Proceedings of Machine Learning Research vol 65:1–39, 2017

Reliably Learning the ReLU in Polynomial Time

Surbhi Goel SURBHI@CS.UTEXAS.EDU
University of Texas at Austin

Varun Kanade VARUNK@CS.OX.AC.UK
University of Oxford and Alan Turing Institute

Adam Klivans KLIVANS@CS.UTEXAS.EDU
University of Texas at Austin

Justin Thaler JUSTIN.THALER@GEORGETOWN.EDU
Georgetown University

Abstract

We give the first dimension-efficient algorithms for learning Rectified Linear Units (ReLUs), which are functions of the form $\mathbf{x} \mapsto \max(0, \mathbf{w} \cdot \mathbf{x})$ with $\mathbf{w} \in \mathbb{S}^{n-1}$. Our algorithm works in the challenging Reliable Agnostic learning model of Kalai et al. (2012) where the learner is given access to a distribution $\mathcal{D}$ on labeled examples but the labeling may be arbitrary. We construct a hypothesis that simultaneously minimizes the false-positive rate and the loss on inputs given positive labels by $\mathcal{D}$, for any convex, bounded, and Lipschitz loss function.

The algorithm runs in polynomial-time (in n) with respect to *any* distribution on $\mathbb{S}^{n-1}$ (the unit sphere in n dimensions) and for any error parameter $\epsilon = \Omega(1/\log n)$ (this yields a PTAS for a question raised by F. Bach on the complexity of maximizing ReLUs). These results are in contrast to known efficient algorithms for reliably learning linear threshold functions, where ϵ must be $\Omega(1)$ and strong assumptions are required on the marginal distribution.

We can compose our results to obtain the first set of efficient algorithms for learning constant-depth networks of ReLU with fixed polynomial-dependence in the dimension. For depth-2 networks of sigmoids, we obtain the first algorithms that have a polynomial dependency in *all parameters*.

Our techniques combine kernel methods and polynomial approximations with a "dual-loss" approach to convex programming. As a byproduct we obtain a number of applications including the first set of efficient algorithms for "convex piecewise-linear fitting" and the first efficient algorithms for noisy polynomial reconstruction of low-weight polynomials on the unit sphere.

Keywords: ReLU, agnostic learning, reliable, kernel methods

1. Introduction

Let $\mathcal{X} = \mathbb{S}^{n-1}$, the set of all unit vectors in $\mathbb{R}^n$, and let $\mathcal{Y} = [0, 1]$. We define a ReLU (Rectified Linear Unit) to be a function $f(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$ equal to $\max(0, \mathbf{w} \cdot \mathbf{x})$ where $\mathbf{w} \in \mathbb{S}^{n-1}$ is a fixed element of $\mathbb{S}^{n-1}$ and $\mathbf{w} \cdot \mathbf{x}$ denotes the standard inner product.[1] The ReLU is a key building block in the area of deep nets, where the goal is to construct a network or circuit of ReLUs that "fits" a training set with respect to various measures of loss. Recently, the ReLU has become the

[1] Throughout this manuscript, bold lower case variables denote vectors. Unbolded lower case variables denote real numbers.

"activation function of choice" for practitioners in deep nets, as it leads to striking performance in various applications (LeCun et al., 2015).

There has been much recent work on understanding the computational complexity of deep neural networks, e.g., Livni et al. (2014); Sedghi and Anandkumar (2014); Janzamin et al. (2015); Zhang et al. (2015, 2016a,b), but some of the simplest issues regarding computational complexity remain open.

One point of this paper is to note that the computational complexity of learning *even a single ReLU* is still open! Our main result is that, *without making any distributional assumptions*, a ReLU can be learned in *fixed polynomial-time in the dimension*, regardless of the error parameter ϵ. The main drawback of our result for learning a ReLU is that in terms of the accuracy parameter, we only obtain a PTAS, as the dependence on ϵ is $2^{O(1/\epsilon)}$. In Section 5, we give a simple reduction showing that learning a single ReLU with respect to distributions on $\{0,1\}^n$ is as hard as learning sparse parities with noise. Subsequent to the posting of our work, Bartlett et al. (2017) observed that this reduction also rules out polynomial-time algorithms for agnostically learning a ReLU with respect to distributions on $\mathbb{S}^{n-1}$ when $\epsilon = 1/\mathsf{poly}(n)$ by scaling down from the Boolean cube to the unit sphere. The exact range of ϵ for which a polynomial-time algorithm is achievable remains open.

In more detail, we provide the first set of efficient algorithms for learning a ReLU. The algorithms succeed with respect to *any* distribution $\mathcal{D}$ on $\mathbb{S}^{n-1}$, tolerate arbitrary labelings (equivalently viewed as adversarial noise, and often referred to as the "non-realizable" setting), and run in polynomial-time for any error parameter $\epsilon = \Omega(1/\log n)$. Further, our algorithms achieve *reliability*, a natural notion for ReLU learning that we describe in Section 1.1. This is in contrast to the problem of learning threshold functions, i.e., functions of the form $\mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$, where only computational hardness results are known (unless stronger assumptions are made on the problem).

To put our results in further context, recall the following two fundamental machine-learning problems:

Problem 1 (Ordinary Least Squares Regression) *Let $\mathcal{D}$ be a distribution on $\mathbb{S}^{n-1} \times [0,1]$. Given i.i.d. examples drawn from $\mathcal{D}$, find $\mathbf{w} \in \mathbb{S}^{n-1}$ that minimizes $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(\mathbf{w} \cdot \mathbf{x} - y)^2]$.*

Problem 2 (Agnostically Learning a Threshold Function) *Let $\mathcal{D}$ be a distribution on $\mathbb{S}^{n-1} \times \{-1,1\}$. Given i.i.d. examples drawn from $\mathcal{D}$, find $\mathbf{w} \in \mathbb{S}^{n-1}$ that approximately minimizes $\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\mathsf{sign}(\mathbf{w} \cdot \mathbf{x}) \neq y]$.*

The term *agnostic* above refers to the fact that the labeling on $\{-1,1\}$ may be *arbitrary*. In this work, we relax the notion of success to *improper learning*, where the learner may output any polynomial-time computable hypothesis achieving a loss that is within ϵ of the optimal solution from the concept class.

Taken together, these two problems are at the core of many important techniques from modern Machine Learning and Statistics. It is well-known how to efficiently solve ordinary least squares and other variants of linear regression; we know of multiple polynomial-time solutions, all extensively used in practice (Rigollet, 2015). In contrast, Problem 2 is thought to be computationally intractable due to the many existing hardness results in the literature (Daniely, 2016; Kalai et al., 2008; Klivans and Sherstov, 2009; Feldman et al., 2009).

The ReLU is a hybrid function that lies "in-between" a linear function and a threshold function in the following sense: restricted to inputs $\mathbf{x}$ such that $\mathbf{w} \cdot \mathbf{x} > 0$, the ReLU is linear, and for inputs $\mathbf{x}$ such that $\mathbf{w} \cdot \mathbf{x} \leq 0$, the ReLU thresholds the value $\mathbf{w} \cdot \mathbf{x}$ and simply outputs zero. In this sense,

we could view the ReLU as a "one-sided" threshold function. Since learning a ReLU has aspects of both linear regression and threshold learning, it is not straightforward to identify a notion of loss that captures both of these aspects.

1.1. Reliably Learning Real-Valued Functions

We introduce a natural model for learning ReLUs inspired by the Reliable Agnostic learning model that was introduced by Kalai et al. (2012) in the context of Boolean functions. The goal will be to minimize both the false positive rate and a loss function (for example, square-loss) on points the distribution labels non-zero. In this work, we give efficient algorithms for learning a ReLU over the unit sphere with respect to any loss function that satisfies mild properties (convexity, monotonicity, boundedness, and Lipschitz-ness).

The Reliable Agnostic model is motivated by the Neyman-Pearson criteria, and is intended to capture settings in which false positive errors are more costly than false negative errors (e.g., spam detection) or vice versa. We observe that the asymmetric manner in which the Reliable Agnostic model (Kalai et al., 2012) treats different types of errors naturally corresponds to the one-sided nature of a ReLU. In particular, there may be settings in which mistakenly predicting a positive value instead of zero carries a high cost.

As a concrete example, imagine that inputs are comments on an online news article. Suppose that each comment is assigned a numerical score of quality or appropriateness, where the true scoring function is reasonably modeled by a linear function of the features of the comment. The newspaper wants to implement an automated system in which comments are either a) rejected outright if the score is below a threshold or b) posted in order of score, possibly after undergoing human review.[2] In this situation, it may be costlier to post (or subject to human review) a low-quality or inappropriate comment than it is to automatically reject a comment that is slightly above the threshold for posting.

More formally, for a function h and distribution $\mathcal{D}$ over $\mathbb{R}^n \times [0,1]$ define the following losses

$$\mathcal{L}_{=0}(h;\mathcal{D}) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq 0 \wedge\ y = 0]$$
$$\mathcal{L}_{>0}(h;\mathcal{D}) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h(\mathbf{x}),y)\cdot\mathbb{I}(y>0)].$$

Here, ℓ is a desired loss function, and $\mathbb{I}(y>0)$ equals 0 if $y \leq 0$ and 1 otherwise. These two quantities are respectively the false-positive rate and the expected loss (under ℓ) on examples for which the true label y is positive.[3]

Let $\mathcal{C}$ be a class of functions mapping $\mathbb{S}^{n-1}$ to $[0,1]$ (e.g., $\mathcal{C}$ may be the class of all ReLUs). Let $\mathcal{C}^+ = \{c \in \mathcal{C} \mid \mathcal{L}_{=0}(c;D) = 0\}$. We say $\mathcal{C}$ is *reliably learnable* if there exists a learning algorithm $\mathcal{A}$ that (with high probability) outputs a hypothesis that 1) has at most ϵ false positive rate and 2) on points with positive labels, has expected loss that is within ϵ of the best c from $\mathcal{C}^+$. That is, the hypothesis must be both *reliable* and competitive with the optimal classifier from the class $\mathcal{C}^+$ (*agnostic*).

[2] For example, The New York Times recently announced that they are moving to a hybrid comment moderation system that combines human and automated review (Etim, 2016).

[3] We restrict $\mathcal{Y} = [0,1]$ as it is a natural setting for the case of ReLUs. However, our results can easily be extended to larger ranges.

1.2. Statements of Our Main Results

We can now state our main theorem giving a poly-time algorithm (in n, the dimension) for reliably learning any ReLU.

All of our results hold for loss functions ℓ that satisfy convexity, monotonicity, boundedness, and Lipschitz-ness. For brevity, we avoid making these requirements explicit in the theorem statements of this introduction, and we omit the dependence of the runtime on the failure probability δ of the algorithm or on the boundedness and Lipschitz parameters of the loss function. All theorem statements in subsequent sections do state explicitly to what class of loss functions they apply, as well as the runtime dependence on these additional parameters.

Theorem 3 *Let $\mathcal{C} = \{\mathbf{x} \mapsto \max(0, \mathbf{w} \cdot \mathbf{x}) : \|\mathbf{w}\|_2 \leq 1\}$ be the class of ReLUs with weight vectors $\mathbf{w}$ satisfying $\|\mathbf{w}\|_2 \leq 1$. There exists a learning algorithm $\mathcal{A}$ that reliably learns $\mathcal{C}$ in time $2^{O(1/\epsilon)} \cdot n^{O(1)}$.*

Remark 4 *We can obtain the same complexity bounds for learning ReLUs in the standard agnostic model with respect to the same class of loss functions. This yields a PTAS (polynomial-time approximation scheme) for an optimization problem regarding ReLUs posed by Bach (2014). See Section 3.4 for details.*

For the problem of learning threshold functions, all known polynomial-time algorithms require strong assumptions on the marginal distribution (e.g., Gaussian (Kalai et al., 2008) or large-margin (Shalev-Shwartz et al., 2011)). In contrast, for ReLUs, we succeed with respect to *any* distribution on $\mathbb{S}^{n-1}$. We leave open the problem of improving the dependence of Theorem 3 on ϵ. We note that for the problem of learning threshold functions—even assuming the marginal distribution is Gaussian—the run-time complexity must be at least $n^{\Omega(\log 1/\epsilon)}$ under the widely believed assumption that learning sparse parities is hard (Klivans and Kothari, 2014). Further, the best *known* algorithms for agnostically learning threshold functions with respect to Gaussians run in time $n^{O(1/\epsilon^2)}$ (Kalai et al., 2008; Diakonikolas et al., 2010). Contrast this to our result for learning ReLUs, where we give polynomial-time algorithms even for ϵ as small as $1/\log n$.

We can compose our results to obtain efficient algorithms for small-depth networks of ReLUs. For brevity, here we state results only for linear combinations of ReLUs (which are often called *depth-two* networks of ReLUs, see, e.g., Eldan and Shamir (2016)). Formal results for other types of networks can be found in Section 4.

Theorem 5 *Let $\mathcal{C}$ be a depth-2 network of ReLUs with k hidden units. Then $\mathcal{C}$ is reliably learnable in time $2^{O(\sqrt{k}/\epsilon)} \cdot n^{O(1)}$.* [4]

The above results are perhaps surprising in light of the hardness result due to Livni et al. (2014) who showed that for $\mathcal{X} = \{0,1\}^n$, learning the difference of even two ReLUs is as hard as learning a threshold function.

Our results also extend to networks of sigmoids where we achieve *polynomial dependence on all parameters, including ϵ*. Livni et al. (2014) state an incomparable result for the same networks but with superpolynomial runtime in n (additionally their setting is the Boolean cube instead of the unit sphere).

[4] A recent manuscript due to Arora et al. (2016) considers the complexity of training depth-2 networks of ReLUs with k hidden units on a sample of size m. They give a proper learning algorithm that runs in time $2^k m^{nk} \mathsf{poly}(m, n, k)$.

Theorem 6 *Let $\mathcal{C}$ be a depth-2 network of sigmoids with k hidden units. Then $\mathcal{C}$ is reliably learnable in time $O(\sqrt{k}/\epsilon) \cdot n^{O(1)}$.*

We also obtain results for *noisy polynomial reconstruction* on the sphere (equivalently, agnostically learning a polynomial) with respect to a large class of loss functions:

Theorem 7 *Let $\mathcal{C}$ be the class of polynomials $p\colon \mathbb{S}^{n-1} \to [-1,1]$ in n variables such that the total degree of p is at most d, and the sum of squares of coefficients of p (in the standard monomial basis) is at most B. Then $\mathcal{C}$ is agnostically learnable under any (unknown) distribution over $\mathbb{S}^{n-1} \times [-1,1]$ in time $\mathsf{poly}(n, d, B, 1/\epsilon)$.*

Andoni et al. (2014) were the first to give efficient algorithms for noisy polynomial reconstruction over non-Boolean domains. In particular, they gave algorithms that succeed on the unit cube but require an underlying product distribution and do not work in the agnostic setting (they also run in time exponential in the degree d).

At a high level the proofs of both Theorem 3 and 7 follow the same outline, but we do not know how to obtain one from the other.

1.3. Applications to Convex Piecewise Regression

We establish a novel connection between learning networks of ReLUs and a broad class of piecewise-linear regression problems studied in machine learning and optimization. The following problem was defined by Magnani and Boyd (2009) as a generalization of the well-known MARS (multivariate adaptive regression splines) framework due to Friedman (1991):

Problem 8 (Convex Piecewise-Linear Regression: Max k-Affine) *Let $\mathcal{C}$ be the class of functions of the form $f(x) = \max(\mathbf{w}_1 \cdot \mathbf{x}, \ldots, \mathbf{w}_k \cdot \mathbf{x})$ with $\mathbf{w}_1, \ldots, \mathbf{w}_k \in \mathbb{S}^{n-1}$ mapping $\mathbb{S}^{n-1}$ to $\mathbb{R}$. Let $\mathcal{D}$ be an (unknown) distribution on $\mathbb{S}^{n-1} \times [-1,1]$. Given i.i.d. examples drawn from $\mathcal{D}$, output h such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(h(\mathbf{x}) - y)^2] \leq \min_{c\in\mathcal{C}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(c(\mathbf{x}) - y)^2] + \epsilon$.*

Applying our learnability results for networks of ReLUs, we obtain the first polynomial-time algorithms for solving the above *max-k-affine* regression problem and the *sum of max-2-affine* regression problem when $k = O(1)$. Boyd and Magnani specifically highlight the case of $k = O(1)$ and provide a variety of heuristics; we obtain the first provably efficient results.

Theorem 9 *There is an algorithm $\mathcal{A}$ for solving the convex piecewise-linear fitting problem (cf. Definition 8) in time $2^{O\left((k/\epsilon)^{\log k}\right)} \cdot n^{O(1)}$.*

We can also use our results for learning networks of ReLUs to learn the so-called "leaky ReLUs" and "parameterized" ReLUs (PReLUs); see Section 4.3 for details. We obtain these results by composing various "ReLU gadgets," i.e., constant-depth networks of ReLUs with a small number of bounded-weight hidden units.

1.4. Hardness

We also prove the first hardness results for learning a *single* ReLU via simple reductions to the problem of learning sparse parities with noise. These results highlight the difference between learning Boolean and real-valued functions and justify our focus on (1) input distributions over $\mathbb{S}^{n-1}$ and (2) learning problems that are *not* scale invariant (for example, learning a linear threshold function over the Boolean domain is equivalent to learning over $\mathbb{S}^{n-1}$ in the distribution-free setting).

Theorem 10 *Let $\mathcal{C}$ be the class of ReLUs over the domain $\mathcal{X} = \{0,1\}^n$. Then any algorithm for reliably learning $\mathcal{C}$ in time $g(\epsilon) \cdot \mathsf{poly}(n)$ for any function g will give a polynomial time algorithm for learning $\omega(1)$-sparse parities with noise (for any $\epsilon = O(1)$).*

Efficiently learning sparse parities (of any superconstant length) with noise is considered one the most challenging problems in theoretical computer science.

1.5. Our Technical Contribution: Achieving Reliability

We stress that the main technical contribution of this work is *achieving reliability* by combining ideas from convex programming and Rademacher complexity with kernel methods. We give a high level outline of this approach below.

Although kernel methods have been used in prior work to learn halfspaces under distributional assumptions (Kalai et al. (2008); Shalev-Shwartz et al. (2011)), the applicability of kernel methods to deep-learning architectures is still not well understood (certainly in contrast to the multitude of papers on the performance of gradient descent e.g., Kawaguchi (2016)). This is in large part because of the wide range of kernel functions/feature mappings to choose from (for example, our choice of the multinomial kernel is crucial to obtain our results on noisy polynomial reconstruction).

As far as we know, a straightforward application of kernel methods with a cleverly chosen feature map could learn ReLUs in polynomial-time in all the parameters.

Now we describe a high-level overview of our proof. Let $\mathcal{C}$ be the class of all ReLUs, and let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be a training set of examples drawn i.i.d. from some arbitrary distribution $\mathcal{D}$ on $\mathbb{S}^{n-1} \times [-1, 1]$. To obtain our main result for reliably learning a single ReLU (cf. Theorem 3), our starting point is Optimization Problem 1 below.

Optimization Problem 1

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \sum_{i:y_i>0} \ell(y_i, \mathsf{max}(0, \mathbf{w}\cdot\mathbf{x}_i)) \\ \text{subject to} \quad & \mathsf{max}(0, \mathbf{w}\cdot\mathbf{x}_i) = 0 \quad \text{for all } i \text{ such that } y_i = 0 \\ & \|\mathbf{w}\|_2 \le 1 \end{aligned}$$

In Optimization Problem 1, ℓ denotes the loss function used to define $\mathcal{L}_{>0}$. Using standard generalization error arguments, it is possible to show that (for reasonable choices of ℓ) if $\mathbf{w}$ is an optimal solution to Optimization Problem 1 when run on a polynomial size sample $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ drawn from $\mathcal{D}$, then it is sufficient to output the hypothesis $h(\mathbf{x}) := \mathsf{max}(0, \mathbf{w}\cdot\mathbf{x})$. Unfortunately Optimization Problem 1 is not convex in $\mathbf{w}$, and hence it may not be possible to find an optimal

solution in polynomial time. Instead, we will give an efficient approximate solution that will suffice for reliable learning.

Our starting point will be to prove the existence of low-degree, low-weight polynomial approximators for every $c \in \mathcal{C}$. The polynomial method has a well established history in computational learning theory (e.g., Kalai et al. (2008) for agnostically learning halfspaces under distributional assumptions), and we can apply classical techniques from approximation theory and recent work due to Sherstov (2012) to construct low-weight, low-degree approximators for any ReLU.

We can then relax Optimization Problem 1 to the space of low-weight polynomials and follow the approach of Shalev-Shwartz et al. (2011) who used tools from Reproducing Kernel Hilbert Spaces (RKHS) to learn low-weight polynomials efficiently (Shalev-Shwartz et al. focused on a relaxation of the 0/1 loss for halfspaces).

The main challenge is to obtain reliability; i.e., to simultaneously minimize the false-positive rate and the loss dictated by the objective function. To do this we take a "dual-loss" approach and carefully construct two loss functions that will both be minimized with high probability. Proving that these losses generalize for a large class of objective functions is subtle and requires "clipping" in order to apply the appropriate Rademacher bound. Our final output hypothesis is $\max(0, h)$ where h is a "clipped" version of the optimal low-weight, low-degree polynomial on the training data, appropriately kernelized.

Our learning algorithms for networks of ReLUs are obtained by generalizing a composition technique due to Zhang et al. (2016a), who considered networks of "smooth" activation functions computed by power series (we discuss this in greater detail in 1.6 below, and in Section 4). Using a sequence of "gadget" reductions, we then show that even small-size networks of ReLUs are surprisingly powerful, yielding the first set of provably efficient algorithms for a variety of piecewise-linear regression problems in high dimension.

1.6. Comparison of Our Algorithms to Shalev-Shwartz et al. (2011) and Zhang et al. (2016a)

While we build on the algorithmic techniques of Shalev-Shwartz et al. (2011) and Zhang et al. (2016a), our algorithms depart from prior work in the following manner.

Zhang et al. (2016a) give algorithms for learning neural networks via composition of kernels. More precisely, they observe that it is possible to compose the kernel used in Shalev-Shwartz et al. (2011) to obtain results for neural networks where the activation functions are exactly computed by a power series with bounded coefficients. Because the ReLU is not differentiable at 0, and thus not computed by a power series, Zhang et al. (2016a) prove learning results for "ReLU-like" activations and not ReLUs. It is not clear whether there is a formal relationship between ReLU-like activations and actual ReLUs.

In contrast to exact computation by power series, our algorithms use the notion of *approximation* by low-weight, *low-degree* polynomials. This subtle but important difference enables us to use tools from approximation theory to obtain results for the actual ReLU function, even though it is not smooth. In addition, for the case of depth-2 networks of sigmoids, this underlies our ability to obtain learning algorithms with runtime that is polynomial in the number of hidden units (cf. Corollary 37), rather than exponential as in Zhang et al. (2016a).

Moreover, the kernel used by Shalev-Shwartz et al. (2011) and Zhang et al. (2016a) will provably give exponentially worse bounds (in the degree) for the results on noisy polynomial reconstruc-

tion that we obtain (cf. Theorem 7). The technical reason that this statement is true is discussed in Remark 18.

2. Preliminaries

2.1. Notation

The input space is denoted by $\mathcal{X}$ and the output space by $\mathcal{Y}$. In most of this paper, we consider settings in which $\mathcal{X} = \mathbb{S}^{n-1}$, the unit sphere in $\mathbb{R}^n$,[5] and $\mathcal{Y}$ is either $[0,1]$ or $[-1,1]$. Let $\mathcal{B}_n(0,r)$ denote the origin centered ball of radius r in $\mathbb{R}^n$.

We denote vectors by boldface lowercase letters such as $\mathbf{w}$ or $\mathbf{x}$, and $\mathbf{w} \cdot \mathbf{x}$ denotes the standard scalar (dot) product. By $\|\mathbf{w}\|$ we denote the standard ℓ_2 (i.e., Euclidean) norm of the vector $\mathbf{w}$; when necessary we will use subscripts to indicate other norms. If $f\colon \mathbb{S}^{n-1} \to \mathbb{R}$ is a real-valued function over the unit sphere, we say that a multivariate polynomial p is an ϵ-approximation to f if $|p(\mathbf{x}) - f(\mathbf{x})| \leq \epsilon$ for all $\mathbf{x} \in \mathbb{S}^{n-1}$. For a natural number $n \in \mathbb{N}$, $[n] = \{0, 1, \ldots, n\}$.

2.2. Concept Classes

Neural networks are composed of units—each unit has some $\mathbf{x} \in \mathbb{R}^n$ as input (for some value of n, and $\mathbf{x}$ may consist of outputs of other units) and the output is typically a linear function composed with a non-linear *activation* function, i.e., the output of a unit is of the form $f(\mathbf{w}\cdot\mathbf{x})$, where $\mathbf{w} \in \mathbb{R}^n$ and $f : \mathbb{R} \to \mathbb{R}$.

Definition 11 (Rectifier) *The rectifier (denoted by σ_{relu}) is an activation function defined as $\sigma_{\mathrm{relu}}(x) = \mathsf{max}(0, x)$.*

Definition 12 ($\mathrm{ReLU}(n, W)$) *For $\mathbf{w} \in \mathbb{R}^n$, let $\mathrm{relu}_{\mathbf{w}} : \mathbb{R}^n \to \mathbb{R}$ denote the function $\mathrm{relu}_{\mathbf{w}}(\mathbf{x}) = \mathsf{max}(0, \mathbf{w} \cdot \mathbf{x})$. Let $W \in \mathbb{R}^+$; we denote by $\mathrm{ReLU}(n, W)$ the class of rectified linear units defined by $\{\mathrm{relu}_{\mathbf{w}} \mid \mathbf{w} \in \mathcal{B}_n(0, W)\}$.*

Our results on reliable learning focus on the class $\mathrm{ReLU}(n, 1)$. We define networks of ReLUs in Section 4, where we also present results on agnostic learning and reliable learning of networks of ReLUs.

Definition 13 ($\mathcal{P}(n, d, B)$) *Let $B \in \mathbb{R}^+$, $n, d \in \mathbb{N}$. We denote by $\mathcal{P}(n, d, B)$ the class of n-variate polynomials p of total degree at most d such that the sum of the squares of the coefficients of p in the standard monomial basis is bounded by B.*

2.3. Learning Models

We consider two learning models in this paper. The first is the standard agnostic learning model (Kearns et al., 1994; Haussler, 1992) and the second is a generalization of the reliable agnostic learning framework (Kalai et al., 2012). We describe these models briefly; the reader may refer to the original articles for further details.

[5] All of our algorithms would also work under arbitrary distributions over the unit *ball*.

Definition 14 (Agnostic Learning (Kearns et al., 1994; Haussler, 1992)) *We say that a concept class $\mathcal{C} \subseteq \mathcal{Y}^{\mathcal{X}}$ is agnostically learnable with respect to loss function $\ell : \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}^+$ (where $\mathcal{Y} \subseteq \mathcal{Y}'$), if for every $\delta, \epsilon > 0$ there exists a learning algorithm $\mathcal{A}$ that for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ satisfies the following. Given access to examples drawn from $\mathcal{D}$, $\mathcal{A}$ outputs a hypothesis $h : \mathcal{X} \to \mathcal{Y}'$, such that with probability at least $1 - \delta$,*

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h(\mathbf{x}), y)] \leq \min_{c \in C} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(c(\mathbf{x}), y)] + \epsilon. \tag{1}$$

Furthermore, if $\mathcal{X} \subseteq \mathbb{R}^n$ and s is a parameter that captures the representation complexity (i.e., description length) of concepts c in $\mathcal{C}$, we say that $\mathcal{C}$ is efficiently agnostically learnable to error ϵ *if $\mathcal{A}$ can output an h satisfying Equation (1) with running time polynomial in n, s, and $1/\delta$.*[6]

Next, we formally describe our extension of the reliable agnostic learning model introduced by Kalai et al. (2012) to the setting of real-valued functions (see Section 1 for motivation). Suppose the data is distributed according to some distribution $\mathcal{D}$ over $\mathcal{X} \times [0, 1]$. For $\mathcal{Y}' \supseteq [0, 1]$, let $h : \mathcal{X} \to \mathcal{Y}'$ be some function and let $\ell : \mathcal{Y}' \times [0, 1] \to \mathbb{R}^+$ be a loss function. We define the following two *losses* for f with respect to the distribution $\mathcal{D}$:

$$\mathcal{L}_{=0}(h; \mathcal{D}) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq 0 \wedge y = 0] \tag{2}$$

$$\mathcal{L}_{>0}(h; \mathcal{D}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h(\mathbf{x}), y) \cdot \mathbb{I}(y > 0)], \tag{3}$$

where $\mathbb{I}(y > 0)$ is 1 if $y > 0$ and 0 otherwise. In words, $\mathcal{L}_{=0}$ considers the zero-one loss on points where the target y equals 0 and $\mathcal{L}_{>0}$ considers the loss (or risk) when $y > 0$. Both of these losses are defined with respect to the distribution $\mathcal{D}$, without conditioning on the events $y = 0$ or $y > 0$. This is necessary to make efficient learning possible—if the probability of the events $y = 0$ or $y > 0$ is too small, it is impossible for learning algorithms to make any meaningful predictions conditioned on those events.

Definition 15 (Reliable Agnostic Learning) *We say that a concept class $\mathcal{C} \subseteq [0, 1]^{\mathcal{X}}$ is reliably agnostically learnable (reliably learnable for short) with respect to loss function $\ell : \mathcal{Y}' \times [0, 1] \to \mathbb{R}^+$ (where $[0, 1] \subseteq \mathcal{Y}'$), if the following holds. For every $\delta, \epsilon > 0$, there exists a learning algorithm $\mathcal{A}$ such that, for every distribution $\mathcal{D}$ over $\mathcal{X} \times [0, 1]$, when $\mathcal{A}$ is given access to examples drawn from $\mathcal{D}$, $\mathcal{A}$ outputs a hypothesis $h : \mathcal{X} \to \mathcal{Y}'$, such that with probability at least $1 - \delta$, the following hold:*

$$\textit{(i)}\ \mathcal{L}_{=0}(h; \mathcal{D}) \leq \epsilon, \qquad \textit{(ii)}\ \mathcal{L}_{>0}(h; \mathcal{D}) \leq \min_{c \in \mathcal{C}^+(\mathcal{D})} \mathcal{L}_{>0}(c; \mathcal{D}) + \epsilon,$$

where $\mathcal{C}^+(\mathcal{D}) = \{c \in \mathcal{C} \mid \mathcal{L}_{=0}(c; \mathcal{D}) = 0\}$. Furthermore, if $\mathcal{X} \subseteq \mathbb{R}^n$ and s is a parameter that captures the representation complexity of concepts c in $\mathcal{C}$, we say that $\mathcal{C}$ is efficiently reliably agnostically learnable to error ϵ if $\mathcal{A}$ can output an h satisfying the above conditions with running time that is polynomial in n, s, and $1/\delta$.[6]

[6] The error parameter ϵ is purposely omitted from the definition of efficiency; in our results we will explicitly state the dependence on ϵ and for what ranges of ϵ the running time remains polynomial in the remaining parameters.

2.3.1. LOSS FUNCTIONS

We have defined agnostic and reliable learning in terms of general loss functions. Below we describe certain properties of loss functions that are required in order for our results to hold. Let $\mathcal{Y}$ denote the range of concepts from the concept class; this will typically be $[-1, 1]$ or $[0, 1]$. Let $\mathcal{Y}' \supseteq \mathcal{Y}$. We consider loss functions of the form, $\ell : \mathcal{Y}' \times \mathcal{Y} \to \mathbb{R}^+$ and define the following properties:

- We say that ℓ is *convex in its first argument* if for every $y \in \mathcal{Y}$ the function $\ell(\cdot, y)$ is convex.
- We say that ℓ is *monotone* if for every $y \in \mathcal{Y}$, if $y'' \leq y' \leq y$, then $\ell(y', y) \leq \ell(y'', y)$ and if $y \leq y' \leq y''$, $\ell(y', y) \leq \ell(y'', y)$. Note that this is weaker than requiring that $|y'-y| \leq |y''-y|$ implies $\ell(y', y) \leq \ell(y'', y)$. This latter condition is not satisfied by several commonly used loss functions, e.g., hinge loss.
- We say that ℓ is *b-bounded* on the interval $[u, v]$, if for every $y \in \mathcal{Y}$, $\ell(y', y) \leq b$ for $y' \in [u, v]$.
- We say that ℓ is *L-Lipschitz* in interval $[u, v]$, if for every $y \in \mathcal{Y}$, $\ell(\cdot, y)$ is L-Lipschitz in the interval $[u, v]$.

The results presented in this work hold for loss functions that are convex, monotone, bounded and Lipschitz continuous in some suitable interval. (Monotonicity is not strictly a requirement for our results, but the sample complexity bounds may be worse for non-monotone loss functions; we point this out when relevant.) These restrictions are quite mild, and virtually every loss function commonly considered in (convex approaches to) machine learning satisfy these conditions. For instance, when $\mathcal{Y} = \mathcal{Y}' = [0, 1]$, it is easy to see that any ℓ_p loss function is convex, monotone, bounded by 1 and p-Lipschitz for $p \geq 1$.

2.4. Kernel Methods

We make use of kernel methods in our learning algorithms. For completeness, we define kernels and a few important results concerning kernel methods. The reader may refer to Hofmann et al. (2008) (or any standard text) for further details.

Any function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel (Mercer, 1909). A kernel K is symmetric if $K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}), \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$; K is positive definite if $\forall n \in \mathbb{N}, \forall \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, the $n \times n$ matrix $\mathbf{K}$, where $\mathbf{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$, is positive semi-definite. For any positive definite kernel, there exists a Hilbert space $\mathcal{H}$ equipped with an inner product $\langle \cdot, \cdot \rangle$ and a function $\psi : \mathcal{X} \to \mathcal{H}$ such that $\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$. We refer to ψ as the *feature map* for K.

By convention, we will use $\cdot$ to denote the standard inner product in $\mathbb{R}^n$ and $\langle \cdot, \cdot \rangle$ for the inner product in a Hilbert Space $\mathcal{H}$. When $\mathcal{H} = \mathbb{R}^n$ for some finite n, we will use $\langle \cdot, \cdot \rangle$ and $\cdot$ interchangeably.

We will use the following variant of the polynomial kernel:

Definition 16 (Multinomial Kernel) *Define $\psi_d \colon \mathbb{R}^n \to \mathbb{R}^{N_d}$, where $N_d = 1+n+\cdots+n^d$, indexed by tuples $(k_1, \ldots, k_j) \in [n]^j$ for each $j \in \{0, 1, \ldots, d\}$, where the entry of $\psi_d(\mathbf{x})$ corresponding to tuple $(k_1, \ldots, k_j)$ equals $x_{k_1} \cdots x_{k_j}$. (When $j = 0$ we have an empty tuple and the corresponding entry is 1.) Define kernel MK_d via:*

$$\mathsf{MK}_d(\mathbf{x}, \mathbf{x}') = \langle \psi_d(\mathbf{x}), \psi_d(\mathbf{x}') \rangle = \sum_{j=0}^{d} (\mathbf{x} \cdot \mathbf{x}')^j.$$

Also define $\mathcal{H}_{\mathsf{MK}_d}$ to be the corresponding Reproducing Kernel Hilbert Space (RKHS).

Observe that MK_d is the sum of standard polynomial kernels (cf. Wikipedia (2016b)) of degree i for $i \in [d]$. However, the feature map conventionally used for a standard polynomial kernel has only $\binom{n+d}{d}$ entries and, under that definition involves coefficients of size as large as $d^{\Theta(d)}$. The feature map ψ_d used by MK_d avoids these coefficients by using N_d entries as defined above (that is, entries of $\psi_d(\mathbf{x})$ are indexed by *ordered* subsets of $[n]$, while entries of the standard feature map are indexed by *unordered* subsets of $[n]$.)

Let $q\colon \mathbb{R}^n \to \mathbb{R}$ be a multivariate polynomial of total degree d. We say that a vector $\mathbf{v} \in \mathcal{H}_{\mathsf{MK}_d}$ *represents* q if $q(\mathbf{x}) = \langle \mathbf{v}, \psi_d(\mathbf{x})\rangle$ for all $\mathbf{x} \in \mathbb{S}^{n-1}$. Note that although the feature map ψ_d is fixed, a polynomial q will have many representations $\mathbf{v}$ as a vector in $\mathcal{H}_{\mathsf{MK}_d}$. Furthermore, observe that the Euclidean norm, $\langle \mathbf{v}, \mathbf{v}\rangle$, of these representations may not be equal.

The following example will play an important role in our algorithms for learning ReLUs. Let $\mathbf{w} \in \mathbb{R}^n$ and let $p(t)$ be a univariate degree-d equal to $\sum_{i=0}^{d} \beta_i t^i$ be given. Define the multivariate polynomial $p_{\mathbf{w}}(\mathbf{x}) := p(\mathbf{w} \cdot \mathbf{x})$.

Consider the representation of $p_{\mathbf{w}}$ as an element of $\mathcal{H}_{\mathsf{MK}_d}$ defined as follows: the entry of index $(k_1, \ldots, k_j) \in [n]^j$ of the representation equals $\beta_j \cdot \prod_{i=1}^{j} w_{k_i}$ for $j \in [d]$. Abusing notation, we use $p_{\mathbf{w}}$ to denote both the multivariate polynomial and the vector in $\mathcal{H}_{\mathsf{MK}_d}$. The following lemma establishes that $p_{\mathbf{w}} \in \mathcal{H}_{\mathsf{MK}_d}$ is indeed a representation of the polynomial $p_{\mathbf{w}}$, and gives a bound on $\langle p_{\mathbf{w}}, p_{\mathbf{w}}\rangle$. The proof follows an analysis applied by Shalev-Shwartz et al. (2011, Lemma 2.4) to a different kernel (cf. Remark 18 below).

Lemma 17 *Let $p(t) = \sum_{i=0}^{d} \beta_i t^i$ be a given univariate polynomial with $\sum_{i=1}^{d} \beta_i^2 \leq B$. For $\mathbf{w}$ such that $\|\mathbf{w}\| \leq 1$, consider the polynomial $p_{\mathbf{w}}(\mathbf{x}) := p(\mathbf{w} \cdot \mathbf{x})$. Then $p_{\mathbf{w}}$ is represented by the vector $p_{\mathbf{w}} \in \mathcal{H}_{\mathsf{MK}_d}$ defined above. Moreover, $\langle p_{\mathbf{w}}, p_{\mathbf{w}}\rangle \leq B$.*

Proof To see that $p_{\mathbf{w}}(\mathbf{x}) = \langle p_{\mathbf{w}}, \psi_d(\mathbf{x})\rangle$ for all $\mathbf{x} \in \mathbb{R}^n$, observe that

$$\begin{aligned} p_{\mathbf{w}}(\mathbf{x}) = p(\mathbf{w} \cdot \mathbf{x}) &= \sum_{i=0}^{d} \beta_i \cdot (\mathbf{w} \cdot \mathbf{x})^i \\ &= \sum_{i=0}^{d} \sum_{(k_1,\ldots,k_i)\in[n]^i} \beta_i \cdot w_{k_1} \cdot \cdots \cdot w_{k_i} \cdot x_{k_1} \cdot \cdots \cdot x_{k_i} \\ &= \langle p_{\mathbf{w}}, \psi_d(\mathbf{x})\rangle. \end{aligned}$$

Furthermore, we can compute

$$\begin{aligned} \langle p_{\mathbf{w}}, p_{\mathbf{w}}\rangle &= \sum_{i=0}^{d} \sum_{(k_1,\ldots,k_i)\in[n]^i} \beta_i^2 \cdot w_{k_1}^2 \cdot \cdots \cdot w_{k_i}^2 \\ &= \sum_{i=0}^{d} \beta_i^2 \cdot \sum_{k_1\in[n]} w_{k_1}^2 \cdot \cdots \cdot \sum_{k_i\in[n]} w_{k_i}^2 \\ &= \sum_{i=0}^{d} \beta_i^2 \|\mathbf{w}\|_2^{2i} = \sum_{i=0}^{d} \beta_i^2 \leq B. \end{aligned}$$

■

Remark 18 *Shalev-Shwartz et al. (2011) proved a bound on the Euclidean norm of representations of polynomials of the form $p(\mathbf{w} \cdot \mathbf{x})$ in the RKHS corresponding to the kernel function $K(\mathbf{x}, \mathbf{y}) = \frac{1}{1-\frac{1}{2}\langle \mathbf{x}, \mathbf{y} \rangle}$. This allowed them to represent functions computed by power series, as opposed to polynomials of (finite) degree d. However, for degree d polynomials, the use of their kernel results in a Euclidean norm bound that is a factor of 2^d worse than what we obtain from Lemma 17. This difference is central to our results on noisy polynomial reconstruction in Section 3.5, where we address this issue in more technical detail.*

2.5. Generalization Bounds

We make use of the following standard generalization bound for hypothesis classes with small Rademacher complexity. Readers unfamiliar with Rademacher complexity may refer to the paper of Bartlett and Mendelson (2002).

Theorem 19 (Bartlett and Mendelson (2002)) *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and let $\ell : \mathcal{Y}' \times \mathcal{Y}$ (where $\mathcal{Y} \subseteq \mathcal{Y}' \subseteq \mathbb{R}$) be a b-bounded loss function that is L-Lispschitz in its first argument. Let $\mathcal{F} \subseteq (\mathcal{Y}')^{\mathcal{X}}$ and for any $f \in \mathcal{F}$, let $\mathcal{L}(f; \mathcal{D}) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$ and $\widehat{\mathcal{L}}(f; S) := \frac{1}{m}\sum_{i=1}^{m} \ell(f(\mathbf{x}_i), y_i)$, where $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \sim \mathcal{D}^m$. Then for any $\delta > 0$, with probability at least $1 - \delta$ (over the random sample draw for S), simultaneously for all $f \in \mathcal{F}$, the following is true:*

$$|\mathcal{L}(f; \mathcal{D}) - \widehat{\mathcal{L}}(f; S)| \leq 4 \cdot L \cdot \mathcal{R}_m(\mathcal{F}) + 2 \cdot b \cdot \sqrt{\frac{\log(1/\delta)}{2m}}$$

where $\mathcal{R}_m(\mathcal{F})$ is the Rademacher complexity of the function class $\mathcal{F}$.

We will combine the following two theorems with Theorem 19 above to bound the generalization error of our algorithms for agnostic and reliable learning.

Theorem 20 (Kakade et al. (2008)) *Let $\mathcal{X}$ be a subset of a Hilbert space equipped with inner product $\langle \cdot, \cdot \rangle$ such that for each $\mathbf{x} \in \mathcal{X}$, $\langle \mathbf{x}, \mathbf{x} \rangle \leq X^2$, and let $\mathcal{W} = \{\mathbf{x} \mapsto \langle \mathbf{x}, \mathbf{w} \rangle \mid \langle \mathbf{w}, \mathbf{w} \rangle \leq W^2\}$ be a class of linear functions. Then it holds that*

$$\mathcal{R}_m(\mathcal{W}) \leq X \cdot W \cdot \sqrt{\frac{1}{m}}.$$

The following result as stated appears in Bartlett and Mendelson (2002) but is originally attributed to Ledoux and Talagrand (1991).

Theorem 21 (Bartlett and Mendelson (2002); Ledoux and Talagrand (1991)) *Let $\psi : \mathbb{R} \to \mathbb{R}$ be Lipschitz with constant L_ψ and suppose that $\psi(0) = 0$. Let $\mathcal{Y} \subseteq \mathbb{R}$, and for a function $f \in \mathcal{Y}^{\mathcal{X}}$, let $\psi \circ f$ denote the standard composition of ψ and f. Finally, for $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$, let $\psi \circ \mathcal{F} = \{\psi \circ f : f \in \mathcal{F}\}$. It holds that $\mathcal{R}_m(\psi \circ \mathcal{F}) \leq 2 \cdot L_\psi \cdot \mathcal{R}_m(\mathcal{F})$.*

2.6. Approximation Theory

First, we show that the rectifier activation function $\sigma_{\mathrm{relu}}(x) = \max(0, x)$ can be ϵ-approximated using a polynomial of degree $O(1/\epsilon)$. This result follows using Jackson's theorem (see, e.g., Newman (1964)). For convenience in later proofs, we will require that in fact the polynomial also takes values in the range $[0, 1]$ on the interval $[-1, 1]$. Of course, this is achieved easily starting from the polynomial obtained from Jackson's theorem and applying elementary transformations.

Lemma 22 *Let $\sigma_{\mathrm{relu}}(x) = \max(0, x)$ and $\epsilon \in (0, 1)$. There exists a polynomial p of degree $O(1/\epsilon)$ such that for all $x \in [-1, 1]$, $|\sigma_{\mathrm{relu}}(x) - p(x)| \leq \epsilon$ and $p([-1, 1]) \subseteq [0, 1]$.*

Proof We can express $\sigma_{\mathrm{relu}}(x) = \max(0, x)$ as $\sigma_{\mathrm{relu}}(x) = (x + |x|)/2$. We know from Jackson's Theorem (Newman, 1964) that there exists a polynomial $\tilde{p}$ of degree $O(1/\epsilon)$ such that for all $x \in [-1, 1]$, $||x| - \tilde{p}(x)| \leq \frac{\epsilon}{2-\epsilon}$. Consider the polynomial $\bar{p}(x) = \frac{\tilde{p}(x)+x}{2}$, which satisfies for any $x \in [-1, 1]$,

$$|\sigma_{\mathrm{relu}}(x) - \bar{p}(x)| = \left|\frac{|x| + x}{2} - \frac{\tilde{p}(x) + x}{2}\right| = \left|\frac{|x| - \tilde{p}(x)}{2}\right| \leq \frac{\epsilon}{2(2-\epsilon)}.$$

Finally, let $p(x) = \frac{2-\epsilon}{2}(\bar{p}(x) - \frac{1}{2}) + \frac{1}{2}$. We have for $x \in [-1, 1]$,

$$|\sigma_{\mathrm{relu}}(x) - p(x)| = \frac{\epsilon}{2}|\sigma_{\mathrm{relu}}(x)| + \frac{2-\epsilon}{2}|\sigma_{\mathrm{relu}}(x) - \bar{p}(x)| + \frac{1}{2}\left|\frac{2-\epsilon}{2} - 1\right| \leq \epsilon.$$

Furthermore, it is clearly the case that $p([-1, 1]) \subseteq [0, 1]$. ■

We remark that a consequence of the linear relationship between $\sigma_{\mathrm{relu}}(x)$ and $|x|$ is that the degree given by Jackson's theorem is essentially the lowest possible (Newman, 1964). Lemma 22 asserts the existence of a (relatively) low-degree approximation p to the rectifier activation function σ_{relu}. We will also require a bound on the sum of the squares of the coefficients of p. Even though Lemma 22 is non-constructive, we are nonetheless able to obtain such a bound below via standard interpolation methods.

Lemma 23 *Let $p(t) = \sum_{i=0}^{d} \beta_i t^i$ be a univariate polynomial of degree d. Let M be such that $\max_{t\in[-1,1]} |p(t)| \leq M$. Then $\sum_{i=0}^{d} \beta_i^2 \leq (d+1) \cdot (4e)^{2d} \cdot M^2$.*

Proof Lemma 4.1 from Sherstov (2012) states that for any polynomial satisfying the conditions in the statement of the lemma, the following holds for all $i \in \{0, \ldots, d\}$:

$$|\beta_i| \leq (4e)^d \max_{j=0,\ldots,d} \left|p\left(\frac{j}{d}\right)\right|.$$

We then have that

$$\sum_{i=0}^{d} \beta_i^2 = \sum_{i=0}^{d} |\beta_i|^2 \leq (d+1) \cdot (4e)^{2d} \cdot M^2.$$

■

Theorem 24 *Let $\mathcal{C} = \mathrm{ReLU}(n, W)$ (for $W \geq 1$) and $\epsilon \in (0, 1)$. Let $\mathcal{X} = \mathbb{S}^{n-1}$. For $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, consider the kernel MK_d, with $\mathcal{H}_{\mathsf{MK}_d}$ and ψ_d the corresponding RKHS and feature map (cf. Definition 16). Then for every $\mathbf{w} \in \mathbb{R}^n$ with $\|\mathbf{w}\| \leq W$, there exists a multivariate polynomial $p_{\mathbf{w}}$ of degree at most $O(W/\epsilon)$, such that, for every $\mathbf{x} \in \mathbb{S}^{n-1}$, $|\mathrm{relu}_{\mathbf{w}}(\mathbf{x}) - p_{\mathbf{w}}(\mathbf{x})| \leq \epsilon$. Furthermore, $p_{\mathbf{w}}(\mathbb{S}^{n-1}) \subseteq [0, W]$ and $p_{\mathbf{w}}$ when viewed as a member of $\mathcal{H}_{\mathsf{MK}_d}$ as described in Section 2.4, satisfies $\langle p_{\mathbf{w}}, p_{\mathbf{w}} \rangle \leq W^2 \cdot 2^{O(W/\epsilon)}$.*

Proof Let p be the univariate polynomial of degree $d = O(W/\epsilon)$ given by Lemma 22 that satisfies $|p(x) - \sigma_{\mathrm{relu}}(x)| \leq \frac{\epsilon}{W}$ for $x \in [-1, 1]$. Let $p(x) = \sum_{i=0}^{d} \beta_i \cdot x^i$; then by Lemma 23, we have $\sum_{i=0}^{d} \beta_i^2 \leq (d+1) \cdot (4e)^{2d} = 2^{O(W/\epsilon)}$ (as $|p(x)| \leq 1$ for $x \in [-1, 1]$).

Let q be the univariate polynomial defined as $q(x) = W \cdot p(x/W)$ for $W > 1$. The degree of q is d, the same as that of p, and if α_i are the coefficients of q, we have $\sum_{i=0}^{d} \alpha_i^2 \leq W^2 \cdot \sum_{i=0}^{d} \beta_i^2 \leq W^2 \cdot 2^{O(W/\epsilon)} = 2^{O(W/\epsilon)}$ (since $W > 1$). Let $p_{\mathbf{w}}(\mathbf{x}) = q(\mathbf{w} \cdot \mathbf{x})$. Note that $|p_{\mathbf{w}}(\mathbf{x}) - \mathrm{relu}_{\mathbf{w}}(x)| = |W \cdot p(\mathbf{w} \cdot \mathbf{x}/W) - W \cdot \mathrm{relu}_{(\mathbf{w}/W)}(\mathbf{x})| \leq \epsilon$ and $p_{\mathbf{w}}(\mathbb{S}^{n-1}) \subseteq q([-1, 1]) \subseteq [0, W]$. Finally, by applying Lemma 17, we get that $\langle p_{\mathbf{w}}, p_{\mathbf{w}} \rangle \leq W^2 \cdot 2^{O(W/\epsilon)}$. ■

3. Reliably Learning the ReLU

In this section, we focus on the problem of reliably learning a single rectified linear unit with weight vectors of norm bounded by 1, i.e., the concept class $\mathrm{ReLU}(n, 1)$. Specifically, our goal is to prove Theorem 3 from Section 1.2. Below we describe the algorithm and then give a full proof of Theorem 3.

3.1. Overview of the Algorithm and Its Analysis

In order to reliably learn ReLUs, it would suffice to solve Optimization Problem 1 (see Section 1). This mathematical program, however, is not convex; hence, we consider a suitable convex relaxation.

The convex relaxation optimizes over polynomials of a suitable degree. Theorem 24 shows that any concept in $\mathrm{ReLU}(n, 1)$ can be uniformly approximated to error ϵ by a degree $O(1/\epsilon)$ polynomial. It will be more convenient to view this polynomial as an element of the RKHS $\mathcal{H}_{\mathsf{MK}_d}$ defined in Definition 16. Recall that the corresponding kernel is $\mathsf{MK}_d(\mathbf{x}, \mathbf{x}') = \sum_{i=0}^{d} (\mathbf{x} \cdot \mathbf{x}')^i$ and the feature map is denoted ψ_d. Thus, instead of minimizing over $\mathbf{w}$ directly as in Optimization Problem 1, Optimization Problem 2 (below) minimizes over $\mathbf{v} \in \mathcal{H}_{\mathsf{MK}_d}$ of suitably bounded norm. In particular, we know that for any $\mathbf{w}$, the corresponding polynomial $p_{\mathbf{w}}$ that ϵ-approximates $\max(0, \mathbf{w} \cdot \mathbf{x})$, when viewed as an element of $\mathcal{H}_{\mathsf{MK}_d}$, satisfies $\langle p_{\mathbf{w}}, p_{\mathbf{w}} \rangle \leq B = 2^{O(1/\epsilon)}$ (see Theorem 24). Recall that $\langle p_{\mathbf{w}}, \psi_d(\mathbf{x}) \rangle = p_{\mathbf{w}}(\mathbf{x})$. Thus, we have the following optimization problem:

Optimization Problem 2

$$\begin{aligned} \underset{\mathbf{v}\in\mathcal{H}_{\mathsf{MK}_d}}{\text{minimize}} \quad & \sum_{i:\, y_i>0} \ell(\langle \mathbf{v}, \psi_d(\mathbf{x}_i)\rangle, y_i) \\ \text{subject to} \quad & \langle \mathbf{v}, \psi_d(\mathbf{x}_i)\rangle \le \epsilon \quad \text{for all } i \text{ such that } y_i = 0 \\ & \langle \mathbf{v}, \mathbf{v}\rangle \le B \end{aligned}$$

Clearly, if $\mathbf{w}$ is a feasible solution to Optimization Problem 1, then the corresponding element $p_{\mathbf{w}} \in \mathcal{H}_{\mathsf{MK}_d}$ is a feasible solution to Optimization Problem 2. We consider the value of the program for the feasible solution $p_{\mathbf{w}}$. For every $\mathbf{x} \in \mathbb{S}^{n-1}$, $p_{\mathbf{w}}(\mathbf{x}) = \langle p_{\mathbf{w}}, \psi_d(\mathbf{x})\rangle \in [0,1]$. Assuming that the loss function ℓ is L-Lipschitz in its first argument in the interval $[0,1]$, we have

$$\left| \sum_{i:\, y_i>0} \ell(\mathrm{relu}_{\mathbf{w}}(\mathbf{x}), y_i) - \sum_{i:\, y_i>0} \ell(\langle p_{\mathbf{w}}, \psi_d(\mathbf{x})\rangle, y_i) \right| \le |\{i \mid y_i > 0\}| \cdot L \cdot \epsilon.$$

Thus, an optimal solution to Optimization Problem 2 achieves a loss on the training data that is within $|\{i \mid y_i > 0\}| \cdot L \cdot \epsilon$ of that achieved by the optimal solution to Optimization Problem 1.

While Optimization Problem 2 is convex, it is still not trivial to solve efficiently. For one, the RKHS $\mathcal{H}_{\mathsf{MK}_d}$ has dimension $n^{\Theta(d)}$. However, materializing such vectors explicitly requires $n^{\Theta(d)}$ time, and Theorem 3 promises a learning algorithm with runtime $2^{O(1/\epsilon)} \cdot n^{O(1)} \ll n^{O(d)}$. As in Shalev-Shwartz et al. (2011), we apply the Representer Theorem (see e.g., Cristianini and Shawe-Taylor (2000)), to guarantee that Optimization Problem 2 can be solved in time that is polynomial in the number of samples used.

The Representer Theorem states that for any vector $\mathbf{v}$, there exists a vector $\mathbf{v}_\alpha = \sum_{i=1}^m \alpha_i \psi_d(\mathbf{x}_i)$ for $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$ such that the loss function of Optimization Problem 2 subject to the constraint $\langle \mathbf{v}, \mathbf{v}\rangle \le B$ does not increase when $\mathbf{v}$ is replaced with $\mathbf{v}_\alpha$. Crucially, we may further constrain these vectors $\mathbf{v}_\alpha$ to obey the inequality $\langle \mathbf{v}_\alpha, \psi_d(\mathbf{x}_i)\rangle \le \epsilon$ for all i such that $y_i = 0$. Thus, Optimization Problem 2 can be reformulated in terms of the variable vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$. This mathematical program is described as Optimization Problem 3 below.

Optimization Problem 3

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{minimize}} \quad & \sum_{i:y_i>0} \ell\left(\sum_{j=1}^m \alpha_j \mathsf{MK}_d(\mathbf{x}_j, \mathbf{x}_i), y_i \right) \\ \text{subject to} \quad & \sum_{j=1}^m \alpha_j \cdot \mathsf{MK}_d(\mathbf{x}_j, \mathbf{x}_i) \le \epsilon \quad \text{for all } i \text{ such that } y_i = 0 \\ & \sum_{i,j=1}^m \alpha_i \cdot \alpha_j \cdot \mathsf{MK}_d(\mathbf{x}_i, \mathbf{x}_j) \le B \end{aligned}$$

Let $\mathbf{K}$ denote the $m \times m$ *Gram* matrix whose $(i,j)^{th}$ entry is $\mathsf{MK}_d(\mathbf{x}_i, \mathbf{x}_j)$. Using the notation $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$, the last constraint is equivalent to $\boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \leq B$. As $\mathsf{MK}_d \succeq 0$, this defines a convex subset of $\mathbb{R}^m$. The remaining constraints are linear in $\boldsymbol{\alpha}$ and whenever the loss function ℓ is convex in its first argument, the resulting program is convex. Thus, Optimization Problem 3 can be solved in time polynomial in m.

3.2. Description of the Output Hypothesis

Let $\boldsymbol{\alpha}^*$ denote an optimal solution to Optimization Problem 3 and let $f(\cdot) = \sum_{i=1}^m \alpha_i^* \mathsf{MK}_d(\mathbf{x}_i, \cdot)$. To obtain strong bounds on the generalization error of our hypothesis, our algorithm does not simply output f itself. The obstacle is that, although f (viewed as an element of $\mathcal{H}_{\mathsf{MK}_d}$) satisfies $\langle f, f \rangle \leq B$, the best bound we can obtain on $|f(\mathbf{x})| = |\langle f, \mathbf{x} \rangle|$ for $\mathbf{x} \in \mathbb{S}^{n-1}$ is $\sqrt{B}$ by the Cauchy-Schwartz inequality. Observe that for many commonly used loss functions, such as the squared loss, this may result in a very poor Lipschitz constant and bound on the loss function, when applied to f in the interval $[-\sqrt{B}, \sqrt{B}]$ (recall that the only bound we have is $B = 2^{O(1/\epsilon)}$). Hence, a direct application of standard generalization bounds (cf. Section 2.5) yields a very weak bound on the generalization error of f itself. For example, suppose $y \in \{0,1\}$ and consider the loss function $\ell(y', y) = \exp(-y'(2y-1)+1) - 1$ if $y'(2y-1) \leq 1$ and $\ell(y', y) = 0$ otherwise (this loss function is like the hinge loss, but the linear side is replaced by an exponential). The Lipschitz constant of ℓ on the interval $[-\sqrt{B}, \sqrt{B}]$ is exponentially large in B, which would lead to a sample complexity bound that is *doubly*-exponentially large in $1/\epsilon$.

To address this issue, we will "clip" the function to always output a value between $[0,1]$:

Definition 25 *Define the function* $\mathrm{clip}_{a,b} : \mathbb{R} \to [a,b]$ *as follows:* $\mathrm{clip}_{a,b}(x) = a$ *for* $x \leq a$, $\mathrm{clip}_{a,b}(x) = x$ *for* $a \leq x \leq b$ *and* $\mathrm{clip}_{a,b}(x) = b$ *for* $b \leq x$.

The hypothesis h output by our algorithm is as follows.

$$h(x) = \begin{cases} 0 & \text{if } \mathrm{clip}_{0,1}(f(x)) \leq 2 \cdot \epsilon \\ \mathrm{clip}_{0,1}(f(x)) & \text{otherwise.} \end{cases}$$

We use a fact due to Ledoux and Talagrand on the Rademacher complexity of composed function classes (Theorem 21) to bound the generalization error. Clipping comes at a small cost, in the sense that it forces us to require that the loss function be monotone. However, we can handle non-monotone losses if the output hypothesis is not clipped, albeit with sample complexity bounds that depend polynomially on the Lipschitz-constant and bound of the loss in the interval $[-\sqrt{B}, \sqrt{B}]$ as opposed to $[0,1]$.

3.3. Formal Version of Theorem 3 and Its Proof

The rest of this section is devoted to the proof of Theorem 3 (or, more precisely, its formal variant Theorem 26 below, which makes explicit the conditions on the loss function ℓ that are required for the theorem to hold). In particular, we show that whenever the sample size m is a sufficiently large polynomial in $2^{O(1/\epsilon)}$, n, and $\log(1/\delta)$, the hypothesis h output by the algorithm satisfies $\mathcal{L}_{=0}(h; \mathcal{D}) = O(\epsilon)$ and $\mathcal{L}_{>0}(h; \mathcal{D}) \leq \min_{c \in \mathcal{C}^+(\mathcal{D})} \mathcal{L}_{>0}(c) + O(\epsilon)$, where $\mathcal{C}^+(\mathcal{D}) = \{\mathrm{relu}_{\mathbf{w}} \in \mathrm{ReLU}(n,1) \mid \mathcal{L}_{=0}(\mathrm{relu}_{\mathbf{w}}; \mathcal{D}) = 0\}$. Rescaling ϵ appropriately completes the proof of Theorem 26.

Theorem 26 (Formal Version of Theorem 3) *Let $\mathcal{X} = \mathbb{S}^{n-1}$ and $\mathcal{Y} = [0,1]$. The concept class $\mathrm{ReLU}(n,1)$ is reliably learnable with respect to any loss function that is convex, monotone, and L-Lipschitz and b-bounded in the interval $[0,1]$. The sample complexity and running time of the algorithm is polynomial in n, b, $\log(1/\delta)$ and $2^{O(L/\epsilon)}$. In particular, $\mathrm{ReLU}(n,1)$ is learnable in time polynomial in n, b and $\log(1/\delta)$ up to error $\epsilon \geq \epsilon_0 = \Theta(L/\log(n))$, where L is the Lipschitz constant of the loss function in the interval $[0,1]$.*

Proof In order to prove the theorem, we need to bound the following two losses for the output hypothesis h.

$$\mathcal{L}_{=0}(h;\mathcal{D}) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq 0 \wedge y = 0] \tag{4}$$

$$\mathcal{L}_{>0}(h;\mathcal{D}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(h(\mathbf{x}),y)\cdot\mathbb{I}(y>0)] \tag{5}$$

First, we analyze $\mathcal{L}_{=0}(h;\mathcal{D})$; in order to analyze this loss, it is useful to consider a slightly different loss function that is $(1/\epsilon)$-Lipschitz in its first argument, $\ell_{\epsilon\text{-zo}}(y',y)$. We define this loss separately for the case when $y>0$ and $y=0$. For $y>0$, we define $\ell_{\epsilon\text{-zo}}(y',y) := 0$ for all y'. For $y=0$, we define

$$\ell_{\epsilon\text{-zo}}(y',0) := \begin{cases} 0 & \text{if } y' \leq \epsilon \\ \frac{y'-\epsilon}{\epsilon} & \text{if } \epsilon < y' \leq 2\cdot\epsilon \\ 1 & \text{if } 2\cdot\epsilon < y'. \end{cases}$$

For $f : \mathcal{X} \to \mathcal{Y}$, let $\mathcal{L}_{\epsilon\text{-zo}}(f;\mathcal{D}) := \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell_{\epsilon\text{-zo}}(f(\mathbf{x}),y)]$. Let $d = O(1/\epsilon)$ be such that Theorem 24 applies for the class $\mathrm{ReLU}(n,1)$, and ψ_d and $\mathcal{H}_{\mathsf{MK}_d}$ the corresponding feature map and Hilbert space. Define $\mathcal{F}_B \subset \mathcal{H}_{\mathsf{MK}_d}$ as the set of all $f \in \mathcal{H}_{\mathsf{MK}_d}$ such that $\langle f,f\rangle \leq B$. Observe that for all $\mathbf{x} \in \mathcal{X} = \mathbb{S}^{n-1}$, $\langle \psi_d(\mathbf{x}), \psi_d(\mathbf{x})\rangle \leq \sum_{i=0}^{d}(\mathbf{x}\cdot\mathbf{x})^i = d+1$. Moreover, the function $\mathrm{clip}_{0,1} : \mathbb{R} \to [0,1]$ satisfies, $\mathrm{clip}_{0,1}(0) = 0$, and $\mathrm{clip}_{0,1}$ is 1-Lipschitz. Thus, Theorems 20 and 21 imply the following:

$$\mathcal{R}_m(\mathcal{F}_B) \leq \sqrt{\frac{(d+1)\cdot B}{m}}, \tag{6}$$

$$\mathcal{R}_m(\mathrm{clip}_{0,1}\circ\mathcal{F}_B) \leq 2\cdot\sqrt{\frac{(d+1)\cdot B}{m}} \tag{7}$$

The loss function $\ell_{\epsilon\text{-zo}}$ is $(1/\epsilon)$-Lipschitz in its first argument and 1-bounded on all of $\mathbb{R}$, so in particular in the interval $[0,1]$; the loss function ℓ (used for $\mathcal{L}_{>0}$) is L-Lipschitz in its first argument and b-bounded in the interval $[0,1]$ (by assumption in the theorem statement). We assume the following bound on m (note that it is polynomial in all the required factors):

$$m \geq \frac{1}{\epsilon^2}\left(8\max\{L,\epsilon^{-1}\}\sqrt{(d+1)\cdot B} + \max\{b,1\}\cdot\sqrt{2\log\frac{1}{\delta}}\right)^2. \tag{8}$$

Representative Sample Assumption: In the rest of the proof we assume that for the sample $S \sim \mathcal{D}^m$ used in the algorithm, it is the case that for loss functions $\ell_{\epsilon\text{-zo}}$ and ℓ and for all $f \in \mathcal{F}_B$, the following hold:

$$|\mathcal{L}_{\epsilon\text{-zo}}(f;\mathcal{D}) - \widehat{\mathcal{L}}_{\epsilon\text{-zo}}(f;S)| \leq \epsilon \tag{9}$$

$$|\mathcal{L}_{>0}(\text{clip}_{0,1} \circ f; \mathcal{D}) - \widehat{\mathcal{L}}_{>0}(\text{clip}_{0,1} \circ f; S)| \leq \epsilon \tag{10}$$

Theorems 19, 20 and 21 together with the bounds on the Rademacher complexity given by (6) and (7) and the facts that $\ell_{\epsilon\text{-zo}}$ is $1/\epsilon$-Lipschitz and 1-bounded on $\mathbb{R}$ and that ℓ is L-Lipschitz and b-bounded on $[0,1]$, imply that for m satisfying (8), this is the case with probability at least $1-2\delta$; we allow the algorithm to fail with probability 2δ.

Now consider the following to bound $\mathcal{L}_{=0}(h;\mathcal{D})$. From here on, f denotes the solution to the optimization problem in the algorithm.

$$\begin{aligned}
\mathcal{L}_{=0}(h;\mathcal{D}) &= \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) > 0 \wedge y = 0] \\
&\leq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell_{\epsilon\text{-zo}}(f(\mathbf{x}),y)] && (11)\\
&= \mathcal{L}_{\epsilon\text{-zo}}(f;\mathcal{D}) \\
&\leq \widehat{\mathcal{L}}_{\epsilon\text{-zo}}(f;S) + \epsilon \leq \epsilon, && (12)
\end{aligned}$$

Above in (11), we use the fact that for any $\mathbf{x}$ such that $h(\mathbf{x}) > 0$, it must be the case that $f(\mathbf{x}) > 2\cdot\epsilon$ and hence if $h(\mathbf{x}) > 0$ and $y = 0$, then $\ell_{\epsilon\text{-zo}}(f(\mathbf{x}),y) = 1$. Inequality (12) holds under the representative sample assumption using (9) (note that we have already accounted for the fact that the algorithm may fail with probability $O(\delta)$).

Next we give bounds on $\mathcal{L}_{>0}(h;\mathcal{D})$. We observe that for a loss function ℓ that is convex in its first argument, monotone, L-Lipschitz, and b-bounded in the interval $[0,1]$, the following holds for any $y \in (0,1]$:

$$\ell(h(\mathbf{x}),y) \leq \ell(\text{clip}_{0,1}(f(\mathbf{x})),y) + 2\epsilon L \tag{13}$$

Clearly, whenever $f(\mathbf{x}) > 2\epsilon$ or $f(\mathbf{x}) < 0$, the above statement is trivially true. If $f(\mathbf{x}) \in [0, 2\epsilon]$ the statement follows from the L-Lipschitz continuity of $\ell(\cdot,y)$ in the interval $[0,1]$.

Let $\mathbf{w} \in \mathbb{R}^n$ be such that $\mathcal{L}_{=0}(\text{relu}_{\mathbf{w}};\mathcal{D}) = 0$ and let $p_{\mathbf{w}}$ be the corresponding polynomial ϵ-approximation in $\mathcal{H}_{\mathsf{MK}_d}$ (cf. Theorem 24). Then consider the following:

$$\begin{aligned}
\mathcal{L}_{>0}(h;\mathcal{D}) &= \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\ell(h(\mathbf{x}),y)\cdot\mathbb{I}(y>0)\right] \\
&\leq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\ell(\text{clip}_{0,1}(f(\mathbf{x})),y)\cdot\mathbb{I}(y>0)\right] + 2\epsilon L && (14)\\
&= \mathcal{L}_{>0}(\text{clip}_{0,1}(f);\mathcal{D}) + 2\epsilon L \\
&\leq \widehat{\mathcal{L}}_{>0}(\text{clip}_{0,1}(f);S) + \epsilon + 2\epsilon L && (15)\\
&\leq \widehat{\mathcal{L}}_{>0}(f;S) + \epsilon + 2\epsilon L && (16)\\
&\leq \widehat{\mathcal{L}}_{>0}(p_{\mathbf{w}};S) + \epsilon + 2\epsilon L && (17)\\
&= \widehat{\mathcal{L}}_{>0}(\text{clip}_{0,1}\circ p_{\mathbf{w}};S) + \epsilon + 2\epsilon L && (18)\\
&\leq \mathcal{L}_{>0}(\text{clip}_{0,1}\circ p_{\mathbf{w}};\mathcal{D}) + 2\epsilon + 2\epsilon L && (19)\\
&\leq \mathcal{L}_{>0}(p_{\mathbf{w}};\mathcal{D}) + 2\epsilon + 2\epsilon L && (20)\\
&\leq \mathcal{L}_{>0}(\text{relu}_{\mathbf{w}};\mathcal{D}) + 2\epsilon + 3\epsilon L && (21)
\end{aligned}$$

Step (14) is obtained simply by applying (13). Step (15) follows using the representative sample assumption using (10). Step (16) follows by the monotone property of $\ell(\cdot,y)$; in particular, it must

always be the case that either $y \leq \mathrm{clip}_{0,1}(f(\mathbf{x})) \leq f(\mathbf{x})$ or $f(\mathbf{x}) \leq \mathrm{clip}_{0,1}(f(\mathbf{x})) \leq y$; thus $\ell(\mathrm{clip}_{0,1}(f(\mathbf{x})), y) \leq \ell(f(\mathbf{x}), y)$. Step (17) follows from the fact that f is the optimal solution to Optimization Problem 3 and $p_{\mathbf{w}}$ is a *feasible* solution. Steps (18) and (20) use the fact that $\mathrm{clip}_{0,1} \circ p_{\mathbf{w}} = p_{\mathbf{w}}$ as $p_{\mathbf{w}}(\mathbb{S}^{n-1}) \subseteq [0,1]$. Step (19) follows under the representative sample assumption using (10). And finally, Step (21) follows as both $\mathrm{relu}_{\mathbf{w}}(\mathbf{x}) \in [0,1]$ and $p_{\mathbf{w}}(\mathbf{x}) \in [0,1]$ for $\mathbf{x} \in \mathbb{S}^{n-1}$, $|p_{\mathbf{w}}(\mathbf{x}) - \mathrm{relu}_{\mathbf{w}}(\mathbf{x})| \leq \epsilon$ and the L-Lipschitz continuity of ℓ in the interval $[0,1]$.

As the argument holds for any $\mathbf{w} \in \mathbb{S}^{n-1}$ satisfying $\mathcal{L}_{=0}(\mathrm{relu}_{\mathbf{w}}; \mathcal{D}) = 0$ this completes the proof of theorem by rescaling ϵ to $\epsilon/(2+3L)$ and δ to $\delta/2$. ■

DISCUSSION: DEPENDENCE ON THE LIPSCHITZ CONSTANT

Theorem 26 gives a sample complexity and running time bound that is polynomial on $2^{O(L/\epsilon)}$ (in addition to being polynomial in other parameters). Recall that, here, L is the Lipschitz constant of the loss function ℓ on the interval $[0,1]$. For many loss functions, such as ℓ_p-loss for constant p, hinge loss, logistic loss, etc., the value of L is a constant. Nonetheless, it is instructive to examine why we obtain such a dependence L, and identify some restricted settings in which this dependence can be avoided.

The dependence of our running time and sample complexity bounds on L arises due to Steps (13) and (21) in the proof of Theorem 26, where the excess error compared to the optimal ReLU is bounded above by $O(L\epsilon)$. This requires us to start with a polynomial that is an $O(\epsilon/L)$-uniform approximation to the σ_{relu} activation function, to ensure excess error at most ϵ. We showed that such an approximating polynomial exists, with degree $O(L/\epsilon)$ and with coefficients whose squares sum to $2^{O(L/\epsilon)}$.

It is sometimes possible to avoid this exponential dependence on L in the setting of agnostic learning (as opposed to reliable learning). Indeed, in the case of agnostic learning there is no need to threshold the output at 2ϵ (this thresholding contributed $2\epsilon L$ to our bound on the excess error established in Inequality (13)); simply clipping the output to be in the range of $\mathcal{Y}$ suffices.

3.4. An Implication for Learning Convex Neural Networks

In a recent work, Bach (2014) considered convex relaxations of optimization problems related to learning neural networks with a single hidden layer and non-decreasing homogeneous activation function.[7] One specific problem raised in his paper Bach (2014, Sec. 6) is understanding the computational complexity of the following problem.

Problem 27 (Incremental Optimization Problem (Bach, 2014)) *Let* $\langle(\mathbf{x}_i, y_i)\rangle_{i=1}^m \in (\mathbb{S}^{n-1} \times [-1,1])^m$. *Find a* $\mathbf{w} \in \mathbb{S}^{n-1}$ *that maximizes* $\frac{1}{m}\sum_{i=1}^m y_i \cdot \mathrm{relu}_{\mathbf{w}}(\mathbf{x}_i)$.

While Bach (2014) considers the setting where $y_i \in \mathbb{R}$, rather than $[-1,1]$, we focus on the case when $y_i \in [-1,1]$. The problem as posed above is an optimization problem on a finite dataset that requires the output solution to be from a specific class, in this case a ReLU. In our setting, this can be rephrased as a (proper) learning problem where the goal is to output a hypothesis that has expected loss, defined by $\ell(y', y) = -y' \cdot y$, not much larger than the best possible ReLU, given access to draws from a distribution over $\mathbb{S}^{n-1} \times [-1,1]$. Here, we relax this goal to improper learning, where

[7] His setting allows potentially uncountably many hidden units along with a sparsity-inducing regularizer.

the algorithm is permitted to output a hypothesis that is not itself a ReLU. The same approach as used in the proof of Theorem 26 can be used to give a polynomial-time approximation scheme for approximately solving this problem to within ϵ of optimal, in time $2^{O(1/\epsilon)} \cdot n^{O(1)}$.

We describe the modified algorithm and the minor differences in the proof below.

Optimization Problem 4

$$\begin{aligned} \underset{\boldsymbol{\alpha}\in\mathbb{R}^m}{\text{minimize}} \quad & \sum_{i=1}^{m} \ell\left(\sum_{j=1}^{m} \alpha_j \mathsf{MK}_d(\mathbf{x}_j, \mathbf{x}_i), y_i\right) \\ \text{subject to} \quad & \sum_{i,j=1}^{m} \alpha_i \alpha_j \mathsf{MK}_d(\mathbf{x}_i, \mathbf{x}_j) \le B \end{aligned}$$

The loss function used is $\ell(y', y) = -y'y$. Let $\boldsymbol{\alpha}^*$ denote an optimal solution to Optimization Problem 4 and let $f(\cdot) = \sum_{i=1}^{m} \alpha_i^* \mathsf{MK}_d(\mathbf{x}_i, \cdot)$. In Problem 27, there is no reliability required and hence we do not threshold negative (or sufficiently small positive) values as was done in Section 3.2. Likewise, we do not clip the function f; this is because while the loss function $\ell(y', y) = -y'y$ is indeed convex in its first argument, 1-Lipschitz on $\mathbb{R}$, and $\sqrt{B}$-bounded on the interval $[-\sqrt{B}, \sqrt{B}]$ (for $y \in [-1, 1]$; note that $|f(\mathbf{x})| \le |\langle f, \psi_d(\mathbf{x})\rangle| \le \sqrt{B}$ by the Cauchy-Schwartz inequality), it is very much *not* monotone. Thus, it is no longer the case that $\mathrm{clip}_{-1,1}(f)$ is a better hypothesis than f itself. We observe that the proof of Theorem 26 only makes use of the monotone nature of ℓ to conclude that expected loss of $\mathrm{clip}_{0,1} \circ f$ is less than that of f. As we no longer output a clipped hypothesis, this is not necessary.

Theorem 28 *Given i.i.d. examples $(\mathbf{x}_i, y_i)$ drawn from an (unknown) distribution $\mathcal{D}$ over $\mathbb{S}^{n-1} \times [-1, 1]$, there is an algorithm that outputs a hypothesis h such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[-y \cdot h(\mathbf{x})] \le \min_{\mathbf{w}\in\mathbb{S}^{n-1}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[-y \cdot \mathrm{relu}_{\mathbf{w}}(\mathbf{x})] + \epsilon$. The algorithm runs in time $2^{O(1/\epsilon)} \cdot n^{O(1)}$.*

3.5. Noisy Polynomial Reconstruction over $\mathbb{S}^{n-1}$

In the noisy polynomial reconstruction problem, a learner is given access to examples drawn from a distribution and labeled according to a function $f(\mathbf{x}) = p(\mathbf{x}) + w(\mathbf{x})$ where p is a polynomial and w is an arbitrary function (corresponding to noise). We will consider a more general scenario, where a learner is given sample access to an *arbitrary* distribution $\mathcal{D}$ on $\mathbb{S}^{n-1} \times [-1, 1]$ and must output the best fitting polynomial with respect to some fixed loss function. We say that the reconstruction is *proper* if, given a hypothesis h encoding a multivariate polynomial, we can obtain any coefficient of our choosing in time polynomial in n.

Note that noisy polynomial reconstruction as defined above is equivalent to the problem of agnostically learning multivariate polynomials. We give an algorithm for noisy polynomial reconstruction whose runtime is $\mathrm{poly}(B, n, d, 1/\epsilon)$, where B is an upper bound on the sum of the squared coefficients of the polynomial in the standard monomial basis. Throughout this section, we refer to the sum of the squared coefficients of p as the *weight* of p.

Analogous problems over the Boolean domain are thought to be computationally intractable. Andoni et al. (2014) were the first to observe that over non-Boolean domains, the problem admits

some non-trivial solutions. In particular, they gave an algorithm that runs in time $\mathrm{poly}(B, n, 2^d, 1/\epsilon)$ with the requirement that the underlying distribution be a product distribution over the unit cube (and that the noise function is structured).

Consider a multivariate polynomial p of degree d such that the sum of the squared coefficients is bounded by B. Denote the coefficient of monomial $x_1^{i_1} \cdots x_n^{i_n}$ by $\beta(i_1, \ldots, i_n)$ for $(i_1, \ldots, i_n) \in \{0, \ldots, d\}^n$. We have

$$p(\mathbf{x}) = \sum_{\substack{(i_1,\ldots,i_n)\in\{0,\ldots,d\}^n \\ i_i+\cdots+i_n\leq d}} \beta(i_1, \ldots, i_n) x_1^{i_1} \cdots x_n^{i_n} \tag{22}$$

such that

$$\sum_{\substack{(i_1,\ldots,i_n)\in\{0,\ldots,d\}^n \\ i_i+\cdots+i_n\leq d}} \beta(i_1, \ldots, i_n)^2 \leq B.$$

Let M be the map that takes an ordered tuple $(k_1, \ldots, k_j) \in [n]^j$ for $j \in [d]$ to the tuple $(i_1, \ldots, i_n) \in \{0, \ldots, d\}^n$ such that $x_{k_1} \cdots x_{k_j} = x_1^{i_1} \cdots x_n^{i_n}$. Let $C(i_1, \ldots, i_n)$ be the number of distinct orderings of the i_j's for $j \in \{0, \ldots, n\}$; $C(i_1, \ldots, i_n)$ which can be computed from the multinomial theorem (cf. Wikipedia (2016a)). Observe that the number of tuples that M maps to $(i_1, \ldots, i_n)$ is precisely $C(i_1, \ldots, i_n)$.

Recall that $\mathcal{H}_{\mathsf{MK}_d}$ denotes the RKHS from Definition 16. Observe that the polynomial p from Equation (22) is represented by the vector $\mathbf{v}_p \in \mathcal{H}_{\mathsf{MK}_d}$ defined as follows. For $j \in [d]$, entry $(k_1, \ldots, k_j)$ of $\mathbf{v}_p$ equals

$$\frac{\beta\left(M\left(k_1, \ldots, k_j\right)\right)}{C\left(M\left(k_1, \ldots, k_j\right)\right)}.$$

It is easy to see that $\mathbf{v}_p$ as defined represents p. Indeed,

$$\begin{aligned}
\langle \mathbf{v}_p, \psi_d(\mathbf{x}) \rangle &= \sum_{j=0}^{d} \sum_{(k_1,\ldots,k_j)\in[n]^j} \frac{\beta\left(M\left(k_1, \ldots, k_j\right)\right)}{C\left(M\left(k_1, \ldots, k_j\right)\right)} x_{k_1} \cdots x_{k_j} \\
&= \sum_{j=0}^{d} \sum_{\substack{(i_1,\ldots,i_n)\in\{0,\ldots,d\}^n \\ i_i+\cdots+i_n=j}} C\left(i_1, \ldots, i_n\right) \frac{\beta\left(i_1, \ldots, i_n\right)}{C\left(i_1, \ldots, i_n\right)} x_1^{i_1} \cdots x_n^{i_n} = p(\mathbf{x}).
\end{aligned}$$

Furthermore, we can compute,

$$\begin{aligned}
\langle \mathbf{v}_p, \mathbf{v}_p \rangle &= \sum_{j=0}^{d} \sum_{(k_1,\ldots,k_j)\in[n]^j} \frac{\beta\left(M\left(k_1, \ldots, k_j\right)\right)^2}{C\left(M\left(k_1, \ldots, k_j\right)\right)^2} \\
&= \sum_{j=0}^{d} \sum_{\substack{(i_1,\ldots,i_n)\in\{0,\ldots,d\}^n \\ i_i+\cdots+i_n=j}} C\left(i_1, \ldots, i_n\right) \frac{\beta\left(i_1, \ldots, i_n\right)^2}{C\left(i_1, \ldots, i_n\right)^2} \\
&\leq \sum_{j=0}^{d} \sum_{\substack{(i_1,\ldots,i_n)\in\{0,\ldots,d\}^n \\ i_i+\cdots+i_n=j}} \beta\left(i_1, \ldots, i_n\right)^2 \leq B.
\end{aligned}$$

Overview of the Algorithm. Let $\mathcal{C}$ be the class of all multivariate polynomials and let $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ be a training set of examples drawn i.i.d. from some arbitrary distribution $\mathcal{D}$ on $\mathbb{S}^{n-1} \times [-1, 1]$. Similar to Optimization Problem 2 in Section 3.1, we wish to solve Optimization Problem 5 below.

Optimization Problem 5

$$\begin{aligned} \underset{\mathbf{v} \in \mathcal{H}_{\mathsf{MK}_d}}{\text{minimize}} \quad & \sum_{i=1}^{m} \ell(\langle \mathbf{v}, \psi_d(\mathbf{x}_i) \rangle, y_i) \\ \text{subject to} \quad & \langle \mathbf{v}, \mathbf{v} \rangle \leq B \end{aligned}$$

Notice from the previous analysis, a degree d polynomial p can be represented as a vector $\mathbf{v}_p \in \mathcal{H}_{\mathsf{MK}_d}$ such that $p(\mathbf{x}) = \langle \mathbf{v}_p, \psi_d(\mathbf{x}) \rangle$ for all $\mathbf{x} \in \mathbb{S}^{n-1}$, and $\langle \mathbf{v}_p, \mathbf{v}_p \rangle \leq B$. Thus, $\mathbf{v}_p$ is a feasible solution to Optimization Problem 5. Optimization Problem 5 can easily be solved in time $\mathrm{poly}(n^d)$, but this runtime is not polynomial in B and n. Instead, just as in Section 3.1, we use the Representer Theorem to solve Optimization Problem 5 in time that is polynomial in the number of samples used. Specifically, the Representer Theorem states that there is an optimal solution to Optimization Problem 5 of the form $\mathbf{v} = \sum_{i=1}^{m} \alpha_i \psi_d(\mathbf{x}_i)$ for some values $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$. Thus, Optimization Problem 5 can be reformulated in terms of the variable vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$. This mathematical program is described as Optimization Problem 6 below.

Optimization Problem 6

$$\begin{aligned} \underset{\boldsymbol{\alpha} \in \mathbb{R}^m}{\text{minimize}} \quad & \sum_{i=1}^{m} \ell \left(\sum_{j=1}^{m} \alpha_j \mathsf{MK}_d(\mathbf{x}_j, \mathbf{x}_i), y_i \right) \\ \text{subject to} \quad & \sum_{i,j=1}^{m} \alpha_i \cdot \alpha_j \cdot \mathsf{MK}_d(\mathbf{x}_i, \mathbf{x}_j) \leq B \end{aligned}$$

Via a standard analysis identical to that of Section 3.1, Optimization Problem 6 is a convex program and can be solved in time polynomial in m, n, and d. Let $\boldsymbol{\alpha}^*$ denote an optimal solution to Optimization Problem 6 and let $f(\cdot) = \sum_{i=1}^{m} \alpha_i^* \mathsf{MK}_d(\mathbf{x}_i, \cdot)$. The hypothesis h output by our algorithm is as follows.

$$h(\mathbf{x}) = \mathrm{clip}_{-1,1}(f(\mathbf{x})).$$

Observe that $h \in \mathrm{clip}_{-1,1} \circ \mathcal{C}$.

3.6. Proper Learning

As discussed in Section 3.2, we require clipping to avoid a weak bound on the generalization error for general loss functions. If, however, we consider learning with respect to any ℓ_p loss for constant

$p \geq 1$, it can be shown that we can do without clipping (with only a polynomial factor increase in sample complexity). In this case, the learner $h = f$ is a *proper learner* in the following sense. Recalling the feature map ψ_d associated with MK_d from Definition 16, we can compute the coefficient $\beta(I)$ for $I = (i_1, \ldots, i_n) \in \{0, \ldots, d\}^n$ corresponding to the monomial $x_1^{i_1} \cdots x_n^{i_n}$.

$$\beta(I) = \sum_{i=1}^{m} \alpha_i^* \sum_{\substack{k_1,\ldots,k_j \in [n]^j \\ j \in \{0,\ldots,d\} \\ M(k_1,\ldots,k_j) = (i_1,\ldots,i_n)}} (\mathbf{x}_i)_{k_1} \cdots (\mathbf{x}_i)_{k_j} = \sum_{i=1}^{m} \alpha_i^* C\left(i_1, \ldots, i_n\right) (\mathbf{x}_i)_1^{i_1} \cdots (\mathbf{x}_i)_n^{i_n}$$

Observe that the above can be easily computed since we know $\mathbf{x}_i$ for all $i \in [m]$, and the function C can be efficiently computed as discussed before using the multinomial theorem. Hence, the hypothesis is itself a polynomial of degree at most d, any desired coefficient of which can be computed efficiently.

3.7. Formal Version of Theorem 7 and Its Proof

The rest of this section is devoted to the proof of Theorem 7 (or, more precisely, its formal variant Theorem 29 below, which makes explicit the conditions on the loss function ℓ that are required for the theorem to hold). In particular, we show that whenever the sample size m is a sufficiently large polynomial in d, n, B, $1/\epsilon$, and $\log(1/\delta)$, the hypothesis h output by the algorithm satisfies

$$\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim D}[\ell(h(\mathbf{x}), y)] \leq \mathsf{opt} + \epsilon.$$

where opt is the error of the best fitting multivariate polynomial p of degree d whose sum of squares of coefficients is bounded by B.

Theorem 29 (Formal Version of Theorem 7) *Let $\mathcal{P}(n, d, B)$ be the class of polynomials $p\colon \mathbb{S}^{n-1} \to [-1, 1]$ in n variables such that the total degree of p is at most d, and the sum of squares of coefficients of p (in the standard monomial basis) is at most B. Let ℓ be any loss function that is convex, monotone, and L-Lipschitz and b-bounded in the interval $[-1, 1]$. Then $\mathrm{poly}(n, d, B)$ is agnostically learnable under any (unknown) distribution over $\mathbb{S}^{n-1} \times [-1, 1]$ with respect to the loss function ℓ in time $\mathsf{poly}(n, d, B, 1/\epsilon, L, b, \log \frac{1}{\delta})$. The learning algorithm is proper if the loss function ℓ is the ℓ_p loss function for constant $p > 0$.*

Proof In order to prove the theorem, we need to bound

$$\mathcal{L}(h; D) = \mathbb{E}_{(\mathbf{x},y)\sim D}[\ell(h(\mathbf{x}), y)].$$

We know that for all $\mathbf{x} \in \mathbb{S}^{n-1}$, $\langle \psi_d(\mathbf{x}), \psi_d(\mathbf{x}) \rangle = d + 1$. Moreover, letting $\mathbf{v}_p$ be the corresponding element of the RKHS for polynomial $p \in \mathcal{C}$, we know from previous analysis that $\langle \mathbf{v}_p, \mathbf{v}_p \rangle \leq B$. In addition, the function $\mathrm{clip}_{-1,1} : \mathbb{R} \to [-1, 1]$ satisfies $\mathrm{clip}_{-1,1}(0) = 0$, and $\mathrm{clip}_{-1,1}$ is 1-Lipschitz. Thus, Theorems 20 and 21 imply the following:

$$\mathcal{R}_m(\mathcal{C}) \leq \sqrt{\frac{(d+1) \cdot B}{m}}, \tag{23}$$

$$\mathcal{R}_m(\mathrm{clip}_{-1,1} \circ \mathcal{C}) \leq 2 \cdot \sqrt{\frac{(d+1) \cdot B}{m}}. \tag{24}$$

By assumption, ℓ is L-Lipschitz in its first argument and b-bounded in the interval $[-1, 1]$. We assume the following bound on m (note that it is polynomial in all the required factors):

$$m \geq \frac{1}{\epsilon^2}\left(8\mathsf{max}\{L, \epsilon^{-1}\}\sqrt{(d+1)\cdot B} + \mathsf{max}\{b, 1\}\cdot\sqrt{2\log\frac{1}{\delta}}\right)^2. \tag{25}$$

In the rest of the proof we assume that for every $f \in \mathcal{P}(n, d, B)$, the following hold:

$$|\mathcal{L}(f; D) - \widehat{\mathcal{L}}(f; S)| \leq \epsilon. \tag{26}$$

$$|\mathcal{L}(\mathrm{clip}_{-1,1} \circ f; D) - \widehat{\mathcal{L}}(\mathrm{clip}_{-1,1} \circ f; S)| \leq \epsilon. \tag{27}$$

Using Theorem 19 together with the bounds on Rademacher complexity given by (23) and (24) and the L-Lipschitz continuity in its first argument and b-boundedness of ℓ on the interval $[-1, 1]$, we get that the above inequalities hold with probability at least $1 - 2\delta$. We let the algorithm fail with probability 2δ.

Now consider the following to bound $\mathcal{L}(h; D)$. Letting p be any polynomial in $\mathcal{P}(n, d, B)$,

$$\mathcal{L}(h; D) \leq \widehat{\mathcal{L}}(h; S) + \epsilon \tag{28}$$

$$\leq \widehat{\mathcal{L}}(f; S) + \epsilon \tag{29}$$

$$\leq \widehat{\mathcal{L}}(p; S) + \epsilon \tag{30}$$

$$\leq \mathcal{L}(p; D) + 2\cdot\epsilon \tag{31}$$

Above in (28), we appeal to (27). In (29), we use the fact that D is a distribution over $\mathbb{S}^{n-1} \times [-1, 1]$, and ℓ is monotone. In (30), we use the fact that the coefficient vector of p is a feasible solution to Optimization Problem 5, and Optimization Problem 6 is a reformulation of Optimization Problem 5. Finally, in (31), we appeal (26).

The theorem now follows by replacing ϵ with $\epsilon/2$, δ with $\delta/2$, and observing that the algorithm runs in time $\mathrm{poly}(m) = \mathsf{poly}(n, d, B, 1/\epsilon, L, b, \log\frac{1}{\delta})$. ■

4. Networks of ReLUs

In this section, we extend learnability results for a single ReLU to network of ReLUs. Our results in this section apply to the standard agnostic model of learning in the case that the output is a linear combination of hidden units. If our output layer, however, is a single ReLU, then our results can be extended to the reliable setting using similar techniques from Section 3.

We will use the same framework as Zhang et al. (2016a), who showed how to learn networks where the activation function is computed *exactly* by a power series (with bounded sum of squares of coefficients B) with respect to loss functions that are bounded on a domain that is a function of B. Their algorithm works by repeatedly composing the kernel of Shalev-Shwartz et al. (2011) and optimizing in the corresponding RKHS.

Note, however, that since σ_{relu} is not differentiable at 0, there is no power series for σ_{relu}, and the approach of Zhang et al. (2016a) cannot be used; their work applies to a smooth activation function that has a shape that is "Sigmoid-like" or "ReLU-like," but is not a good approximation to σ_{relu} in a precise mathematical sense.

We generalize their results to activation functions that are *approximated* by polynomials. This allows us to capture many classes of activation functions including ReLUs. Our clipping technique also allows us to work with respect to a broader class of loss functions.

Our results for learning networks of ReLUs have a number of new applications. First, we give the first efficient algorithms for learning "parameterized" ReLUs and "leaky" ReLUs. Second, we obtain the first polynomial-time approximation schemes for convex piecewise-linear regression (see Section 4.5 for details). As far as we are aware, there were no provably efficient algorithms known for these types of multivariate piecewise-linear regression problems.

4.1. Notation

We use the following notation of Zhang et al. (2016a). Consider a network with D hidden layers and an output unit (we assume that the output is one-dimensional). Let $\sigma : \mathbb{R} \to \mathbb{R}$ denote the activation function applied at each unit of all the hidden layers. Let $n^{(i)}$ denote the number of units in hidden layer i with $n^{(0)} = n$ (i.e., input dimension) and $w_{jk}^{(i)}$ be the weight of the edge between unit j in layer i and unit k in layer $i+1$. We define, $y_j^{(i)}$ to be the function that maps $\mathbf{x} \in \mathcal{X}$ to the output of unit j in layer i,

$$y_j^{(i)}(\mathbf{x}) = \sigma\left(\sum_{k=1}^{n^{(i-1)}} w_{jk}^{(i-1)} \cdot y_k^{(i-1)}(\mathbf{x})\right),$$

where $y_j^{(0)}(\mathbf{x}) = \mathbf{x}$ for all j. We similarly define $h_j^{(i)}$ to be the function that maps $\mathbf{x} \in \mathcal{X}$ to the input of unit j in layer $i+1$:

$$h_j^{(i)}(\mathbf{x}) = \sum_{k=1}^{n^{(i)}} w_{jk}^{(i)} \cdot y_k^{(i)}(\mathbf{x}).$$

Finally, we define the output of the network as a function $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}$ as

$$\mathcal{N}(\mathbf{x}) = \sum_{k=1}^{n^{(D)}} w_{1k}^{(D)} \cdot y_k^{(D)}(\mathbf{x}).$$

For a better understanding of the above notation, consider a fully-connected network $\mathcal{N}_1$ with a single hidden layer (these are also known as depth-2 networks) consisting of k units:

$$\mathcal{N}_1 : \mathbf{x} \mapsto \sum_{i=1}^{k} u_i \sigma(\mathbf{w}_i \cdot \mathbf{x}).$$

In this case, output of unit $i \in [k]$ in the hidden layer is $y_i^{(1)}(\mathbf{x}) = \sigma(\mathbf{w}_i \cdot \mathbf{x})$ and the input to the same unit is $h_i^{(0)}(\mathbf{x}) = \mathbf{w}_i \cdot \mathbf{x}$.

We consider a class of networks with edge weights of bounded ℓ_1 or ℓ_2 norm. The class is formalized as follows.

Definition 30 (Weight-bounded Networks) *Let $\mathcal{N}[\sigma, D, W, M]$ be the class of fully-connected networks with D hidden layers and σ as the activation function. Additionally, the weights are*

constrained such that $\sum_{j=1}^{n}(w_{ij}^{(0)})^2 \leq M^2$ *for all units* i *in layer 0 and* $\sum_{k=1}^{n^{(i)}}|w_{jk}^{(i)}| \leq W$ *for all units* j *in all layers* $i \in \{1,\ldots,D\}$*. Also, the inputs to each unit are bounded in magnitude by* M*, i.e.,* $h_j^{(l)}(\mathbf{x}) \in [-M, M]$ *with* $M \geq 1$ *for each* $l < D$ *and* $j = 1,\ldots,n^{(l+1)}$.

We consider activation functions which can be approximated by polynomials with sum of squares of coefficients bounded. We term them *low-weight approximable* activation functions, formalized as follows.

Definition 31 (Low-weight Approximable Functions) *For activation function* $\sigma : \mathbb{R} \to \mathbb{R}$*, for* $\epsilon \in (0,1)$*,* $M \geq 1$*,* $B \geq 1$*, we say that a polynomial* $p(t) = \sum_{i=1}^{d}\beta_i t^i$ *is a degree* d*,* (ϵ, M, B)*-approximation to* σ *if for every* $t \in [-M, M]$*,* $|\sigma(t) - p(t)| \leq \epsilon$ *and furthermore,* $\sum_{i=0}^{d} 2^i\beta_i^2 \leq B$.

4.2. Approximate Polynomial Networks

We first bound the error incurred when each activation function is replaced by a corresponding low-weight polynomial approximation.

Theorem 32 (Approximate Polynomial Network) *Let* σ *be an activation function that is 1-Lipschitz*[8] *and such that there exists a degree* d *polynomial* p *that is a* $(\frac{\epsilon}{W^D D}, 2M, B)$ *approximation for* σ*, with* $\epsilon \in (0,1)$*, with* $d, M, B \geq 1$*. Then, for all* $\mathcal{N} \in \mathcal{N}[\sigma, D, W, M]$*, there exists* $\bar{\mathcal{N}} \in \mathcal{N}[p, D, W, 2M]$ *such that*

$$\sup_{\mathbf{x}\in\mathbb{S}^{n-1}} \left|\mathcal{N}(\mathbf{x}) - \bar{\mathcal{N}}(\mathbf{x})\right| \leq \epsilon.$$

Proof Let $\mathcal{N} \in \mathcal{N}[\sigma, D, W, M]$ and let $\bar{\mathcal{N}} \in \mathcal{N}[p, D, W, M]$ be such that it has the same structure and weights as $\mathcal{N}$ with the activation replaced with p. For $\mathcal{N}$ let $h^{(i)}(\mathbf{x})$ be the inputs to layer $i+1$ and $y^{(i)}(\mathbf{x})$ be the outputs of layer i as defined previously. Correspondingly, for $\bar{\mathcal{N}}$ let $\bar{h}^{(i)}(\mathbf{x})$ be the inputs to layer $i+1$ and $\bar{y}^{(i)}(\mathbf{x})$ be the outputs of layer i. We prove by induction on layer i that for all units j of layer i,

$$\sup_{\mathbf{x}\in\mathbb{S}^{n-1}} \left|h_j^{(i)}(\mathbf{x}) - \bar{h}_j^{(i)}(\mathbf{x})\right| \leq \frac{i\epsilon}{W^{D-i}D}. \tag{32}$$

For layer $i = 0$, we have $h_j^{(0)}(\mathbf{x}) = \bar{h}_j^{(0)}(\mathbf{x}) = \mathbf{w}_j^{(0)} \cdot \mathbf{x} \in [-M, M]$ which trivially satisfies (32). Now, we prove that the desired property holds for layer l, assuming the following holds for layer $l-1$. We have for all units j in layer $l-1$,

$$\sup_{\mathbf{x}\in\mathbb{S}^{n-1}} \left|h_j^{(l-1)}(\mathbf{x}) - \bar{h}_j^{(l-1)}(\mathbf{x})\right| \leq \frac{(l-1)\epsilon}{W^{D-l+1}D}. \tag{33}$$

Note that this implies that $\left|\bar{h}_j^{(l-1)}(\mathbf{x})\right| \leq \left|h_j^{(l-1)}(\mathbf{x})\right| + \frac{(l-1)\epsilon}{W^{D-l+1}D} \leq 2M$. Here the second inequality follows from the assumption that inputs to each unit are bounded by M and $\epsilon < 1$. We have for all $\mathbf{x}$ and j,

$$\left|h_j^{(l)}(\mathbf{x}) - \bar{h}_j^{(l)}(\mathbf{x})\right| = \left|\sum_{k=1}^{n^{(l)}} w_{jk}^{(l)} \cdot \sigma\left(h_k^{(l-1)}(\mathbf{x})\right) - \sum_{k=1}^{n^{(l)}} w_{jk}^{(l)} \cdot p\left(\bar{h}_k^{(l-1)}(\mathbf{x})\right)\right|$$

[8] Note that this is not a restriction, as we have not explicitly constrained the weights W. Thus, to allow a Lipschitz constant L, we simply replace W by WL.

$$
\begin{aligned}
&= \sum_{k=1}^{n^{(l)}} \left|w_{jk}^{(l)}\right| \left|\sigma\left(h_k^{(l-1)}(\mathbf{x})\right) - p\left(\bar{h}_k^{(l-1)}(\mathbf{x})\right)\right| \\
&\leq \sum_{k=1}^{n^{(l)}} \left|w_{jk}^{(l)}\right| \left(\left|\sigma\left(h_k^{(l-1)}(\mathbf{x})\right) - \sigma\left(\bar{h}_k^{(l-1)}(\mathbf{x})\right)\right| + \frac{\epsilon}{W^D D}\right) \qquad (34) \\
&\leq \sum_{k=1}^{n^{(l)}} \left|w_{jk}^{(l)}\right| \left(\left|h_k^{(l-1)}(\mathbf{x}) - \bar{h}_k^{(l-1)}(\mathbf{x})\right| + \frac{\epsilon}{W^D D}\right) \qquad (35) \\
&\leq \sum_{k=1}^{n^{(l)}} \left|w_{jk}^{(l)}\right| \left(\frac{(l-1)\epsilon}{W^{D-l+1} D} + \frac{\epsilon}{W^D D}\right) \qquad (36) \\
&= \|\mathbf{w}_j^{(l)}\|_1 \frac{l \cdot \epsilon}{W^{D-l+1} D} \\
&\leq \frac{l \cdot \epsilon}{W^{D-l} D} \qquad (37)
\end{aligned}
$$

Step (34) follows since $\bar{h}_j^{(l-1)}(\mathbf{x}) \in [-2M, 2M]$ and p uniformly $\frac{\epsilon}{W^D D}$-approximates σ in $[-2M, 2M]$. Step (35) follows from σ being 1-Lipschitz. Step (36) follows from (33). Finally Step (37) follows from $\|\mathbf{w}_j^{(l)}\|_1 \leq W$ which is given. This completes the inductive proof.

We conclude by noting that $\mathcal{N}(\mathbf{x}) = h_1^{(D)}(\mathbf{x})$ and $\bar{\mathcal{N}}(\mathbf{x}) = \bar{h}_1^{(D)}(\mathbf{x})$. Thus, from above we get,

$$
\sup_{\mathbf{x} \in \mathbb{S}^{n-1}} \left|\mathcal{N}(\mathbf{x}) - \bar{\mathcal{N}}(\mathbf{x})\right| = \sup_{\mathbf{x} \in \mathbb{S}^{n-1}} \left|h_1^{(N)}(\mathbf{x}) - \bar{h}_1^{(N)}(\mathbf{x})\right| \leq \epsilon.
$$

This completes the proof. ■

Given the above transformation to a polynomial network and associated error bounds, we apply the main theorem of Zhang et al. (2016a) combined with the clipping technique from Section 3 to obtain the following result:

Theorem 33 (Learnability of Neural Network) *Let σ be an activation function that is 1-Lipschitz*[8] *and such that there exists a degree d polynomial p that is an $(\frac{\epsilon}{(L+1)\cdot W^D \cdot D}, 2M, B)$ approximation for σ, for $d, B, M \geq 1$. Let ℓ be a loss function that is convex, L-Lipschitz in the first argument, and b bounded on $[-2M \cdot W, 2M \cdot W]$. Then there exists an algorithm that outputs a predictor $\widehat{f}$ such that with probability at least $1-\delta$, for any (unknown) distribution $\mathcal{D}$ over $\mathbb{S}^{n-1} \times [-M \cdot W, M \cdot W]$,*

$$
\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\widehat{f}(\mathbf{x}), y)] \leq \min_{\mathcal{N} \in \mathcal{N}[\sigma, D, W, M]} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\mathcal{N}(\mathbf{x}), y)] + \epsilon.
$$

The time complexity of the above algorithm is bounded by $n^{O(1)} \cdot B^{O(d)^{D-1}} \cdot \log(1/\delta)$, where d is the degree of p, and B is a bound on $\sum_{i=0}^{d} 2^i \beta_i^2$ (see Defn. 31).

Proof From Theorem 32 we have that for all $\mathcal{N} \in \mathcal{N}[\sigma, D, W, M]$, there is a network $\bar{\mathcal{N}} \in \mathcal{N}[p, D, W, M]$ such that

$$
\sup_{\mathbf{x} \in \mathbb{S}^{n-1}} \left|\mathcal{N}(\mathbf{x}) - \bar{\mathcal{N}}(\mathbf{x})\right| \leq \frac{\epsilon}{L+1}.
$$

Since the loss function ℓ is L-Lipschitz, this implies that

$$\ell(\bar{\mathcal{N}}(x), y) - \ell(\mathcal{N}(x), y) \leq L \cdot |\bar{\mathcal{N}}(x) - \mathcal{N}(x)| \leq \frac{L}{L+1} \cdot \epsilon. \tag{38}$$

Let $\mathcal{N}_{\min} = \arg\min_{\mathcal{N} \in \mathcal{N}[\sigma, D, W, M]} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\mathcal{N}(\mathbf{x}), y)]$. By the above, we get that there exists $\bar{\mathcal{N}}_{\min} \in \mathcal{N}[p, D, W, M]$ such that

$$\begin{aligned}
\min_{\bar{\mathcal{N}} \in \mathcal{N}[p,D,W,M]} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\bar{\mathcal{N}}(\mathbf{x}), y)] &\leq \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(\bar{\mathcal{N}}_{\min}(\mathbf{x}), y)] \\
&\leq \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\mathcal{N}_{\min}(\mathbf{x}), y)] + L \cdot \epsilon \\
&= \min_{\mathcal{N} \in \mathcal{N}[\sigma,D,W,M]} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\mathcal{N}(\mathbf{x}), y)] + \frac{L}{L+1} \cdot \epsilon.
\end{aligned}$$

Now from Zhang et al. (2016a, Theorem 1), we know that there exists an algorithm that outputs a predictor $\widehat{f}$ such that with probability at least $1 - \delta$ for any distribution $\mathcal{D}$

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\widehat{f}(\mathbf{x}), y)] \leq \min_{\bar{\mathcal{N}} \in \mathcal{N}[p,D,W,M]} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\bar{\mathcal{N}}(\mathbf{x}), y)] + \frac{\epsilon}{L+1}.$$

For loss functions that take on large values on the range of the predictor, we instead output the clipped version of the predictor $\mathsf{clip}(\widehat{f})$ in order to satisfy the requirements of the Rademacher bounds (as in Section 3).

The runtime of the algorithm is $\mathrm{poly}(n, (L+1)/\epsilon, \log(1/\delta), H^D(1))$, where $H(a) = \sqrt{\sum_{i=0}^{d} 2^i \beta_i a^{2i}}$, and $H^{(D)}$ is obtained by composing H with itself D times. By simple algebra, we conclude that $H^D(1)$ is bounded by $B^{O(d)^{D-1}}$.

Combining the above inequalities, we have

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\widehat{f}(\mathbf{x}), y)] \leq \min_{\mathcal{N} \in \mathcal{N}[D,W,M,\sigma_{\mathrm{relu}}]} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\mathcal{N}(\mathbf{x}), y)] + \epsilon.$$

This completes the proof. ■

We can now state the learnability result for ReLU networks as follows.

Corollary 34 (Learnability of ReLU Network) *There exists an algorithm that outputs a predictor $\widehat{f}$ such that with probability at least $1 - \delta$ for any distribution $\mathcal{D}$ over $\mathbb{S}^{n-1} \times [-M \cdot W, M \cdot W]$, and loss function ℓ which is convex, L-Lipschitz in the first argument, and b bounded on $[-2M \cdot W, 2M \cdot W]$,*

$$\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\widehat{f}(\mathbf{x}), y)] \leq \min_{\mathcal{N} \in \mathcal{N}[D,W,M,\mathrm{relu}]} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(\mathcal{N}(\mathbf{x}), y)] + \epsilon.$$

The time complexity of the above algorithm is bounded by $n^{O(1)} \cdot 2^{((L+1)\cdot M \cdot W^D \cdot D \cdot \epsilon^{-1})^D} \cdot \log(1/\delta)$.

The proof of the corollary follows from applying Theorem 33 for the activation function σ_{relu} since σ_{relu} is 1-Lipschitz and *low-weight approximable* (from Theorem 22 and 23). We obtain the following corollary specifically for depth-2 networks.

Corollary 35 *Depth-2 networks with k hidden units and activation function σ_{relu} such that the weight vectors have ℓ_2-norm bounded by 1 are agnostically learnable over $\mathbb{S}^{n-1} \times [-\sqrt{k}, \sqrt{k}]$ with respect to loss function ℓ which is convex, $O(1)$-Lipschitz in the first argument, and b bounded on $[-2\sqrt{k}, 2\sqrt{k}]$ in time $n^{O(1)} \cdot 2^{O(\sqrt{k}/\epsilon)} \cdot \log(1/\delta)$.*

The proof of the corollary follows from setting $L = O(1)$, $D = 1$, $M = 1$ and $W = \sqrt{k}$ in Theorem 34. $W = \sqrt{k}$ follows from bounding the ℓ_1-norm of the weights given a bound on the ℓ_2-norm.

We remark here that the above analysis holds for fully-connected networks with activation function $\sigma_{\mathrm{sig}}(x) = \frac{1}{1+e^{-x}}$ (Sigmoid function). Note that σ_{sig} is 1-Lipschitz. The following lemma due to Livni et al. (2014, Lemma 2) exhibits a low degree polynomial approximation for σ_{sig}. It is in turn based on a result of Shalev-Shwartz et al. (2011, Lemma 2).

Lemma 36 (Livni et al. (2014)) *For $\epsilon \in (0, 1)$, there exists a polynomial $p(a) = \sum_{i=1}^{d} \beta_i a^i$ for $d = O(\log(1/\epsilon))$ such that for all $a \in [-1, 1]$, $|p(a) - \sigma_{\mathrm{sig}}(a)| \leq \epsilon$.*

Let $p(a) = \sum_{i=1}^{d} \beta_i a^i$ be the uniform ϵ-approximation σ_{sig} which is guaranteed to exist by the above lemma. Using a similar trick as in Lemma 22, we can further bound $p([-1, 1]) \subseteq [0, 1]$. Also, using Lemma 23, we can show that $\sum_{i=0}^{d} 2^i \beta_i^2$ is bounded by $(1/\epsilon)^{O(1)}$. This shows that σ_{sig} is *low-weight approximable*.

Using Theorem 33, we state the following learnability result for depth-2 sigmoid networks.

Corollary 37 *Depth-2 networks with k hidden units and sigmoidal activation function such that the weight vectors have ℓ_2-norm bounded by 1 are agnostically learnable over $\mathbb{S}^{n-1} \times [-\sqrt{k}, \sqrt{k}]$ with respect to loss function ℓ which is convex, $O(1)$-Lipschitz in the first argument, and b bounded on $[-2\sqrt{k}, 2\sqrt{k}]$ in time $\mathrm{poly}(n, k, 1/\epsilon, \log(1/\delta))$.*

Observe that the above result is polynomial in all parameters. Livni et al. (cf. Livni et al. (2014, Theorem 5)) state an incomparable result for learning sigmoids: their runtime is superpolynomial in n for $L = \omega(1)$, where L is the bound on ℓ_1-norm of the weight vectors (L may be as large as $\sqrt{k}$ in the setting of Corollary 37). They, however, work over the Boolean cube (whereas we are working over the domain $\mathbb{S}^{n-1}$).

4.3. Application: Learning Parametric Rectified Linear Unit

A Parametric Rectified Linear Unit (PReLU) is a generalization of ReLU introduced by He et al. (2015). Compared to the ReLU, it has an additional parameter that is learned. Formally, it is defined as

Definition 38 (Parametric Rectifier) *The parametric rectifier (denoted by σ_{PReLU}) is an activation function defined as*

$$\sigma_{PReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ a \cdot x & \text{if } x < 0 \end{cases}$$

where a is a learnable parameter.

Note that we can represent $\sigma_{\mathrm{PReLU}}(x) = \mathsf{max}(0, x) - a \cdot \mathsf{max}(0, -x) = \sigma_{\mathrm{relu}}(x) - a \cdot \sigma_{\mathrm{relu}}(-x)$ which is a depth-2 network of ReLUs. Therefore, we can state the following learnability result for a single PReLU parameterized by a weight vector $\mathbf{w}$ based on learning depth-2 ReLU networks.

Corollary 39 *Let PReLU with the parameter a be such that $|a|$ is bounded by a constant and the weight vector $\mathbf{w}$ has 2-norm bounded by 1. Then, PReLU is agnostically learnable over $\mathbb{S}^{n-1}$ with respect to any $O(1)$-Lipschitz loss function in time $n^{O(1)} \cdot 2^{O(1/\epsilon)} \cdot \log(1/\delta)$.*

The proof of the corollary follows from setting $L = 1$, $D = 1$, $M = 1$ and $W = O(1)$ in Theorem 34.

The condition that $|a|$ be bounded by 1 is reasonable as in practice the value of a is very rarely above 1 as observed by He et al. (2015). Also note that Leaky-ReLUs (Maas et al., 2013) are PReLUs with fixed a (usually 0.01). Hence, we can agnostically learn them under the same conditions using an identical argument as above. Note that a network of PReLU can also be similarly learned as a ReLU by replacing each ReLU in the network by a linear combination of two ReLUs as described before.

4.4. Application: Learning the Piecewise Linear Transfer Function

Several functions have been used to relax the $0/1$ loss in the context of learning linear classifiers. The best example is the sigmoid function discussed earlier. Here we consider the piecewise linear transfer function. Formally, it is defined as

Definition 40 (Piecewise Linear Transfer Function) *The C-Lipschitz piecewise linear transfer function (denoted by σ_{pw}) is an activation function defined as*

$$\sigma_{pw}(x) = \mathsf{max}\left(0, \mathsf{min}\left(\frac{1}{2} + Cx, 1\right)\right).$$

Note that we can represent $\sigma_{\text{pw}}(x) = \mathsf{max}\left(0, \frac{1}{2} + Cx\right) - \mathsf{max}\left(0, -\frac{1}{2} + Cx\right) = \sigma_{\text{relu}}\left(\frac{1}{2} + Cx\right) - \sigma_{\text{relu}}\left(-\frac{1}{2} + Cx\right)$ which is a depth-2 network of ReLU. Therefore, we can state the following learnability result for a piecewise linear transfer function parameterized by weight vector $\mathbf{w}$ following a similar argument as in the previous section.

Corollary 41 *The class of C-Lipschitz piecewise linear transfer functions parametrized by weight vector $\mathbf{w}$ with 2-norm bounded by 1 is agnostically learnable over $\mathbb{S}^{n-1}$ with respect to any $O(1)$-Lipschitz loss function in time $n^{O(1)} \cdot 2^{O(C/\epsilon)} \cdot \log(1/\delta)$.*

The proof of the corollary follows from setting $L = 1$, $D = 1$, $M = 1$ and $W = O(L)$ in Theorem 34.

Shalev-Shwartz et al. (2011) in Appendix A solved the above problem for l_1 loss and gave a running time with dependence on C, ϵ as $\text{poly}\left(\exp\left(\frac{C^2}{\epsilon^2}\log\left(\frac{C}{\epsilon}\right)\right)\right)$. Our approach gives an exponential improvement in terms of $\frac{C}{\epsilon}$ and works for general constant Lipschitz loss functions.

4.5. Application: Convex Piecewise-Linear Fitting

In this section we can use our learnability results for networks of ReLUs to give polynomial-time approximation schemes for convex piecewise-linear regression (Magnani and Boyd, 2009). These problems have been studied in optimization and notably in machine learning in the context of Multivariate Adaptive Regression Splines (Friedman, 1991). Note that these are *not* the same as *univariate* piecewise or segmented regression problems, for which polynomial-time algorithms are known.

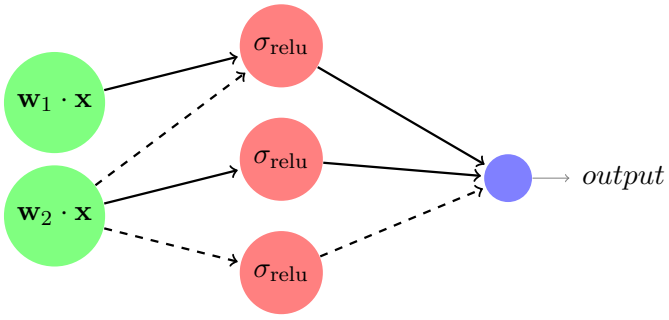

Figure 1: Representation of $\max(\mathbf{w}_1 \cdot \mathbf{x}, \mathbf{w}_2 \cdot \mathbf{x})$ as a depth-2 ReLU network. Note that solid edges represent a weight of 1, dashed edges represent a weight of -1, and the absence of an edge represents a weight of 0.

Although our algorithms run in time exponential in k (the number of affine functions), we note that no provably efficient algorithms were known prior to our work even for the case $k = 2$.[9]

The key idea will be to reduce piecewise regression problems to an optimization problem on networks of ReLUs using simple ReLU "gadgets." We formally describe the problems and describe the gadgets in detail.

4.5.1. SUM OF MAX 2-AFFINE

We start with a simple class of convex piecewise linear functions represented as a sum of a fixed number of functions where each of these functions is a maximum of 2 affine functions. This is formally defined as follows.

Definition 42 (Sum of k Max 2-Affine Fitting (Magnani and Boyd, 2009)) *Let $\mathcal{C}$ be the class of functions of the form $f(x) = \sum_{i=1}^{k} \max(\mathbf{w}_{2i-1} \cdot \mathbf{x}, \mathbf{w}_{2i} \cdot \mathbf{x})$ with $\mathbf{w}_1, \ldots, \mathbf{w}_{2k} \in \mathbb{S}^{n-1}$ mapping $\mathbb{S}^{n-1}$ to $\mathbb{R}$. Let $\mathcal{D}$ be an (unknown) distribution on $\mathbb{S}^{n-1} \times [-k, k]$. Given i.i.d. examples drawn from $\mathcal{D}$, for any $\epsilon \in (0, 1)$ find a function h (not necessarily in C) such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(h(\mathbf{x})-y)^2] \leq \min_{c\in\mathcal{C}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(c(\mathbf{x}) - y)^2] + \epsilon$.*

It is easy to see that $\max(a, b) = \max(0, a - b) + \max(0, b) - \max(0, -b) = \sigma_{\text{relu}}(a - b) + \sigma_{\text{relu}}(b) - \sigma_{\text{relu}}(-b)$ where $\sigma_{\text{relu}}(a) = \max(0, a)$. This is simply a linear combination of ReLUs. We can thus represent $\max(\mathbf{w}_1 \cdot \mathbf{x}, \mathbf{w}_2 \cdot \mathbf{x})$ as a depth-2 network (see Figure 1). Adding copies of this, we can represent a sum of k max 2-affine functions as a depth-2 network $\mathcal{N}_\Sigma$ with $3k$ hidden units and activation function σ_{relu} satisfying the following properties,

- $\|\mathbf{w}_j^{(0)}\| \leq 2$
- $\|\mathbf{w}_1^{(1)}\|_1 \leq 3k$
- Each input to each unit is bounded in magnitude by 2.

[9] Boyd and Magnani (Magnani and Boyd, 2009) specifically focus on the case of small k, writing "Our interest, however, is in the case when the number of terms k is relatively small, say no more than 10, or a few 10s."

The first property holds as $\|\mathbf{w}_j^{(0)}\| \leq \max(\|\mathbf{w}_{2j-1} - \mathbf{w}_{2j}\|, \|\mathbf{w}_{2j-1}\|, \|\mathbf{w}_{2j}\|) \leq \|\mathbf{w}_{2j-1}\| + \|\mathbf{w}_{2j}\| \leq 2$ using the triangle inequality. The second holds because each of the k max sub-networks contributes 3 to $\|\mathbf{w}_1^{(1)}\|_1$. The third is implied by the fact that each input to each unit is bounded by $|\max(\mathbf{w}_1 \cdot \mathbf{x}, -\mathbf{w}_1 \cdot \mathbf{x}, (\mathbf{w}_1 - \mathbf{w}_2) \cdot \mathbf{x})| \leq 2$.

Theorem 43 *Let $\mathcal{C}$ be as in Definition 42, then there is an algorithm $\mathcal{A}$ for solving sum of k max 2-affine fitting problem in time $n^{O(1)} 2^{O\left((k^2/\epsilon)\right)} \log(1/\delta)$.*

Proof As per our construction, we know that there exists a network $\mathcal{N}_\Sigma$ with activation function σ_{relu} and 1 hidden layer such that $\|\mathbf{w}_j^{(0)}\|_2 \leq 2$ and $\|\mathbf{w}_1^{(1)}\|_1 \leq 3k$. Also, input to each unit is bounded in magnitude by 2. Thus, using Theorem 34 with $K = 1$, $M = 2$ and $W = 3k$ we get that there exists an algorithm that solves the sum of k max 2-affine fitting problem in time $n^{O(1)} \cdot 2^{(O(k^2/\epsilon))} \cdot \log(1/\delta)$. ■

4.5.2. MAX k-AFFINE

In this section, we move to a more general convex piecewise linear function represented as the maximum of k affine functions. This is formally defined as follows.

Definition 44 (Max k-Affine Fitting (Magnani and Boyd, 2009)) *Let $\mathcal{C}$ be the class of functions of the form $f(x) = \max(\mathbf{w}_1 \cdot \mathbf{x}, \ldots, \mathbf{w}_k \cdot \mathbf{x})$ with $\mathbf{w}_1, \ldots, \mathbf{w}_n \in \mathbb{S}^{n-1}$ mapping $\mathbb{S}^{n-1}$ to $\mathbb{R}$. Let $\mathcal{D}$ be a distribution on $\mathbb{S}^{n-1} \times [-1, 1]$. Given i.i.d. examples drawn from $\mathcal{D}$, for any $\epsilon \in (0, 1)$ find a function h (not necessarily in C) such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(h(\mathbf{x}) - y)^2] \leq \min_{c\in\mathcal{C}} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[(c(\mathbf{x}) - y)^2] + \epsilon$.*

Note that this form is universal since any convex piecewise-linear function can be expressed as a max-affine function, for some value of k. However, we focus on bounded k and give learnability bounds in terms of k.

Observe that max k-affine can be expressed in a complete binary tree structure of height $\lceil \log k \rceil$ with a max operation at each unit and $\mathbf{w}_i \cdot \mathbf{x}$ for $i \in [k]$ at the k leaf units (for example, see Figure 2). Note that if k is not a power of 2, then we can trivially add leaves with value $\mathbf{w}_1 \cdot \mathbf{x}$ and make it a complete tree.

Thus, the class of convex piecewise linear functions can be expressed as a network of ReLUs with $\lceil \log k \rceil$ hidden layers by replacing each max unit in the tree by 3 ReLUs and adding an output unit. See Figure 3 for the construction for $k = 4$.

More formally, we have a network $\mathcal{N}_{\max}$ with $\lceil \log k \rceil$ hidden layers and one output unit with σ_{relu} as the activation function. Hidden layer i has $3 \cdot 2^{\lceil \log k \rceil - i}$ units. The weight vectors for the units in the first hidden layer are

$$\mathbf{w}_{3j-m}^{(0)} = \begin{cases} \mathbf{w}_{2j} - \mathbf{w}_{2j-1} & m = 0 \\ \mathbf{w}_{2j-1} & m = 1 \\ -\mathbf{w}_{2j-1} & m = 2 \end{cases}$$

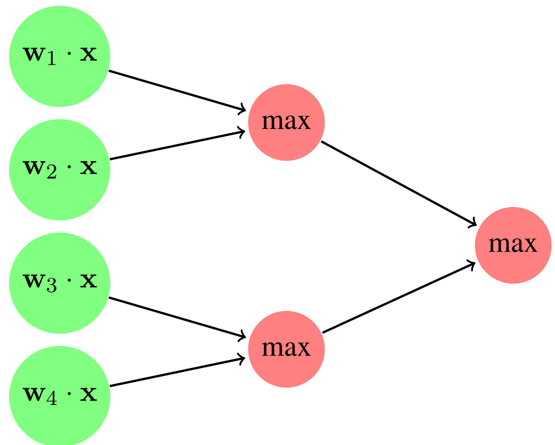

Figure 2: Tree structure for evaluating max k-affine with $k = 4$.

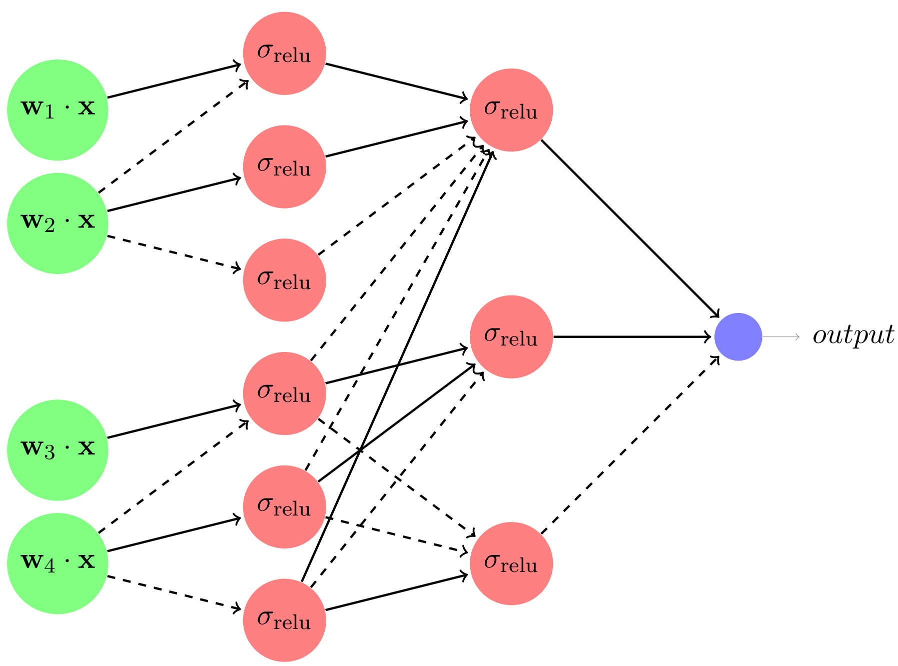

Figure 3: Network with σ_{relu} for evaluating max k-affine with $k = 4$. Note that solid edges represent a weight of 1, dashed edges represent a weight of -1 and the absence of an edge represents a weight of 0.

for $j \in [3 \cdot 2^{\lceil \log k \rceil - 1}]$. Further, the weight vectors input to hidden layer $i \in \{2, \ldots, \lceil \log k \rceil\}$ of the network are

$$\mathbf{w}_{3j-m}^{(i-1)} = \begin{cases} \mathbf{e}_{6j} + \mathbf{e}_{6j-1} - \mathbf{e}_{6j-2} - (\mathbf{e}_{6j-3} + \mathbf{e}_{6j-4} - \mathbf{e}_{6j-5}) & m = 0 \\ \mathbf{e}_{6j-3} + \mathbf{e}_{6j-4} - \mathbf{e}_{6j-5} & m = 1 \\ -(\mathbf{e}_{6j-3} + \mathbf{e}_{6j-4} - \mathbf{e}_{6j-5}) & m = 2. \end{cases}$$

for $j \in [3 \cdot 2^{\lceil \log k \rceil - i + 1}]$. Note, $\mathbf{e}_i$ refers to the vector with 1 at position i and 0 everywhere else. Finally the weight vector for the output unit is $w_1^{(\lceil \log k \rceil)} = \mathbf{e}_1 + \mathbf{e}_2 - \mathbf{e}_3$. The following properties of $\mathcal{N}_{\mathsf{max}}$ are easy to deduce.

- $\|\mathbf{w}_j^{(0)}\|_2 \leq 2$
- $\|\mathbf{w}_j^{(i)}\|_1 \leq 6$ for $i \in [\lceil \log k \rceil]$
- The input to each unit is bounded by 2.

Here, the first and third conditions are the same conditions as in the previous section. The second holds by the values of the weights defined above. Using the above construction, we obtain the following result.

Theorem 45 *Let $\mathcal{C}$ be as in Definition 44, then there is an algorithm $\mathcal{A}$ for solving the max k-affine fitting problem in time $n^{O(1)} \cdot 2^{O(k/\epsilon)^{\lceil \log k \rceil}} \cdot \log(1/\delta)$.*

Proof As per our construction, we know that there exists a network $\mathcal{N}_{\mathsf{max}}$ with activation function σ_{relu} and $\lceil \log k \rceil$ hidden layers such that $\|\mathbf{w}_j^{(0)}\|_2 \leq 2$ and $\|\mathbf{w}_j^{(i)}\|_1 \leq 6$ for $i \in [\lceil \log k \rceil]$. Also, input to each unit is bounded by 2. Thus, using Theorem 34 with $D = [\lceil \log k \rceil]$, $M = 2$ and $W = 6$, we get that there exists an algorithm that solves the max k-affine problem in the required time. ■

5. Hardness of Learning ReLU

We also establish the first hardness results for learning a *single* ReLU with respect to distributions supported on the Boolean hypercube ($\{0,1\}^n$). The high-level "takeaway" from our hardness results is that learning functions of the form $\mathsf{max}(0, \mathbf{w} \cdot \mathbf{x})$ where $|\mathbf{w} \cdot \mathbf{x}| \in \omega(1)$ is as hard as solving notoriously difficult problems in computational learning theory. This justifies our focus in previous sections on input distributions supported on $\mathbb{S}^{n-1}$ and indicates that learning real-valued functions on the sphere is one avenue for *avoiding* the vast literature of hardness results on Boolean function learning.

To begin, we recall the following problem from computational learning theory widely thought to be computationally intractable.

Definition 46 *(Learning Sparse Parity with Noise) Let $\chi_S : \{0,1\}^n \to \{0,1\}^n$ be an unknown parity function on a subset S, $|S| \leq k$, of n inputs bits (i.e., any input, restricted to S, with an odd number of ones is mapped to 1 and 0 otherwise). Let $\mathcal{C}_k$ be the concept class of all parity functions on subsets S of size at most k. Let $\mathcal{D}$ be a distribution on $\{0,1\}^n \times \{-1,1\}$ and define*

$$\mathsf{opt} = \min_{\chi \in \mathcal{C}_k} \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[\chi(\mathbf{x}) \neq y].$$

The Sparse Learning Parity with Noise *problem is as follows: Given i.i.d. examples drawn from $\mathcal{D}$, find h such that $\Pr_{(\mathbf{x},y)\sim\mathcal{D}}[h(\mathbf{x}) \neq y] \leq \mathsf{opt} + \epsilon$.*

Our hardness assumption is as follows:

Assumption 47 *For every algorithm $\mathcal{A}$ that solves the Sparse Learning Parity with Noise problem, there exists $\epsilon = O(1)$ and $k \in \omega(1)$ such that $\mathcal{A}$ requires time $n^{\Omega(k)}$.*

Any algorithm breaking the above assumption would be a major result in theoretical computer science. The best known algorithms due to Blum et al. (2003) and Valiant (2015) run in time $2^{O(n/\log n)}$ and $n^{0.8k}$, respectively. Under this assumption, we can rule out polynomial-time algorithms for reliably learning ReLUs on distributions supported on $\{0,1\}^n$.

Theorem 48 *Let $\mathcal{C}$ be the class of ReLUs over the domain $\mathcal{X} = \{0,1\}^n$ with the added restriction that $\|\mathbf{w}\|_1 \leq 2k$. Any algorithm $\mathcal{A}$ for reliably learning $\mathcal{C}$ in time $g(\epsilon) \cdot \mathsf{poly}(n)$ for any function g will give a polynomial time algorithm for learning sparse parities with noise of size k for $\epsilon = O(1)$.*

Proof We will show how to use a reliable ReLU learner to agnostically learn *conjunctions* on $\{0,1\}^n$ and use an observation due to Feldman and Kothari (2015) who showed that agnostically learning conjunctions is harder than the Sparse Learning Parity with Noise problem. Let $\mathcal{CO}_k$ be the concept class of all Boolean conjunctions of length at most k.

Notice that for the domain $\mathcal{X} = \{0,1\}^n$, the conjunction of literals $x_1, \ldots, x_k$ can be computed exactly as $\mathsf{max}(0,\ x_1 + \cdots + x_k - (k-1))$. Fix an arbitrary distribution $\mathcal{D}$ on $\{0,1\}^n \times \{0,1\}$ and define

$$\mathsf{opt} = \min_{c \in \mathcal{CO}_k} \Pr_{(\mathbf{x},y)\sim\mathcal{D}}[c(\mathbf{x}) \neq y].$$

Kalai et al. (2008, Theorem 5) observed that in order output a hypothesis h with error $\mathsf{opt} + \epsilon$ it suffices to minimize (to within ϵ) the following quantity:

$$\mathsf{opt}_1 = \min_{c \in \mathcal{CO}_k} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[|c(\mathbf{x}) - y|].$$

Consider the following transformed distribution $\mathcal{D}'$ on $\{0,1\}^n \times \{\epsilon, 1+\epsilon\}$ that adds a small positive ϵ to every y output by $\mathcal{D}$. Note that this changes opt_1 by at most ϵ. Further, all labels in $\mathcal{D}'$ are now positive. Since every $c \in \mathcal{CO}_k$ is computed exactly by a ReLU, and the reliable learning model demands that we minimize $\mathcal{L}_{>0}(h; \mathcal{D}')$ over all ReLUs, algorithm $\mathcal{A}$ will find an h such that $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}'}[|h(\mathbf{x}) - y|] \leq \mathsf{opt}_1 + \epsilon \leq \mathsf{opt} + 2\epsilon$. By appropriately rescaling ϵ, we have shown how to agnostically learn conjunctions using reliable learner $\mathcal{A}$. This completes the proof. ■

The above proof also shows hardness of learning ReLUs agnostically. Note the above hardness result holds if we require the learning algorithm to succeed on all domains where $|(w \cdot x)|$ can grow without bound with respect to n:

Corollary 49 *Let $\mathcal{A}$ be an algorithm that learns ReLUs on all domains $\mathcal{X} \subseteq \mathbb{R}^n$ where $(\mathbf{w} \cdot \mathbf{x})$ may take on values that are $\omega(1)$ with respect to the dimension n. Then any algorithm for reliably learning $\mathcal{C}$ in time $g(\epsilon) \cdot poly(n)$ will break the Sparse Learning Parity with Noise hardness assumption.*

Finally, we point out Kalai et al. (2012) proved that reliably learning conjunctions is also as hard as PAC Learning DNF formulas. Thus, by our above reduction, any efficient algorithm for reliably learning ReLUs would give an efficient algorithm for PAC learning DNF formulas (again this would be considered a breakthrough result in computational learning theory).

6. Conclusions and Open Problems

We have given the first set of efficient algorithms for ReLUs in a natural learning model. ReLUs are both effective in practice and, unlike linear threshold functions (halfspaces), admit non-trivial learning algorithms for *all* distributions with respect to adversarial noise. We "sidestepped" the hardness results in Boolean function learning by focusing on problems that are not entirely scale-invariant with respect to the choice of domain (e.g., reliably learning ReLUs). The obvious open question is to improve the dependence of our main result on $1/\epsilon$. We can handle $\epsilon = 1/\log n$, and as mentioned in the introduction, $\epsilon = 1/\mathsf{poly}(n)$ seems difficult. Is it possible to obtain a run-time of $\mathsf{poly}(n, k) \cdot 2^{O(1/\epsilon)}$ for depth-2 networks of ReLUs with k hidden units?

Acknowledgements. The authors are grateful to Sanjeev Arora and Roi Livni for helpful feedback and useful discussions on this work.

References

Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning sparse polynomial functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 500–510, 2014.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anribit Mukherjee. Understanding deep neural networks with rectified linear units, 2016. URL: https://arxiv.org/abs/1611.01491.

Francis Bach. Breaking the curse of dimensionality with convex neural networks. 2014.

Peter Bartlett, Daniel Kane, and Adam Klivans. Personal communication. 2017.

Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002. URL http://www.jmlr.org/papers/v3/bartlett02a.html.

Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *JACM: Journal of the ACM*, 50, 2003.

Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *STOC*, pages 105–117. ACM, 2016. ISBN 978-1-4503-4132-5. URL http://dl.acm.org/citation.cfm?id=2897518.

Ilias Diakonikolas, Daniel M. Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *FOCS*, pages 11–20. IEEE Computer Society, 2010. ISBN 978-0-7695-4244-7.

Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 907–940, 2016. URL http://jmlr.org/proceedings/papers/v49/eldan16.html.

Bassey Etim. Approve or Reject: Can You Moderate Five New York Times Comments? *The New York Times*, 2016. URL http://www.nytimes.com/interactive/2016/09/20/insider/approve-or-reject-moderation-quiz.html. Originally published September 20, 2016. Retrieved October 4, 2016.

V. Feldman, P. Gopalan, S. Khot, and A. K. Ponnuswami. On agnostic learning of parities, monomials, and halfspaces. *SIAM J. Comput*, 39(2):606–645, 2009. URL http://dx.doi.org/10.1137/070684914.

Vitaly Feldman and Pravesh Kothari. Agnostic learning of disjunctions on symmetric distributions. *Journal of Machine Learning Research*, 16:3455–3467, 2015.

Jerome H. Friedman. Multivariate adaptive regression splines. *Ann. Statist*, 1991.

David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015.

Sham M. Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. 2008.

Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008. doi: 10.1137/060649057.

Adam Tauman Kalai, Varun Kanade, and Yishay Mansour. Reliable agnostic learning. *Journal of Computer and System Sciences*, 78(5):1481–1495, 2012.

Kenji Kawaguchi. Deep learning without poor local minima. In Daniel D. Lee, Masashi Sugiyama, Ulrike V. Luxburg, Isabelle Guyon, and Roman Garnett, editors, *NIPS*, pages 586–594, 2016. URL http://papers.nips.cc/book/advances-in-neural-information-processing-systems-29-2016.

Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Mach. Learn.*, 17(2-3):115–141, 1994.

Adam Klivans and Pravesh Kothari. Embedding hard learning problems into gaussian space. In *RANDOM*, 2014.

Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci*, 75(1):2–12, 2009. URL http://dx.doi.org/10.1016/j.jcss.2008.07.008.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7533):436–444, May 2015.

Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer, 1991.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. pages 855–863, 2014. URL http://papers.nips.cc/paper/5267-on-the-computational-efficiency-of-training-neural-networks.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

Alessandro Magnani and Stephen P. Boyd. Convex piecewise-linear fitting. *Optimization and Engineering*, 10(1):1–17, 2009. ISSN 1573-2924. doi: 10.1007/s11081-008-9045-3. URL http://dx.doi.org/10.1007/s11081-008-9045-3.

James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

Donald J. Newman. Rational approximation to $|x|$. *Michigan Math. J.*, 11(1):11–14, 03 1964.

Phillippe Rigollet. *High-Dimensional Statistics*. MIT, 1st edition, 2015.

Hanie Sedghi and Anima Anandkumar. Provable methods for training neural networks with sparse connectivity. *arXiv preprint arXiv:1412.2693*, 2014.

Shai Shalev-Shwartz, Ohad Shamir, and Karthik Sridharan. Learning kernel-based halfspaces with the 0-1 loss. *SIAM J. Comput.*, 40(6):1623–1646, 2011.

Alexander A. Sherstov. Making polynomials robust to noise. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 747–758, New York, NY, USA, 2012. ACM.

Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, May 2015. doi: 10.1145/2728167. URL http://doi.acm.org/10.1145/2728167.

Wikipedia. Multinomial theorem — Wikipedia, the free encyclopedia, 2016a. URL: https://en.wikipedia.org/wiki/Multinomial_theorem.

Wikipedia. Polynomial kernel — Wikipedia, the free encyclopedia, 2016b. URL: https://en.wikipedia.org/wiki/Polynomial_kernel.

Yuchen Zhang, Jason D Lee, Martin J Wainwright, and Michael I Jordan. Learning halfspaces and neural networks with random initialization. *arXiv preprint arXiv:1511.07948*, 2015.

Yuchen Zhang, Jason Lee, and Michael Jordan. ℓ_1 networks are improperly learnable in polynomial-time. In *ICML*, 2016a.

Yuchen Zhang, Percy Liang, and Martin J Wainwright. Convexified convolutional neural networks. *arXiv preprint arXiv:1609.01000*, 2016b.