(c_+))^2}{6}$. Thus, A will return a hypothesis of squared error at least $\frac{(f_{\mathrm{lim}} - f(c_+))^2}{6}$. Therefore, we can distinguish the two cases in Theorem 4.6 for $\zeta' \leq \frac{(f_{\mathrm{lim}} - f(c_+))^2}{16\alpha}$ and $m = N^{t+1}$ with algorithm A. This contradicts Assumption 4.3. □

5 Conclusions and Further Work

In this paper, we prove that for a single neuron with a convergent activation function, any algorithm achieving error $\alpha \cdot \mathrm{OPT}$ (for constant α) must require superpolynomial running time under a plausible assumption on an average-case hardness of CSPs. In the supplementary material, we also prove hardness for superconstant α, under a stronger (but still plausible) assumption. An immediate open question here would be whether one can prove the stronger result in the supplementary material under the weaker Assumption 4.3.

Another interesting future direction would be to expand the understanding to the case where the activation function is not convergent. In some cases (e.g. identity activation), the problem is clearly computationally feasible. Is there a characterization of the activation functions for which the learning problem is hard? And for which values of α do the hardness results hold?

References

S. R. Allen, R. O'Donnell, and D. Witmer. How to refute a random CSP. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015*, pages 689–708. IEEE Computer Society, 2015.

Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 211–220. IEEE, 2008.

A. Daniely. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the 48th Annual Symposium on Theory of Computing, STOC 2016*, pages 105–117, 2016.

I. Diakonikolas, S. Goel, S. Karmalkar, A. R. Klivans, and M. Soltanolkotabi. Approximation schemes for relu regression. In *Conference on Learning Theory, COLT 2020*, volume 125 of *Proceedings of Machine Learning Research*, pages 1452–1485. PMLR, 2020a.

I. Diakonikolas, D. Kane, and N. Zarifis. Near-optimal SQ lower bounds for agnostically learning halfspaces and relus under gaussian marginals. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020b.

I. Diakonikolas, D. M. Kane, T. Pittas, and N. Zarifis. The optimality of polynomial regression for agnostic learning under gaussian marginals in the SQ model. In *Conference on Learning Theory, COLT 2021*, volume 134 of *Proceedings of Machine Learning Research*, pages 1552–1584. PMLR, 2021.

S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. A comprehensive survey and performance analysis of activation functions in deep learning. *CoRR*, abs/2109.14545, 2021. URL `https://arxiv.org/abs/2109.14545`.

V. Feldman. Statistical query learning. In *Encyclopedia of Algorithms*, pages 2090–2095. 2016.

V. Feldman, E. Grigorescu, L. Reyzin, S. Vempala, and Y. Xiao. Statistical algorithms and a lower bound for detecting planted cliques. *J. ACM*, 64 (2):8:1–8:37, 2017a.

V. Feldman, C. Guzman, and S. S. Vempala. Statistical query algorithms for mean vector estimation and stochastic convex optimization. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 1265–1277. SIAM, 2017b.

S. Goel, V. Kanade, A. Klivans, and J. Thaler. Reliably learning the relu in polynomial time. In *Conference on Learning Theory*, pages 1004–1042, 2017.

S. Goel, S. Karmalkar, and A. R. Klivans. Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 8582–8591, 2019.

S. Goel, A. Gollakota, and A. R. Klivans. Statistical-query lower bounds via functional gradients. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

A. Kalai and R. Sastry. The isotron algorithm: High-dimensional isotonic regression. In *COLT 2009*, 2009.

M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

M. J. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proc. 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 392–401. ACM, 1993.

P. Manurangsi and D. Reichman. The computational complexity of training relu (s). *arXiv preprint arXiv:1810.04207*, 2018.

G. Schoenebeck. Linear level lasserre lower bounds for certain k-csps. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 593–602. IEEE Computer Society, 2008.

J. Síma. Training a single sigmoidal neuron is hard. *Neural Comput.*, 14(11):2709–2728, 2002.

M. Soltanolkotabi. Learning relus via gradient descent. In *Advances in neural information processing systems*, pages 2007–2017, 2017.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

G. Yehudai and O. Shamir. Learning a single neuron with gradient methods. In *Conference on Learning Theory*, pages 3756–3786. PMLR, 2020.

Supplementary Material: Hardness of Learning a Single Neuron with Adversarial Label Noise

A Proof of Lemma 2.2

Here we give a more general version of Lemma 2.2 which will be useful in proving a stronger hardness of learning result in the subsequent sections.

Lemma A.1. *Let $f : \mathbb{R} \to \mathbb{R}$ be a convergent activation function. We define $f_{\mathrm{gap}}(c_-) = \sup_{u \le c_-} |f_{\mathrm{lim}} - f(u)|$. Consider a function $g : \mathbb{R} \to \{f_{\mathrm{lim}}, f(c_+)\}$ and $a, b \in \mathbb{Z}$ with $a < b$. Then, for any $c_- < c_+$, there exists a degree-$O(b-a)$ polynomial $p : \mathbb{R} \to \mathbb{R}$ such that*

(i) $|g(u) - f(p(u))| \le f_{\mathrm{gap}}(c_-)$ *for* $u \in \mathbb{Z}$ *and* $u \in (a, b)$,

(ii) $|f_{\mathrm{lim}} - f(p(u))| \le f_{\mathrm{gap}}(c_-)$ *for* $u \notin (a, b)$, *and*

(iii) $\|p\|_1 \le O(c_-) \cdot \max(|a|, |b|)^{O(b-a)}$.

Proof. We may assume without loss of generality that $b - a$ is even (as otherwise, we may consider $a - 1, b$ instead, where we let $g(a-1) = f_{\mathrm{lim}}$). We will construct a polynomial p such that

1. For $u \in \mathbb{Z}$ and $u \in (a, b)$, $p(u) = c_+$ if $g(u) = f(c_+)$ and $p(u) = c_-$ if $g(u) = f_{\mathrm{lim}}$.
2. For $u \notin (a, b)$, $p(u) \le c_-$.

One can construct such a polynomial p through the following process:

1. First, for each $u \in \mathbb{Z}$ and $u \in (a, b)$, we construct a polynomial p_u that
   (a) $p_u(u) = 1$,
   (b) for $v \in \mathbb{Z}$, $v \in (a, b)$ and $v \ne u$, $p_u(v) = 0$,
   (c) for $v \notin (a, b)$, $p_u(v) \le 0$.

   To construct p_u, we start by taking the Lagrange basis function

   $$q_u(v) = \prod_{i \in \mathbb{Z}, i \in [a,b] \text{ and } i \ne u} \frac{v - i}{u - i} .$$

   It is not hard to see that this satisfies the first two conditions. Notice also that the degree of q_u is $b - a$ which is even. Now consider the following cases:

   - Case I: the leading coefficient of q_u is negative. Here we may take $p_u = q_u$ as it already satisfies the third condition.
   - Case II: the leading coefficient of q_u is positive. In this case, we may take $p_u(v) = q_u(v) \cdot \frac{(v-a)(b-v)}{(u-a)(b-u)}$. It is not hard to see that the first two conditions remain true. Furthermore, the degree of p_u remains even and its leading coefficient is now negative, which implies the third condition.

   Thus, in both cases, we have constructed the desired p_u and its degree is at most $b - a + 2$.

2. We define the polynomial p' as follows

   $$p'(v) = \sum_{u \in \mathbb{Z}, u \in (a,b) \text{ and } g(u) = f(c_+)} p_u(v) .$$

3. Finally, we construct our desired polynomial p as $p(v) = (c_+ - c_-) \cdot p'(v) + c_-$.

Then observe that such a polynomial p satisfies properties (i) and (ii). Also notice that $\|p_u\|_1 = \max(|a|, |b|)^{O(b-a)}$ for all u, thus, $\|p\|_1 = O(c_-) \cdot \max(|a|, |b|)^{O(b-a)}$. □

By the definition of f, f_{gap} must be a monotone nondecreasing function and $\lim_{u \to -\infty} f_{\text{gap}}(u) = 0$. Therefore, taking c_- to be a sufficiently small universal constant such that $f_{\text{gap}}(c_-) \le \nu$, Lemma A.1 yields Lemma 2.2.

B Proof of Lemma 4.4

To prove Lemma 4.4, firstly, we use the concentration property of the sum of inputs to get the following lemma.

Lemma B.1. *Given Assumption 4.3, there exist constants $b > 0$ and $\eta \in (0, 1/2)$ such that for any sufficiently large constant $K \in \mathbb{N}$, with $m = O(n^{b\sqrt{K}\log K})$, if we are given an input of m $(C, 1)$ tuples where C is a (K, t)-XOR clause, then the following two cases cannot be distinguished in $\text{poly}(n, m)$ time:*

- *(YES Case) There exists an input assignment $\mathbf{z} \in \{\pm 1\}^n$ such that $1 - \eta - \exp(-\Omega(t^2/K))$ fraction of the $(C, 1)$ tuples satisfies the following properties.*
  - *(i) $(C, 1)$ is satisfied by $\mathbf{z}$, and,*
  - *(ii) The (K, t)-XOR clause C has the sum of inputs inside the interval $[-2t + 2, 2t - 2]$.*
- *(NO Case) For each tuple $(C, 1)$, C is sampled u.a.r. from all possible clauses.*

Proof. We reduce the problem in Assumption 4.3 to the problem above. Given an instance of the problem in Assumption 4.3 as a sequence of tuples

$$((C_1, y_1), \cdots, (C_m, y_m)) .$$

For each tuple (C, y), we randomly negate every literal in the clause C with probability $1/2$ independently. Then we negate y if we had negated an odd number of literals in C, and leave y unchanged otherwise. If this process ends up with $y = -1$, we negate the first literal again and change y to $+1$. This gives a new sequence of tuples as an instance for the problem here

$$((C'_1, 1), \cdots, (C'_m, 1)) .$$

We claim if the original instance is a YES instance (resp. a NO instance), then this new instance is also in the YES case (resp. the NO case). The argument for the NO case here is immediate. For the YES case, there is a value assignment $\mathbf{z}$ that makes $1 - \eta$ fraction of constraints satisfied with the K-XOR predicate. Since we flipped both the clauses and the labels accordingly, $\mathbf{z}$ still makes $1 - \eta$ fraction of constraints satisfied with the K-XOR predicate in the new instance. For each clause $C' = (l_1, \cdots, l_K)$, $l_i \sim_U \{\pm 1\}$ for all $i \neq 1$, thus, $\mathbf{Pr}[l_1 + \cdots + l_k \notin [-2t + 2, 2t - 2]] = \exp(-\Omega(t^2/K))$. Noting that $l_1 + \cdots + l_k \in [-2t + 2, 2t - 2]$ implies that $k\text{−XOR}(l_1, \cdots, l_K) = (k, t)\text{−XOR}(l_1, \cdots, l_K)$. Therefore, $\mathbf{z}$ makes $1 - \eta - \exp(-\Omega(t^2/K))$ fraction of constraints satisfied with (K, t)-XOR predicate in the new instance. □

Proof of Lemma 4.4. We reduce the problem in Lemma B.1 to the problem in Lemma 4.4. Given an instance of the problem in Lemma B.1 is

$$((C_1, 1), \cdots, (C_m, 1)) .$$

We first randomly permute the sequence so the clauses that cannot be satisfied will be randomly distributed. Then we divide them to buckets so there are L clauses in each bucket and treat each bucket as an L-$\text{MAJ}_{(k,t)\text{-XOR}}$ clause.

$$(((C_1, \cdots, C_L), 1), \cdots, ((\cdots, C_m), 1)) . \tag{1}$$

Then for each $((C_1, \cdots, C_L), 1)$ tuple, with $1/2$ probability we negate the label and negate the first literal of each C. With the remaining $1/2$ probability, we leave the tuple unchanged. This gives a new instance for the problem in Lemma 4.4 as

$$(((C'_1, \cdots, C'_L), y_1), \cdots, ((\cdots, C'_m), y_{m'})) . \tag{2}$$

We claim if the original instance is in the YES case (resp. the NO case), then this new instance is in the YES case (resp. is the NO case). The NO case here is immediate. For the YES case, there is a value assignment $\mathbf{z}$ that makes $1-\eta-\exp(-\Omega(t^2/K))$ fraction of the K-XOR clauses satisfying the following properties.

(i) $(C, 1)$ is satisfied by $\mathbf{z}$, and,

(ii) The (K,t)-XOR clause C has the sum of its inputs inside the interval $[-2t+2, 2t-2]$.

With $t = \sqrt{K}\log^{2/3} K$ and a sufficient large constant K, $\eta + \exp(-\Omega(t^2/K))$ is at most a constant in $(0, 1/2)$. Then, using the Chernoff bound, one can show that the same $\mathbf{z}$ makes $1-\zeta$ fraction of L-MAJ$_{(k,t)\text{-XOR}}$ clauses in (1) satisfied for arbitrary small constant $\zeta \in (0,1)$ with L being a sufficiently large constant depending on ζ. Consider we flipped both the clauses and labels accordingly in (1) and for each K-XOR clause, we changed at most one literal. Thus, $\mathbf{z}$ makes $1-\zeta$ fraction of L-MAJ$_{(k,t)\text{-XOR}}$ clauses in (2) satisfy the following properties.

(i) $((C'_1, \cdots, C'_L), y)$ is satisfied by $\mathbf{z}$, and,

(ii) Every (K,t)-XOR clause C' has the sum of inputs inside the interval $[-2t, 2t]$. □

C Larger Inapproximability Ratio under Stronger Assumption

In this section, we prove a stronger (super constant) inapproximability ratio under a stronger computational hardness assumption. Our improved result applies to convergent activation functions that converge "sufficiently quickly". To formalize this notion, we introduce the definition of *fast convergent* Convergent Activation function:

Definition C.1 (Fast Convergent Activation Functions)**.** Let f be a Convergent Activation function. Without loss of generality, assume that $\lim_{u\to-\infty} f(u)$ exists. We say that f is *fast convergent* if there exist a constant $c < 0$ such that for $u < c$ we have that $|f_{\text{lim}} - f(u)| \le (-u)^{-\Omega(1)}$. We also define $f_{\text{gap}}(c_-) = \sup_{u\le c_-} |f_{\text{lim}} - f(u)|$. Note from the definition of f, for $u < c$, we have that $f_{\text{gap}}(u) = (-u)^{-\Omega(1)}$.

It is easy to see that most of the commonly used activations such as the Logistic, Sigmoid, Tanh, ReLU, ELU, Swish, and Mish satisfy the condition above.

Remark C.2. *We note our results also hold under a weaker convergence speed. Specifically, we only need that there exists a constant $c < 0$ such that for $u < c$ it holds that $|f_{\text{lim}} - f(u)| = \exp(-\Omega(\log^{1/2}(-u)))$.*

C.1 SQ Hardness

We formally state the main theorem of this section here.

Theorem C.3. *For any fast convergent Convergent Activation function $f : \mathbb{R} \to \mathbb{R}$ the following holds: For any constant $\nu \in (0, 1/2)$, any SQ algorithm that can distinguish between the following two cases of joint distribution $(\mathbf{x}, y)$ on $\{\pm 1\}^N \times \{f_{\text{lim}}, f(c_+)\}$ requires either $N^{\omega(1)}$ queries or at least one query with $N^{-\omega(1)}$ accuracy.*

- *(YES Case)* $\mathbf{E}[(y - f(\langle \mathbf{w}, \mathbf{x}\rangle))^2] = \exp(-\Omega(\log^{\nu} N))$ *for some* $\mathbf{w} \in \mathbb{R}^N$ *such that* $\|\mathbf{w}\|_1 = O(N)$.
- *(NO Case)* $y \sim_U \{f_{\text{lim}}, f(c_+)\}$ *independently of* $\mathbf{x}$.

Proof Overview. To prove our theorem, we will follow the approach of Step II and Step III in Section 3, using Lemma A.1 and Definition C.1 instead.

C.1.1 Step II: Polynomial Approximation

Lemma C.4. *For any fast convergent activation function $f : \mathbb{R} \to \mathbb{R}$ the following holds: For any constant $c \in (0,1)$, $t \le K \le n^{1-c}$ and $c_- : \mathbb{N} \to (-\infty, c_+)$, any SQ algorithm that can distinguish the following two cases of joint distributions $(\mathbf{x}, y)$ supported on $\{\pm 1\}^n \times \{f_{\text{lim}}, f(c_+)\}$ either requires $n^{\Omega(K)}$ many queries or at least one query with $n^{-\Omega(K)}$ accuracy.*

- *(YES Case)* $\mathbf{E}[(y - f(p(\mathbf{x})))^2] \le f^2_{\text{gap}}(c_-(n)) + O(f_{\text{gap}}(c_-(n))) + \exp(-\Omega(t^2/K))$*, where p is a degree-$O(t)$ and* $\|p\|_1 = O(c_-(n)) \cdot t^{O(t)}$.

- *(NO Case)* $y \sim_U \{f_{\lim}, f(c_+)\}$ *independently of* $\mathbf{x}$.

Proof. We reduce from the problem in Lemma 3.4 to this problem using the same methods as in the proof of Lemma 3.5. The only difference here is that we apply Lemma A.1 instead of Lemma 2.2. □

C.1.2 Step III: From Polynomial to Learning a Neuron

Considering a polynomial is linear in the Veronese mapping of the inputs, we have Theorem C.3.

Proof of Theorem C.3. We reduce from Lemma C.4 by applying the Veronese mapping of degree-$O(t)$ on the sample $\mathbf{x}$ to map it to $\mathbf{x}' \in \{0,1\}^N$, where $N = n^{O(t)}$. The labels are kept the same. Our choice of parameter is $K = \log n$, $t = \log^b n$ so that the $\exp(-\Omega(t^2/K))$ term in the YES case is at most $\exp(-\Omega(\log^\nu N))$, i.e., $\frac{2b-1}{b+1} \geq \nu$. We let $c_-(n) = -N^{1/2}$ so that the $f_{\mathrm{gap}}^2(c_-(n))$ term and $O(f_{\mathrm{gap}}(c_-(n)))$ term in the YES case is at most $N^{-\Omega(1)}$.

The NO case is immediate. The YES case of Lemma 3.5 together with our choice of parameters implies that $\mathbf{E}[(f(\langle \mathbf{w}, \mathbf{x}' \rangle) - y)^2] = \exp(-\Omega(\log^\nu N))$, where $\mathbf{w}$ is the coefficient vector of p and $\|\mathbf{w}\|_1 = \|p\|_1 = N^{1/2} t^{O(t)} = N^{1/2} \log^{O(t)} n = O(N^{1/2} n^{t/2}) = O(N)$. This completes the proof. □

C.2 Computational Hardness

In this section, we prove a computational hardness result with a larger (super-constant) inapproximability ratio by introducing a stronger assumption.

Assumption C.5. *There exists a constant $b > 0$ such that, for $\eta = 2^{-\log^{0.5} n}$ and $K = \log n$, with $m = n^{bK}$, $\mathrm{CSP}_m^{\mathrm{rand}, 1-\eta}(K\text{-XOR})$ cannot be solved in $\mathrm{poly}(n, m)$ time.*

Then we formally state the main theorem of this section.

Theorem C.6. *Let $f : \mathbb{R} \to \mathbb{R}$ be any fast convergent activation function and $\alpha = \exp(O(\log^{\nu'} n))$, where $\nu' \in (0, 1/2)$ is an universal constant. Under Assumption C.5, there is no polynomial-time α-approximate learner for a single neuron defined by f on $\mathbb{R}^n$ with ℓ_2-weight $\mathrm{poly}(n)$.*

Proof Overview. To prove the theorem, we will follow the template of Step II and Step III in Section 4 using Lemma A.1, Definition C.1 and Assumption C.5 instead. Note that Step I can be skipped, as we do not need to boost the completeness here.

C.2.1 Step II: Polynomial Approximation

First, we need to truncate the K-XOR functions as (K, t)-XOR functions in order to approximate them with a polynomial.

Lemma C.7. *Given Assumption C.5, there exists a constant $b > 0$ such that for $\eta = 2^{-\log^{0.5} n}$ and $0 < t < K = \log n$, with $m = n^{bK}$, if we are given an input of m (C, y) tuples where C is a (K, t)-XOR clause, then the following two cases cannot be distinguished in $\mathrm{poly}(n, m)$ time:*

- *(YES Case) There exists an input assignment $\mathbf{z} \in \{\pm 1\}^n$ such that $1 - \eta - \exp(-\Omega(t^2/K))$ fraction of the (C, y) tuples satisfies the following properties.*
  - *(i) (C, y) is satisfied by $\mathbf{z}$, and,*
  - *(ii) The (K, t)-XOR clause C has the sum of inputs inside the interval $[-2t, 2t]$.*
- *(NO Case) For each tuple (C, y), C is sampled u.a.r. from all possible clauses and the label $y \sim_U \{\pm 1\}$.*

Proof. We reduce from the problem in Assumption C.5. Given an instance of the problem in Assumption C.5 as a sequence of tuples

$$((C_1, y_1), \cdots, (C_m, y_m)) .$$

For each tuple (C, y), we randomly negate every literal in the clause C with probability $1/2$ independently. Then we negate y if we had negated an odd number of literals in C, and leave y unchanged otherwise. This gives a new sequence of tuples as an instance for the problem here

$$((C'_1, y'_1), \cdots, (C'_m, y'_m)) .$$

We claim if the original instance is in the YES case (resp. the NO case), then this new instance is in the YES case (resp. the NO case). The NO case here is immediate. For the YES case, there is a value assignment $\mathbf{z}$ that makes $1-\eta$ fraction of constraints satisfied with K-XOR predicate. Since we flipped both the clauses and the labels accordingly, $\mathbf{z}$ still makes $1-\eta$ fraction of constraints satisfied with K-XOR predicate in the new instance. For each clause $C' = (l_1, \cdots, l_K)$, $l_i \sim_U \{\pm 1\}$ for all i, thus, $\mathbf{Pr}[l_1 + \cdots + l_k \notin [-2t, 2t]] = \exp(-\Omega(t^2/K))$. Noting that $l_1 + \cdots + l_k \in [-2t, 2t]$ implies $k\text{−XOR}(l_1, \cdots, l_K) = (k,t)\text{−XOR}(l_1, \cdots, l_K)$. Therefore, $\mathbf{z}$ makes $1 - \eta - \exp(-\Omega(t^2/K))$ fraction of constraints satisfied with (K,t)-XOR predicate in the new instance. □

Then we approximate the (K,t)-XOR functions with polynomials.

Lemma C.8. *Let f be a fast convergent activation function. Given Assumption C.5, there exists a constant $b > 0$ such that, for $\eta = 2^{-\log^{0.5} n}$, $0 < t < K = \log n$ and $c_- : \mathbb{N} \to (-\infty, c_+)$, with $m = O(n^{bK})$, if we are given an input of m $(\mathbf{x}, y) \in \{-1, 0, +1\}^{Kn} \times \{f_{\text{lim}}, f(c_+)\}$ tuples, then the following two cases cannot be distinguished in $\text{poly}(n, m)$ time:*

- *(YES Case) There exists a degree-$O(t)$ polynomial p satisfying the following properties.*
  - *(i) $\|p\|_1 \le O(c_-(n)) \cdot (Kn)^{O(t)}$, and,*
  - *(ii) $1 - \eta - \exp(-\Omega(t^2/K))$ fraction of tuples satisfy $|f(p(\mathbf{x})) - y| \le f_{\text{gap}}(c_-(n))$ and the remaining $\eta + \exp(-\Omega(t^2/K))$ fraction satisfies $|f(p(\mathbf{x})) - y| \le f_{\text{gap}}(c_-(n)) + |f_{\text{lim}} - f(c_+)|$.*
- *(NO Case) $y \sim_U \{f_{\text{lim}}, f(c_+)\}$ independently of $\mathbf{x}$.*

Proof. We reduce from the problem in Lemma C.7. The proof is the same as the proof of 4.5. The only difference is that we uses Lemma A.1 to approximate (K,t)-XOR instead of using Lemma 2.2 to approximate $L\text{-MAJ}_{(K,t)\text{-XOR}}$. □

C.2.2 Step III: From Polynomial to Learning a Neuron

Considering a polynomial is linear in the Veronese mapping of the inputs, we have Theorem C.9.

Theorem C.9. *Let f be a fast convergent activation function. Given Assumption C.5, there exists a constant $\nu \in (0, 1/2)$ such that if we are given an input of $\text{poly}(N)$ $(\mathbf{x}, y) \in \{-1, 0, +1\}^N \times \{f_{\text{lim}}, f(c_+)\}$ tuples, then the following two cases cannot be distinguished in $\text{poly}(N)$ time:*

- *(YES Case) There exists $\mathbf{w}$ such that*
  - *(i) $\|\mathbf{w}\|_1 \le N^{O(1)}$, and,*
  - *(ii) $\mathbf{E}[(y - f(\langle \mathbf{w}, \mathbf{x} \rangle))^2] \le \exp(-\Omega(\log^\nu n))$.*
- *(NO Case) $y \sim_U \{f_{\text{lim}}, f(c_+)\}$ independently of $\mathbf{x}$.*

Proof. We reduce from the problem in Lemma C.8. Suppose we have a $\text{poly}(N)$-time algorithm A for solving the problem in Theorem C.9 with $m = N^u$ tuples. Let $d = O(t)$ be p's degree in Lemma C.8 and $N = (Kn)^{O(d)}$ be the dimension after applying degree-d Veronese mapping. We let $c_-(n) = -N$ and $t = \log^b n$ in Lemma C.8 and select universal constant b such that $b < 1$ and $\exp(-\Omega(t^2/K)) = \exp(-\Omega(\log^\nu n))$, i.e., $\frac{2b-1}{b+1} > \nu$.

Given an instance of the problem in Lemma C.8, we apply the Veronese mapping of degree-d on $\mathbf{x}$. Note that A runs in $\text{poly}(N) = \text{poly}((Kn)^t) = \text{poly}(n^{bK}) = \text{poly}(m)$ time. Furthermore, there are $m = n^{bK} = N^{bK/t} = N^{\omega(1)}$ many samples which are sufficient for A.

We claim that the two cases in Lemma C.8 satisfy the conditions in the corresponding cases here. For both cases, there are at least N^u tuples. The NO case here is immediate. For the YES case, $\|\mathbf{w}\|_1 = \|p\|_1 \le O(c_-(n)) \cdot (Kn)^{O(t)} \le O(N) \cdot (Kn)^{O(t)} \le N^{O(1)}$, since $N = (Kn)^{O(t)}$. For the expected error, we can write

$\mathbf{E}[(y - h(\mathbf{x}))^2] = (\eta + \exp(-\Omega(t^2/K))) \cdot (|f_{\lim} - f(c_+)| + f_{\mathrm{gap}}(-N))^2 + (1 - \eta - \exp(-\Omega(t^2/K))) \cdot f_{\mathrm{gap}}^2(-N) \leq \exp(-\Omega(\log^\nu N))$. This completes the proof. □

Theorem C.6 then follows from Theorem C.9.

Proof of Theorem C.6. Suppose we have an algorithm A that is an α-approximate agnostic learner and runs in time $O(N^t)$. Then, we claim that we can solve the problem in Theorem C.9 for $\nu > \nu'$ and $m = N^{t+1}$. The inputs distribution for A will be the uniform distribution on the set of m tuples in Theorem C.9 with $\epsilon = \frac{(f_{\lim} - f(c_+))^2}{16}$.

In the YES case, w.h.p. A will return a hypothesis of squared error at most $\alpha \cdot \exp(-\Omega(\log^\nu N)) + \epsilon \leq \frac{(f_{\lim} - f(c_+))^2}{8}$ for N greater than a sufficiently large constant. In the NO case, no hypothesis can have a nontrivial advantage over the constant hypothesis $h(\mathbf{x}) = \frac{f_{\lim} + f(c_+)}{2}$ on the support it has not seen. Since A can see at most $O(N^t)$ samples among all the N^{t+1} samples, with sufficiently large N, A cannot return a hypothesis with error smaller than $\frac{(f_{\lim} - f(c_+))^2}{6}$. Thus, A will return a hypothesis of squared error at least $\frac{(f_{\lim} - f(c_+))^2}{6}$. Therefore, we can distinguish the two cases in Theorem C.9 for $\nu > \nu'$ and $m = N^{t+1}$ with algorithm A. This contradicts Assumption C.5. □