Proceedings of Machine Learning Research vol 99:1–30, 2019 32nd Annual Conference on Learning Theory

Learning Neural Networks with Two Nonlinear Layers in Polynomial Time

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

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

Editors: Alina Beygelzimer and Daniel Hsu

Abstract

We give a polynomial-time algorithm for learning neural networks with one layer of sigmoids feeding into any Lipschitz, monotone activation function (e.g., sigmoid or ReLU). The algorithm succeeds with respect to *any* distribution on the unit ball in n dimensions (hidden weight vectors in the first layer have unit norm). This is the first efficient algorithm for learning a general class of neural networks with more than one nonlinear layer that makes no restrictions on the VC-dimension of the network.

Algorithms for learning relaxations of our model (e.g., allowing larger weight vectors in the first layer) would lead to breakthroughs on notoriously hard problems in Boolean function learning. Thus, our results are "best possible" with respect to current techniques.

Our algorithm– *Alphatron*– is an iterative update rule that combines isotonic regression with kernel methods. We use this algorithm to give a simple reduction for translating PAC learning algorithms to the more general, real-valued setting of *probabilistic concepts*, a model that (unlike PAC learning) requires non-i.i.d. noise-tolerance. This substantially improves many longstanding results for PAC learning Boolean functions.

Keywords: neural networks, efficient algorithm, isotonic regression, kernel methods, probabilistic concepts

1. Introduction

Giving provably efficient algorithms for learning neural networks is a fundamental challenge in the theory of machine learning. Most work in computational learning theory has led to negative results showing that– from a worst-case perspective– even learning the simplest architectures seems computationally intractable (Livni et al. (2014); Song et al. (2017)). For example, there are known hardness results for agnostically learning a single ReLU (learning a ReLU in the non-realizable setting) (Goel et al. (2016)).

As such, much work has focused on finding algorithms that succeed after making various restrictive assumptions on both the network's architecture and the underlying marginal distribution. Recent work gives evidence that for gradient-based algorithms these types of assumptions are actually necessary (Shamir (2016)). In this paper, we focus on understanding the frontier of efficient neural network learning: what is the most expressive class of neural networks that can be learned, provably, in polynomial-time without taking any additional assumptions?

Our Results. We give a polynomial-time algorithm that learns neural networks with one layer of sigmoids feeding into any smooth, monotone activation function (for example, sigmoid or ReLU). Both the first hidden layer of sigmoids and the output activation function have corresponding hidden weight vectors. The algorithm succeeds with respect to any distribution on the unit ball in n dimensions. The network can have an arbitrary feedforward structure, and we assume nothing about these weight vectors other than boundedness: each weight vector has 2-norm at most one in the first layer (the weight vectors in the second layer may have polynomially large norm). These networks, even over the unit ball, have polynomially large VC dimension (if the first layer has m hidden units, the VC dimension will be $\Omega(m)$ (Lee et al. (1994)).

Moving beyond one nonlinear layer has been a challenging problem in the theory of machine learning: techniques for learning one nonlinear layer in polynomial-time seem to break down when even a single nonlinear output is added. Although some kernel methods can be composed to handle higher depths, strong assumptions are then required on the architecture, restricting the expressiveness/VC-dimension of the network.

While our techniques "only" handle one additional nonlinear output layer, it is this additional layer that allows us to give broad generalizations of many well-known results in computational learning theory including well-studied function classes such as DNF formulas, constant-depth circuits, and functions of halfspaces. We extend known PAC learning results for these Boolean function classes to the more general setting of real-valued *probabilistic concepts*. One immediate implication is the *first set of algorithms for non-iid noise tolerance for all of the above function classes.*

Further, we show that efficient algorithms for learning more general networks than what we consider here (for example by allowing larger norm weight vectors in the first layer) would give breakthrough results for notoriously difficult problems in Boolean function learning. In particular, we show that learning one layer of $\omega(1)$ ReLUs in polynomial time would yield an $f(k)\mathsf{poly}(n)$-time algorithm for learning k-juntas, a notoriously hard problem in theoretical computer science for which the best known running time is currently $n^{0.8k}$ due to Valiant (2015). Additionally we observe a superpolynomial SQ-lower bound for learning linear combinations of $\omega(1)$ ReLUs, which rules out essentially all known efficient learning algorithms (with the exception of learning parities). We obtain similar results for learning a single layer of sigmoids with polynomially large weights.

Technical Contributions. At the core of our algorithm, which we call *Alphatron*, is a combination of kernel methods with an additive update rule inspired by work in isotonic regression. The algorithm outputs a hypothesis that gives efficient oracle access to interpretable features. That is, if the output activation function is u, Alphatron constructs a hypothesis of the form $u(f(\mathbf{x}))$ where f is an implicit encoding of products of features from the instance space, and f yields an efficient algorithm for random access to the coefficients of these products.

The algorithm is reminiscent of Boosting, but it is simpler, and does not require re-weighting distributions or adding noise to labels. It is also very versatile and can be used to derive many new results for efficiently learning function classes:

- Let $c(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be any feedforward neural network with one hidden layer of sigmoids of size k feeding into any activation function u that is monotone and L-Lipschitz. Given independent draws $(\mathbf{x}, y)$ from $\mathbb{S}^{n-1} \times [0,1]$ with $\mathbb{E}[y|\mathbf{x}] = c(\mathbf{x})$, we obtain an efficiently computable hypothesis $u(f(\mathbf{x}))$ such that $\mathbb{E}[(c(\mathbf{x}) - u(f(\mathbf{x})))^2] \leq \epsilon$ with running time and sample complexity $\mathsf{poly}(n, k, 1/\epsilon, L)$ (the algorithm succeeds with high probability). Note that the related (but incomparable) problem of distribution-free PAC learning intersections of halfspaces is cryptographically hard (Klivans and Sherstov (2009)).

- With an appropriate choice of kernel function, we show that Alphatron can learn more general, real-valued versions of well-studied Boolean concept classes in the probabilistic concept model due to Kearns and Schapire. We subsume and improve known algorithms for uniform distribution learning of DNF formulas (queries), majorities of halfspaces, majorities of AC^0 circuits, and submodular functions, among others. We achieve the first non-i.i.d. noise-tolerant algorithms[1] for learning these classes[2]. Our technical contributions include
  - Extending the KM algorithm for finding large Fourier coefficients (Kushilevitz and Mansour (1993)) to the setting of probabilistic concepts. For the uniform distribution on the hypercube, we can combine a "KMtron" algorithm with a projection operator to learn smooth, monotone combinations of L_1-bounded functions. This is the first extension of Jackson's celebrated "Harmonic Sieve" algorithm for learning DNFs to a non-iid noise model and improves the approach of Gopalan et al. (2008) for agnostically learning decision trees.
  - Generalizing the "low-degree" algorithm due to Linial et al. (1993) to show that for any circuit class that can be approximated by low-degree Fourier polynomials, we can learn monotone combinations of these circuits in the probabilistic concept model. This gives the most general result known for learning monotone functions of many circuit classes including constant depth circuits and halfspaces.
  - Using low-weight (as opposed to just low-degree) polynomial approximators for intersections of halfspaces with a (constant) margin to obtain the *first polynomial-time* algorithms for learning smooth, monotone combinations (intersection is a special case). The previous best result was a quasipolynomial-time algorithm for PAC learning the special case of ANDs of halfspaces with a (constant) margin (Klivans and Servedio (2008)).

We learn specifically with respect to square loss, though this will imply polynomial-time learnability for most commonly studied loss functions. When the label Y is a deterministic Boolean function of X, it is easy to see that small square loss will imply small $0/1$ loss.

High-Level Overview of Our Approach. Our starting point is the Isotron algorithm, due to Kalai and Sastry (2009), and a refinement due to Kakade et al. (2011) called the GLMtron. These algorithms efficiently learn any generalized linear model (GLM): distributions on instance-label pairs $(\mathbf{x}, y)$ where the conditional mean of y given $\mathbf{x}$ is equal to $u(\mathbf{w} \cdot \mathbf{x})$ for some (known) smooth, non-decreasing function u and unknown weight vector $\mathbf{w}$. Their algorithms are simple and use an iterative update rule to minimize square-loss, a non-convex optimization problem in this setting. They remark that their algorithms can be kernelized, but no concrete applications are given.

Around the same time, Shalev-Shwartz et al. (2011) used kernel methods and general solvers for convex programs to give algorithms for learning a halfspace under a distributional assumption corresponding to a margin in the non-realizable setting (agnostic learning). Their kernel was composed

1. Previously these classes were known to be learnable in the presence of classification noise, where each is label is flipped independently with some fixed probability.
2. Non-iid/agnostic noise tolerance was known for majorities of halfspaces only for $\epsilon < 1/k^2$, where k is the number of halfspace (Kalai et al. (2008)).

by Zhang et al. (2016) to obtain results for learning sparse neural networks with certain smooth activations, and Goel et al. (2016) used a similar approach in conjunction with general tools from approximation theory to obtain learning results for a large class of nonlinear activations including ReLU and Sigmoid.

Our main idea is to simply combine the above approaches: implicitly embed the first layer of nonlinear neurons into some higher dimensional Hilbert space, and apply a (kernelized) version of GLMtron to obtain an algorithm for learning the function u composed with this implicit embedding. In retrospect, this is a natural algorithm to analyze, and we show that it leads to improvements on many well-studied problems in learning theory.

One technical issue is that the first layer is now *approximated* by a function in some high dimensional space, and the isotonic regression algorithm must account for this error. As such, we introduce a learning rate and a slack variable to obtain an appropriate convergence rate and follow the outline of the analysis of GLMtron. For the reader familiar with kernel methods, the resulting algorithm is similar to performing gradient descent on the support vectors of a target element in a reproducing kernel hilbert space (RKHS). We can then carefully apply a particular polynomial-based kernel along with several results from approximation theory to give our main results.

A key maneuver is to work in the probabilistic concept model of learning, where the noise, while not iid, must be mean-zero. Prior work on kernel methods for learning neural networks has focused almost exclusively on learning in the agnostic model. This model is *too* challenging, in the sense that the associated optimization problems seem computationally intractable (even for a single ReLU). Interestingly, working in the probabilistic concept model allows us to avoid regularization, which would be the standard approach for kernel methods such as kernelized ridge regression.

On Bounding the 2-Norm. Our algorithms have an exponential dependence on the 2-norm of the weight vectors in the first layer (although the dependence on the dimension is a fixed polynomial). Still, networks with unit norm weight vectors with respect to distributions over the unit sphere have polynomially large VC dimension (if the first layer has m hidden units, the VC dimension will be $\Omega(m)$ (Lee et al. (1994))). In addition, as we discuss in Section 3, relaxing this norm bound leads to hard problems in Boolean function learning.[3]

Bounding the norm of the weight vectors also aligns nicely with practical tools for learning neural networks. Most gradient-based training algorithms for learning deep nets initialize hidden weight vectors to have unit norm and use techniques such as batch normalization or regularization to prevent the norm of the weight vectors from becoming large.

Related Work. The literature on provably efficient algorithms for learning neural networks is extensive. In this work we focus on common nonlinear activation functions: sigmoid, ReLU, or threshold. For linear activations, neural networks compute an overall function that is linear and can be learned efficiently using any polynomial-time algorithm for solving linear regression. Livni et al. (2014) observed that neural networks of constant depth with constant degree polynomial activations are equivalent to linear functions in a higher dimensional space (polynomials of degree d are equivalent to linear functions over n^d monomials). It is known, however, that any polynomial that computes or even ϵ-approximates a single ReLU requires degree $\Omega(1/\epsilon)$ (Goel et al. (2016)). Thus, linear methods alone do not suffice for obtaining our results.

3. In the *agnostic* model, learning even a single ReLU– *even with a bounded norm weight vector*– with respect to any distribution on the unit sphere is as hard as learning sparse parity with noise (Goel et al. (2016)). As such, for *distribution-free* learnability, it is necessary to have some bound on the norm *and* some structure in the noise model.

The vast majority of recent work on learning neural networks applies to neural networks with one nonlinear layer under various assumptions. Works that fall into these categories include (Klivans et al. (2004); Klivans and Meka (2013); Janzamin et al. (2015); Sedghi and Anandkumar (2014); Zhang et al. (2017, 2016); Zhong et al. (2017); Goel and Klivans (2017); Du et al. (2017); Goel et al. (2018); Ge et al. (2017); Soltanolkotabi (2017); Gao et al. (2018); Ge et al. (2018)). Goel et al. (2016) used kernel methods to give an efficient, agnostic learning algorithm for sums of sigmoids (i.e., one hidden layer of sigmoids) with respect to any distribution on the unit ball for hidden weight vectors with unit norm. A relevant work is due to Daniely (2017) who used kernel methods to learn neural networks, but his techniques require strong norm bounds in the corresponding Hilbert space and give results only for networks with restricted VC dimension. In some sense, using tools from isotonic regression allows us to learn one extra nonlinear layer with kernel methods "for free" without a dramatic blowup in of the corresponding Hilbert space norm.

Notation. Vectors are denoted by bold-face and $||\cdot||$ denotes the standard 2-norm of the vector. We denote the space of inputs by $\mathcal{X}$ and the space of outputs by $\mathcal{Y}$. In our paper, $\mathcal{X}$ is usually the unit sphere/ball and $\mathcal{Y}$ is $[0,1]$ or $\{0,1\}$. Standard scalar (dot) products are denoted by $\mathbf{a}\cdot\mathbf{b}$ for vectors $\mathbf{a},\mathbf{b}\in\mathbb{R}^n$, while inner products in a Hilbert space are denoted by $\langle\mathbf{a},\mathbf{b}\rangle$ for elements $\mathbf{a},\mathbf{b}$ in the Hilbert space. We denote the standard composition of functions f_1 and f_2 by $f_1\circ f_2$.

We assume the reader has a basic working knowledge of kernel methods. For a good resource on kernel methods in machine learning we refer the reader to Schölkopf and Smola (2002). We denote a kernel function by $\mathcal{K}:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$. In this paper we restrict to kernels which admit a feature map ψ and corresponding Hilbert space $\mathcal{H}$ equipped with an inner product such that $\mathcal{K}(\mathbf{x},\mathbf{x}')=\langle\psi(\mathbf{x}),\psi(\mathbf{x}')\rangle$ where ψ. Intuitively, the kernel function computes a "similarity score" of two inputs in some higher dimensional space.

2. The Alphatron Algorithm

Here we present our main algorithm Alphatron (Algorithm 1) and a proof of its correctness. Fix a joint distribution $\mathcal{D}$ on $X\times Y$ and recall that our goal is to learn a function $u(f(X))$ where $f(X)$ is the conditional mean of the underlying distribution (i.e., $\mathbb{E}[Y|X]$). The function f is assumed to have a corresponding bounded-norm element in some high-dimensional Hilbert space that approximates f in ℓ_2. Our algorithm keeps track of a candidate set of weights $\{\alpha_i\}_{i=1}^m$ corresponding to the "support vectors" of f (i.e., an implicit representation of f). Given a draw from $\mathcal{D}$, the weights are updated additively in proportion to how far our current hypothesis is from predicting the correct value. Kernel perceptron is a special case of our algorithm when u is the identity function, there is no noise, and $0/1$ loss is used instead of square loss.

We define the error of hypothesis h in two ways: $\varepsilon(h)=\mathbb{E}_{\mathbf{x},y}[(h(\mathbf{x})-\mathbb{E}[y|\mathbf{x}])^2]$ and $err(h)=\mathbb{E}_{\mathbf{x},y}[(h(\mathbf{x})-y)^2]$. Our objective is to minimize ε. It is easy to see that $\varepsilon(h)=err(h)-err(\mathbb{E}[y|\mathbf{x}])$ thus minimizing ε is equivalent to minimizing err. Let us denote $\widehat{\varepsilon},\widehat{err}$ to be the empirical versions (average taken over a sample) of the two errors. The following theorem generalizes Theorem 1 of Kakade et al. (2011) to the bounded noise setting for a high dimensional feature space. We follow the same outline, and their theorem can be recovered by setting $\psi(\mathbf{x})=\mathbf{x}$ and ξ as the zero function.

Theorem 1 *Let $\mathcal{K}$ be a kernel function corresponding to feature map ψ such that $\forall\boldsymbol{x}\in\mathcal{X}, ||\psi(\boldsymbol{x})||\leq 1$. Consider samples $(\boldsymbol{x_i},y_i)_{i=1}^m$ drawn iid from distribution $\mathcal{D}$ on $\mathcal{X}\times[0,1]$ such that $E[y|\boldsymbol{x}]=$*

Algorithm 1 Alphatron

```
Input : data $\langle(\mathbf{x_i}, y_i\rangle_{i=1}^m \in \mathbb{R}^n \times [0,1]$, non-decreasing[4] $L$-Lipschitz function $u : \mathbb{R} \to [0,1]$, kernel
        function $\mathcal{K}$ corresponding to feature map $\psi$, learning rate $\lambda > 0$, number of iterations $T$,
        held-out data of size $N$ $\langle\mathbf{a_j}, b_j\rangle_{j=1}^N \in \mathbb{R}^n \times [0,1]$
$\alpha^1 := 0 \in \mathbb{R}^m$
for $t = 1, \ldots, T$ do
    $h^t(\mathbf{x}) := u(\sum_{i=1}^m \alpha_i^t \mathcal{K}(\mathbf{x}, \mathbf{x_i}))$ for $i = 1, 2, \ldots, m$ do
        $\alpha_i^{t+1} := \alpha_i^t + \frac{\lambda}{m}(y_i - h^t(\mathbf{x_i}))$
    end
end
Output: $h^r$ where $r = \arg\min_{t\in\{1,\ldots,T\}} \sum_{j=1}^N (h^t(\mathbf{a_j}) - b_j)^2$
```

$u(\langle \boldsymbol{v}, \psi(\boldsymbol{x})\rangle + \xi(\boldsymbol{x}))$ where $u : \mathbb{R} \to [0,1]$ is a known L-Lipschitz non-decreasing function, $\xi : \mathbb{R}^n \to [-M, M]$ for $M > 0$ such that $\mathbb{E}[\xi(\boldsymbol{x})^2] \le \epsilon$ and $||\boldsymbol{v}|| \le B$. Then for $\delta \in (0,1)$, with probability $1-\delta$, Alphatron with $\lambda = 1/L, T = CBL\sqrt{m/\log(1/\delta)}$ and $N = C'm\log(T/\delta))$ for large enough constants $C, C' > 0$ outputs a hypothesis h such that,

$$\varepsilon(h) \le O\left(L\sqrt{\epsilon} + LM\sqrt[4]{\frac{\log(1/\delta)}{m}} + BL\sqrt{\frac{\log(1/\delta)}{m}}\right).$$

Alphatron runs in time $\mathsf{poly}(n, m, \log(1/\delta), t_{\mathcal{K}})$ where $t_{\mathcal{K}}$ is the time required to compute the kernel function $\mathcal{K}$.

2.1. General Theorems Involving Alphatron

In this section we use Alphatron to give our most general learnability results for the probabilistic concept (p-concept) model. We then state several applications in the next section. Here we show that if a function can be approximated by an element of an appropriate Hilbert space (referred to as a reproducing kernel Hilbert space or RKHS), then it is p-concept learnable. We assume that the kernel function is efficiently computable, that is, computable in polynomial time in the input dimension. Formally, we define approximation as follows:

Definition 2 ((ϵ, B, M)-approximation) *Let f be a function mapping domain $\mathcal{X}$ to $\mathbb{R}$ and $\mathcal{D}$ be a distribution over $\mathcal{X}$. Let $\mathcal{K}$ be a kernel function with corresponding RKHS $\mathcal{H}$ and feature vector ψ. We say f is (ϵ, B, M)-approximated by $\mathcal{K}$ over $\mathcal{D}$ if there exists some $\boldsymbol{v} \in \mathcal{H}$ with $||\boldsymbol{v}|| \le B$ such that for all $\boldsymbol{x} \in \mathcal{X}$, $|f(\boldsymbol{x}) - \langle \boldsymbol{v}, \psi(\boldsymbol{x})\rangle| \le M$ and $\mathbb{E}[(f(\boldsymbol{x}) - \langle \boldsymbol{v}, \psi(\boldsymbol{x})\rangle)^2] \le \epsilon^2$.*

Combining Alphatron and the above approximation guarantees, we have the following general theorem:

Theorem 3 *Consider distribution $\mathcal{D}$ on $\mathcal{X} \times [0,1]$ such that $\mathbb{E}[y|\boldsymbol{x}] = u(f(\boldsymbol{x}))$ where u is a known L-Lipschitz non-decreasing function and f is (ϵ, B, M)-approximated over $\mathcal{D}_{\mathcal{X}}$ by some kernel function K such that $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \le 1$. Then for $\delta \in (0,1)$, there exists an algorithm that draws m*

4. We present the algorithm and subsequent results for non-decreasing function u. Non-increasing functions can also be handled by negating the update term, i.e., $\alpha_i^{t+1} := \alpha_i^t - \frac{\lambda}{m}(y_i - h^t(\mathbf{x_i}))$.

iid samples from $\mathcal{D}$ and outputs a hypothesis h such that with probability $1-\delta$, $\varepsilon(h) \leq O(L\epsilon)$ for $m = O\left(\left(\frac{LM}{\epsilon}\right)^4 + \left(\frac{BL}{\epsilon}\right)^2\right) \cdot \log(1/\delta)$ in time $\mathsf{poly}(n, B, M, L, 1/\epsilon, \log(1/\delta))$ where n is the dimension of $\mathcal{X}$.

Proof Let $\mathcal{H}$ be the RKHS corresponding to $\mathcal{K}$ and ψ be the feature vector. Since f is (ϵ, B, M)-approximated by kernel function $\mathcal{K}$ over $\mathcal{D}_{\mathcal{X}}$, we have $\forall$ $\mathbf{x}, f(\mathbf{x}) = \langle \mathbf{v}, \psi(\mathbf{x})\rangle + \xi(\mathbf{x})$ for some function $\xi : \mathcal{X} \to [-M, M]$ with $\mathbb{E}[\xi(\mathbf{x})^2] \leq \epsilon^2$. Thus $E[y|\mathbf{x}] = u(f(\mathbf{x})) = u\left(\langle \mathbf{v}, \psi(\mathbf{x})\rangle + \xi(\mathbf{x})\right)$. Applying Theorem 1, we have that Alphatron outputs a hypothesis h such that

$$\varepsilon(h) \leq CL\left(\epsilon + M\sqrt[4]{\frac{\log(1/\delta)}{m}} + B\sqrt{\frac{\log(1/\delta)}{m}}\right)$$

for some constants $C > 0$. Also Alphatron requires at most $O(BL\sqrt{m/\log(1/\delta)})$ iterations. Setting m as in theorem statement gives us the required result. ■

For the simpler case when f is *uniformly* approximated by elements in the RKHS we have,

Definition 4 ((ϵ, B)-uniform approximation) *Let f be a function mapping domain $\mathcal{X}$ to $\mathbb{R}$ and $\mathcal{D}$ be a distribution over $\mathcal{X}$. Let $\mathcal{K}$ be a kernel function with corresponding RKHS $\mathcal{H}$ and feature vector ψ. We say f is (ϵ, B)-uniformly approximated by $\mathcal{K}$ over $\mathcal{D}$ if there exists some $\mathbf{v} \in \mathcal{H}$ with $||\mathbf{v}|| \leq B$ such that for all $\mathbf{x} \in \mathcal{X}$, $|f(\mathbf{x}) - \langle \mathbf{v}, \psi(\mathbf{x})\rangle| \leq \epsilon$.*

Theorem 5 *Consider distribution $\mathcal{D}$ on $\mathcal{X} \times [0,1]$ such that $\mathbb{E}[y|\mathbf{x}] = u(f(\mathbf{x}))$ where u is a known L-Lipschitz non-decreasing function and f is (ϵ, B)-approximated by some kernel function $\mathcal{K}$ such that $\mathcal{K}(\mathbf{x}, \mathbf{x}') \leq 1$. Then for $\delta \in (0,1)$, there exists an algorithm that draws m iid samples from $\mathcal{D}$ and outputs a hypothesis h such that with probability $1-\delta$, $\varepsilon(h) \leq O(L\epsilon)$ for $m \geq \left(\frac{BL}{\epsilon}\right)^2 \cdot \log(1/\delta)$ in time $\mathsf{poly}(n, B, L, 1/\epsilon, \log(1/\delta))$ where n is the dimension of $\mathcal{X}$.*

Proof The proof is the same as the proof of Theorem 3 by re-examining the proof of Theorem 1 and noticing that $\xi(\mathbf{x})$ is uniformly bounded by ϵ in each inequality. ■

3. Learning Neural Networks

In this section we give polynomial time learnability results for neural networks with two nonlinear layers in the p-concept model. Following Safran and Shamir (2016), we define a neural network with one (nonlinear) layer with k units as $\mathcal{N}_1 : \mathbf{x} \to \sum_{i=1}^k \mathbf{b}_i \sigma(\mathbf{a_i} \cdot \mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{a_i} \in \mathbb{S}^{n-1}$ for $i \in \{1, \cdots, k\}$, $\mathbf{b} \in \mathbb{S}^{k-1}$. We subsequently define a neural network with two (nonlinear) layers with one unit in layer 2 and k units in hidden layer 1 as $\mathcal{N}_2 : \mathbf{x} \to \sigma'\left(\mathcal{N}_1(x)\right) = \sigma'\left(\sum_{i=1}^k \mathbf{b}_i \sigma(\mathbf{a_i} \cdot \mathbf{x})\right)$ for $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{a_i} \in \mathbb{S}^{n-1}$ for $i \in \{1, \cdots, k\}$, $\mathbf{b} \in \mathbb{S}^{k-1}$ and $\sigma, \sigma' : \mathbb{R} \to \mathbb{R}$.

Goel et al. (2016) showed that one nonlinear layer neural networks can be uniformly approximated by multivariate polynomials with bounded degree and weight for activations such as ReLU and sigmoid (see Appendix A.4). These multivariate polynomials can then be represented as bounded-norm elements in the Hilbert space for the multinomial kernel (see Definition 24; for technical reasons we must use the multinomial kernel to take advantage of low-weight rather than low-degree approximations). Combining this observation with Alphatron (Theorem 5), we obtain our main result for learning classes of neural networks with two nonlinear layers in polynomial time:

Theorem 6 *Consider samples $(\boldsymbol{x_i}, y_i)_{i=1}^m$ drawn iid from distribution $\mathcal{D}$ on $\mathbb{S}^{n-1} \times [0,1]$ such that $E[y|\boldsymbol{x}] = \mathcal{N}_2(\boldsymbol{x})$ with $\sigma' : \mathbb{R} \to [0,1]$ is a known L-Lipschitz non-decreasing function and $\sigma = \sigma_{sig}$ is the sigmoid function. There exists an algorithm that outputs a hypothesis h such that, with probability $1-\delta$,*

$$\mathbb{E}_{\boldsymbol{x},y\sim\mathcal{D}}\left[(h(\boldsymbol{x}) - \mathcal{N}_2(\boldsymbol{x}))^2\right] \leq \epsilon$$

for $m = \left(\frac{kL}{\epsilon}\right)^{O(1)} \cdot \log(1/\delta)$. The algorithm runs in time polynomial in m and n.

We also obtain results for networks of ReLUs (see Theorem 37), but the dependence on the number of hidden units, ϵ, and L are exponential (the algorithm still runs in polynomial-time in the dimension). This dependence arises due to the fact that a $\Omega(1/\epsilon)$ degree polynomial is required for approximating a single ReLU.
Remark. Although our algorithm does not recover the parameters of the network, it still outputs a hypothesis with interpretable features. Refer to Appendix D for more details.

4. Generalizing PAC Learning to Probabilistic Concepts

In this section we show how known algorithms for PAC learning boolean concepts can be generalized to the probabilistic concept model. We use Alphatron to learn real-valued versions of these well-studied concepts.
Notation. We follow the notation of Gopalan et al. (2008). For any function $P : \{-1,1\}^n \to \mathbb{R}$, we denote the Fourier coefficients by $\widehat{P}(S)$ for all $S \subseteq [n]$. The support of P, i.e., the number of non-zero Fourier coefficients, is denoted by $\mathsf{supp}(P)$. The norms of the coefficient vectors are defined as $L_p(P) = \left(\sum_S |\widehat{P}(S)|^p\right)^{1/p}$ for $p \geq 1$ and $L_\infty(P) = \max_S |\widehat{P}(S)|$. Similarly, the norm of the function P are defined as $||P||_p = \mathbb{E}_{x\in\{-1,1\}^n}\left[\sum_S |P(x)|^p\right]^{1/p}$ for $p \geq 1$. Also, the inner product $P \cdot Q = \mathbb{E}_{x\in\{-1,1\}^n}[P(x)Q(x)]$.

4.1. Generalized DNF Learning with Queries

Here we give an algorithm, KMtron, which combines isotonic regression with the KM algorithm (Kushilevitz and Mansour (1993)) for finding large Fourier coefficients of a function (given query access to the function). The main application of KMtron is a generalization of celebrated results for PAC learning DNF formulas (Jackson (1997)) to the setting of probabilistic concepts. That is, we can efficiently learn any conditional mean function that is a smooth, monotone combination of L_1-bounded functions.

In KMtron (Algorithm 2), the KM algorithm takes the place of the "kernel trick" used by Alphatron to provide an estimate for the update step in isotonic regression. Viewed this way, the KM algorithm can be re-interpreted as a query-algorithm for giving estimates of the gradient of square-loss with respect to the uniform distribution on Boolean inputs.

Note that $\mathsf{proj}_K(P)$ for $P : \{-1,1\}^n \to \mathbb{R}$ maps P to the closest Q in convex set $K = \{Q : \{-1,1\}^n \to \mathbb{R} \mid L_1(Q) \leq k\}$, i.e., $\mathsf{proj}_K(P) = \arg\min_{Q\in K} ||Q-P||_2$. This is necessary to keep the L_1 weight of the polynomial to be bounded. To efficiently run KMtron, we require efficient query access to $u \circ P - u \circ P_{t-1}$. Since P_{t-1} is stored as a sparse polynomial, and we are given query access for $u \circ P$, we can efficiently compute $u(P(x)) - u(P_{t-1}(x))$ for any x. We can extend the algorithm to handle distribution queries (p-concept), i.e., for any x of our choosing we obtain

Algorithm 2 KMtron

```
Input  : Function $u : \mathbb{R} \to [0,1]$ non-decreasing and $L$-Lipschitz, query access to $u \circ P$ for some
         function $P : \{-1,1\}^n \to \mathbb{R}$, learning rate $\lambda \in (0,1]$, number of iterations $T$, error
         parameter $\theta$
$P_0 = 0$
for $t = 1, \dots, T$ do
    $P'_t := P_{t-1} + \lambda \mathsf{KM}(u \circ P - u \circ P_{t-1}, \theta)$
    $P_t = \mathsf{KM}(\mathsf{proj}_K(P'_t), \theta)$
end
Output: Return $u \circ P_t$ where $P_t$ is the best over $t = 1, \cdots, T$
```

a sample of y where $\mathbb{E}[y|x] = u(P(x))$. Gopalan et al. (2008) (c.f. Appendix A.1) observed that using distribution queries instead of function queries to the conditional mean is equivalent as long as the number of queries is polynomial. The following theorem proves the correctness of KMtron.

Theorem 7 *For any non-decreasing L-Lipschitz $u : \mathbb{R} \to [0,1]$ and function $P : \{-1,1\}^n \to \mathbb{R}$ such that $L_1(P) \le k$, given query access to $u \circ P$, KMtron run with $\lambda = \frac{\epsilon}{2L}, T = \frac{2k^2L^2}{\epsilon^2}$ and $\theta \le \frac{C'\epsilon^4}{L^4k^3}$ for sufficiently small constant $C' > 0$ and outputs P^* such that $\mathbb{E}_{x\in\{-1,1\}^n}[(u(P(x)) - u(P^*(x)))^2] \le \epsilon$. The runtime of KMtron is $\mathsf{poly}(n, k, L, 1/\epsilon)$.*

Corollary 8 *Let P_i be such that $L_1(P_i) \le k$ for $i \in [s]$. If we have query access to y for all x such that $\mathbb{E}[y|x] = u\left(\frac{1}{s}\sum_{i=1}^s P_i\right)$ for non-decreasing L-Lipschitz u, then using the above, we can learn the conditional mean function in time $\mathsf{poly}(n, k, L, 1/\epsilon)$ with respect to the uniform distribution on $\{-1,1\}^n$.*

Observe that the complexity bounds are *independent of the number of terms*. This follows from the fact that $L_1\left(\frac{1}{s}\sum_{i=1}^s P_i\right) \le k$. This leads to the following new learning result for DNF formulas: fix a DNF f and let $\mathsf{frac}(f(x))$ denote the fraction of terms of f satisfied by x. Fix monotone, L-Lipschitz function u. For uniformly chosen input x, label y is equal to 1 with probability $u(\mathsf{frac}(f(x)))$. Then in time polynomial in n, $1/\epsilon$, and L, KMtron outputs a hypothesis h such that $\mathbb{E}[(h(x) - u(\mathsf{frac}(f(x))))^2] \le \epsilon$ (recall $L_1(\mathsf{AND}) = 1$). Note that the running time has no dependence on the number of terms.

As an easy corollary, we also obtain a simple (no Boosting required) polynomial time query-algorithm for learning DNFs under the uniform distribution[5]:

Corollary 9 *Let f be a DNF formula from $\{-1,1\}^n \to \{0,1\}$ with s terms. Then f is PAC learnable under the uniform distribution using membership queries in time $\mathsf{poly}(n, s, 1/\epsilon)$.*

4.2. Extending the "Low-Degree" Algorithm

Here we show that Alphatron can be used to learn any smooth, monotone combination of function classes that are approximated by *low-degree* polynomials (our other results require us to take advantage of *low-weight* approximations).

5. Feldman (2012) was the first to obtain a query-algorithm for PAC learning DNF formulas with respect to the uniform distribution that did not require a Boosting algorithm.

Definition 10 *For a class of functions $\mathcal{C}$, let $u(\mathcal{C})$ denote monotone function u applied to a linear combination of (polynomially many) functions from $\mathcal{C}$.*

For the domain of $\{-1, 1\}^n$ and degree parameter d, our algorithm will incur a sample complexity and running time factor of n^d, so the "kernel trick" is not necessary (we can work explicitly in the feature space). The main point is that using isotonic regression (as opposed to the original "low-degree" algorithm due to Linial et al. (1993)), we can learn $u(\mathcal{C})$ for any smooth, monotone u and class $\mathcal{C}$ that has low-degree Fourier approximations (we also obtain non-i.i.d. noise tolerance for these classes due to the definition of the probabilistic concept model). While isotonic regression has the flavor of a boosting algorithm, we do not need to change the underlying distribution on points or add noise to the labels, as all boosting algorithms do.

Definition 11 *(ϵ, d)-Fourier concentration A function $f : \{-1, 1\}^n \to \mathbb{R}$ is said to be (ϵ, d)-Fourier concentrated if $\sum_{S:|S|>d} \widehat{f}(S)^2 \leq \epsilon^2$ where $\widehat{f}(S)$ for all $S \subseteq [n]$ are the discrete Fourier coefficients of f.*

Theorem 12 *Consider distribution $\mathcal{D}$ on $\{-1, 1\}^n \times [0, 1]$ whose marginal is uniform on $\{-1, 1\}^n$ and $\mathbb{E}[y|\boldsymbol{x}] = u(f(\boldsymbol{x}))$ for some known non-decreasing L-Lipschitz $u : \mathbb{R} \to [0, 1]$ and $f : \{-1, 1\}^n \to [-M, M]$. If f is (ϵ, d)-Fourier concentrated then there exists an algorithm that draws m iid samples from $\mathcal{D}$ and outputs hypothesis h such that with probability $1 - \delta$, $\varepsilon(h) \leq O(L\epsilon)$ for $m = \mathsf{poly}(n^d, M, 1/\epsilon, \log(1/\delta))$ in time $\mathsf{poly}(n^d, M, 1/\epsilon, \log(1/\delta))$.*

The above can be generalized to linear combinations of Fourier concentrated functions:

Lemma 13 *Let $f = \sum_{i=1}^{k} a_i f_i$ where $f_i : \{-1, 1\}^n \to \mathbb{R}$ and $a_i \in \mathbb{R}$ for all i. If for all i, f_i is (ϵ_i, d_i)-Fourier concentrated, then f is (ϵ, d)-Fourier concentrated for $\epsilon = \sqrt{\left(\sum_{i=1}^{k} a_i^2\right)\left(\sum_{j=1}^{k} \epsilon_j^2\right)}$ and $d = \max_i d_i$.*

Many concept classes are known to be approximated by low-degree Fourier polynomials. Combining Theorem 12 and Lemma 13, we immediately obtain the following results in the probabilistic concept model whose running time matches or improves their best-known PAC counterparts:

- $u(\mathsf{AC}^0)$, generalizing majorities of constant depth circuits (Jackson et al. (2002)).
- $u(\mathsf{LTF})$, generalizing majorities of linear threshold functions (Kalai et al. (2008)).
- $u(\mathsf{SM})$, generalizing submodular functions (Cheraghchi et al. (2012)).

Using the fact that the AND function has a *uniform* approximator of degree $O(\sqrt{n}\log(1/\epsilon))$ (Paturi (1992)), we also obtain a $2^{\tilde{O}(\sqrt{n})}$ time algorithm for *distribution-free* learning of $u(\mathsf{AND})$ in the probabilistic concept model (this class includes the set of all polynomial-size DNF formulas).

4.3. Learning Monotone Functions of Halfspaces with a Margin as Probabilistic Concepts

In this section we consider the problem of learning a smooth combining function u of k halfspaces with a margin ρ. We assume that all examples lie on the unit ball $\mathbb{S}^{n-1}$ and that for each weight vector w, $||\mathbf{w}|| = 1$. For simplicity we also assume each halfspace is origin-centered, i.e. $\theta = 0$ (though our techniques easily handle the case of nonzero θ).

Theorem 14 *Consider samples $(\boldsymbol{x_i}, y_i)_{i=1}^m$ drawn iid from distribution $\mathcal{D}$ on $\mathbb{S}^{n-1} \times [0,1]$ such that $\mathbb{E}[y|\boldsymbol{x}] = u\left(\sum_{i=1}^t \boldsymbol{a}_i h_i(\boldsymbol{x})\right)$ where $u : \mathbb{R}^n \to [0,1]$ is a L-Lipschitz non-decreasing function, h_i are origin-centered halfspaces with margin ρ on $\mathcal{X}$ and $||\boldsymbol{a}||_1 = A$. There exists an algorithm that outputs a hypothesis h such that with probability $1-\delta$,*

$$\mathbb{E}_{\boldsymbol{x},y\sim\mathcal{D}}\left[\left(h(\boldsymbol{x}) - u\left(\sum_{i=1}^t \boldsymbol{a}_i h_i(\boldsymbol{x})\right)\right)^2\right] \le \epsilon$$

for $m = \left(\frac{LA}{\epsilon}\right)^{O(1/\rho)} \log(1/\delta)$. The algorithm runs in time polynomial in m and n.

Remark. If $\mathbb{E}[y|\mathbf{x}] = u\left(\frac{1}{t}\sum_{i=1}^t h_i(\mathbf{x})\right)$, that is, a function of the fraction of true halfspaces, then the run-time is *independent of the number of halfspaces* t. This holds since $A = 1$ in this case.

We now show that Theorem 14 immediately implies the first *polynomial-time* algorithm for PAC learning intersections of halfspaces with a (constant) margin. Consider t-halfspaces $\{h_1, \ldots, h_t\}$. An intersection of these t-halfspaces is given by $f_{\mathsf{AND}}(\mathbf{x}) = \wedge_{i=1}^t h_i(\mathbf{x})$.

Corollary 15 *There exists an algorithm that PAC learns any intersection of t-halfspaces with margin $\rho > 0$ on $\mathbb{S}^{n-1}$ in time $\mathsf{poly}\left(n, \left(\frac{t}{\epsilon}\right)^{(C/\rho)}, \log(1/\delta)\right)$ for some constant C.*

This result improves the previous best bound due to Klivans and Servedio (2008) that had (for constant ρ) a quasipolynomial dependence on the number of halfspaces t. Klivans and Servedio used random projection along with kernel perceptron and the complete quadratic kernel to obtain their results. We remark that if we are only interested in the special case of PAC learning an intersection of halfspaces with a margin (as opposed to learning in the probabilistic concept model), our algorithm is equivalent to running kernel perceptron with the multinomial kernel (and a Chebsyshev approximation that will result in an improved $O(1/\sqrt{\rho})$ dependence). Our improved running time bound follows essentially from a better analysis of the polynomial approximator's weights in the multinomial kernel space using a lemma from Sherstov (2012).

5. Hardness Results for Learning Neural Networks

In this section we reduce problems from Boolean function learning to learning neural networks with one nonlinear layer of ReLU or sigmoid activations. Refer to Appendix C for a discussion on related work.

DNF and Junta Learning. A DNF formula $f : \{0,1\}^n \to \{0,1\}$ is a disjunction of r terms where each term is a conjunction of at most p literals. We have $f(x) = T_1 \vee T_2 \vee \ldots \vee T_r$ where $T_a = l_{a,1} \wedge l_{a,2} \wedge \ldots \wedge l_{a,p}$. Here $l_{a,b}$ are literals which can be a Boolean variable x_i for some $i \in [n]$ or its negation. Learning DNF formulas in polynomial time is one of the outstanding open problems in learning theory. In particular, learning k-term DNF formulas in polynomial-time would imply an algorithm for learning $\log k$-juntas over n variables in polynomial-time (a k-junta is a Boolean function that depends on only k out of n variables). Such a result seems far out of reach (in fact any algorithm with running time $f(k) \cdot \mathsf{poly}(n)$ would be a breakthrough result). The current best algorithm for learning k-juntas is due to Valiant (2015) that runs in time $n^{\frac{(\omega+\epsilon)k}{3}} < n^{0.8k}$ where ω is the exponent of matrix multiplication.

Our Results. It is not difficult to see that a ReLU can compute a term *exactly* and a sigmoid can approximately compute a term. Concretely, σ_{ReLU} can compute a conjunction as follows, $l_1 \wedge \ldots \wedge l_p = \sigma_{\mathsf{ReLU}}\left(\left(\sum_{i=1}^{p} s_i x_i\right) - p + 1\right)$ where x_i is the Boolean variable corresponding to the literal l_i and s_i is -1 if the variable is negated else 1. Similarly, sigmoid can approximate a conjunction as follows, $l_1 \wedge \ldots \wedge l_p = \sigma_{\mathsf{sig}}\left(L\left(\left(\sum_{i=1}^{p} s_i x_i\right) - p + \frac{1}{2}\right)\right) \pm O\left(e^{-L/2}\right)$. As L grows larger, the sigmoidal output moves exponentially closer to the conjunction value. These approximations for "large slope" sigmoids explain why we must bound the two-norm of the weight vector for our results. Also note that, given Boolean variables $x_1, \ldots, x_p$, the following 1-Lipschitz-monotone activation unit can compute a disjunction, $x_1 \vee \ldots \vee x_p = \sigma_{\mathsf{OR}}\left(\sum_{i=1}^{p} x_i\right)$ where $\sigma_{\mathsf{OR}}(a)$ is 0 if $a \leq 0$, a if $0 < a \leq 1$ and 1 otherwise.

Using the principle of inclusion-exclusion, a DNF formula can be written as a sum in the following manner: $f(x) = \sum\limits_{a=1}^{k} T_a - \sum\limits_{1 \leq a < b \leq r} T_a \wedge T_b + \sum\limits_{1 \leq a < b < c \leq r} T_a \wedge T_b \wedge T_c + \ldots$. Observe that $T_a \wedge T_b$ is just a conjunction of the union of $l_{a,\cdot}$ and $l_{b,\cdot}$ literals. Thus f can be represented *exactly* as a linear combination of 2^k conjunctions. Using the fact that ReLUs can compute conjunctions and the fact that k term DNF can compute $\log k$-juntas, we have the following theorem:

Theorem 16 *If there is an algorithm that learns the class of one nonlinear layer neural networks with σ_{ReLU} activation and k hidden units in time $\mathsf{poly}(k, n, 1/\epsilon)$ in the realizable model, then there is an algorithm that can learn the class of $\log\log k$-juntas in time $\mathsf{poly}(k, n, 1/\epsilon)$.*

We obtain similar results for sigmoids but in the p-concept model:

Theorem 17 *If there is an algorithm that learns the class of one nonlinear layer neural networks with σ_{sig} activation and k hidden units in time $\mathsf{poly}(k, n, 1/\epsilon)$ in the p-concept model, then there is an algorithm that can learn the class of $\log\log k$-juntas in time $\mathsf{poly}(k, n, 1/\epsilon)$.*

For the special case of parity, an old result by Bruck (1990) showed that a linear combination of $k+1$ halfspaces can exactly compute parity on k input bits. Further the margin of each halfspace in the construction is $\Omega(1/k)$ therefore we can use "high slope" sigmoids to approximate the halfspaces. For $k = n$ and $L = Cn$, we can show that the distribution of a linear combination of k sigmoids with slope L is exponentially close to the parity learning problem on n bits. This implies that a single layer of sigmoids is indistinguishable (by any subexponential time algorithm) from a function class with exponential SQ dimension. In particular, no SQ algorithm for learning one layer of sigmoids in the p-concept model can run in time polynomial in the number of sigmoids and polynomial in the 2-norm of the hidden weight vectors.

Since difference of ReLUs compute a halfspace, this also implies that parity on k bits can be computed by a linear combination of $O(k)$ ReLUs. This fact was also observed in Mukherjee and Basu (2017). We observe that this directly implies a SQ lower bound for learning one nonlinear layer of ReLUs,

Theorem 18 *The function class of linear combinations of k ReLUs has SQ-dimension $n^{\Omega(k)}$.*

SQ algorithms capture all known learning algorithms (with the exception of learning parity functions; for a complete discussion see Feldman (2016)). Song et al. (2017) proved a type of SQ-lower bound for learning one nonlinear layer of ReLUs with respect to continuous distributions; their work implies that SQ algorithms must use large batch sizes (though not necessarily superpolynomial).

References

Jaume Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105, 2013.

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 and Adam Kalai. A note on learning from multiple-instance examples. *Machine Learning*, 30(1):23–29, 1998.

Jehoshua Bruck. Harmonic analysis of polynomial threshold functions. *SIAM Journal on Discrete Mathematics*, 3(2):168–177, 1990.

Mahdi Cheraghchi, Adam R. Klivans, Pravesh Kothari, and Homin K. Lee. Submodular functions are noise stable. In Yuval Rabani, editor, *SODA*, pages 1586–1592. SIAM, 2012.

Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009.

Amit Daniely. A ptas for agnostically learning halfspaces. In *Conference on Learning Theory*, pages 484–502, 2015.

Amit Daniely. Sgd learns the conjugate kernel class of the network. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 2419–2427, 2017. URL http://papers.nips.cc/paper/6836-sgd-learns-the-conjugate-kernel-class-of-the-network.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71, 1997.

Simon S Du, Jason D Lee, Yuandong Tian, Barnabas Poczos, and Aarti Singh. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *arXiv preprint arXiv:1712.00779*, 2017.

Vitaly Feldman. Learning dnf expressions from fourier spectrum. In Shie Mannor, Nathan Srebro, and Robert C. Williamson, editors, *COLT*, volume 23 of *JMLR Proceedings*, pages 17.1–17.19. JMLR.org, 2012. URL http://jmlr.org/proceedings/papers/v23/.

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

Weihao Gao, Ashok Vardhan Makkuva, Sewoong Oh, and Pramod Viswanath. Learning one-hidden-layer neural networks under general input distributions. *arXiv preprint arXiv:1810.04133*, 2018.

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

Rong Ge, Rohith Kuditipudi, Zhize Li, and Xiang Wang. Learning two-layer neural networks with symmetric inputs. *arxiv preprint arxiv:1810.06793*, 2018.

Surbhi Goel and Adam Klivans. Eigenvalue decay implies polynomial-time learnability of neural networks. In *NIPS*, 2017.

Surbhi Goel, Varun Kanade, Adam Klivans, and Justin Thaler. Reliably learning the relu in polynomial time. *arXiv preprint arXiv:1611.10258*, 2016.

Surbhi Goel, Adam Klivans, and Raghu Meka. Learning one convolutional layer with overlapping patches. *arXiv preprint arXiv:1802.02547*, 2018.

Parikshit Gopalan, Adam Tauman Kalai, and Adam R Klivans. Agnostically learning decision trees. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 527–536. ACM, 2008.

Francisco Herrera, Sebastián Ventura, Rafael Bello, Chris Cornelis, Amelia Zafra, Dánel Sánchez-Tarragó, and Sarah Vluymans. Multiple instance learning. In *Multiple Instance Learning*, pages 17–33. Springer, 2016.

Jeffrey C. Jackson. An efficient membership-query algorithm for learning dnf with respect to the uniform distribution. *J. Comput. Syst. Sci*, 55(3):414–440, 1997.

Jeffrey C. Jackson, Adam R. Klivans, and Rocco A. Servedio. Learnability beyond AC$\hat{0}$. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 776–784, New York, May 19–21 2002. ACM Press.

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. In *Advances in neural information processing systems*, pages 793–800, 2009.

Sham M. Kakade, Adam Kalai, Varun Kanade, and Ohad Shamir. Efficient learning of generalized linear and single index models with isotonic regression. In *NIPS*, pages 927–935, 2011. URL http://papers.nips.cc/book/advances-in-neural-information-processing-systems-24-2011.

Adam Kalai and Ravi Sastry. The isotron algorithm: High-dimensional isotonic regression. In *COLT*, 2009. URL http://www.cs.mcgill.ca/~colt2009/papers/001.pdf#page=1.

Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008.

Michael J Kearns and Robert E Schapire. Efficient distribution-free learning of probabilistic concepts. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 382–391. IEEE, 1990.

A. Klivans, R. O'Donnell, and R. Servedio. Learning intersections and thresholds of halfspaces. *JCSS: Journal of Computer and System Sciences*, 68, 2004.

Adam R. Klivans and Raghu Meka. Moment-matching polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:8, 2013. URL http://eccc.hpi-web.de/report/2013/008.

Adam R Klivans and Rocco A Servedio. Learning intersections of halfspaces with a margin. *Journal of Computer and System Sciences*, 74(1):35–48, 2008.

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.

Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

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

Lee, Bartlett, and Williamson. Lower bounds on the VC-dimension of smoothly parametrized function classes. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 1994.

Linial, Mansour, and Nisan. Constant depth circuits, fourier transform, and learnability. *JACM: Journal of the ACM*, 40, 1993.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pages 855–863, 2014.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in neural information processing systems*, pages 2627–2635, 2014.

Mossel, O'Donnell, and Servedio. Learning juntas. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2003.

Anirbit Mukherjee and Amitabh Basu. Lower bounds over boolean inputs for deep neural networks with relu gates. *arXiv preprint arXiv:1711.03073*, 2017.

Ramamohan Paturi. On the degree of polynomials that approximate symmetric Boolean functions (preliminary version). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on the Theory of Computing*, pages 468–474, Victoria, British Columbia, Canada, 4–6 May 1992.

Sivan Sabato and Naftali Tishby. Multi-instance learning with any hypothesis class. *Journal of Machine Learning Research*, 13(Oct):2999–3039, 2012.

Itay Safran and Ohad Shamir. Depth separation in relu networks for approximating smooth non-linear functions. *CoRR*, abs/1610.09887, 2016. URL http://arxiv.org/abs/1610.09887.

Simone Scardapane, Steven Van Vaerenbergh, Simone Totaro, and Aurelio Uncini. Kafnets: kernel-based non-parametric activation functions for neural networks. *arXiv preprint arXiv:1707.04035*, 2017.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

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.

Ohad Shamir. Distribution-specific hardness of learning neural networks. *arXiv preprint arXiv:1609.01037*, 2016.

Alexander A Sherstov. Making polynomials robust to noise. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 747–758. ACM, 2012.

Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

Mahdi Soltanolkotabi. Learning ReLUs via gradient descent. *arXiv preprint arXiv:1705.04591*, 2017.

Le Song, Santosh Vempala, John Wilmes, and Bo Xie. On the complexity of learning neural networks. *arXiv preprint arXiv:1707.04615*, 2017.

Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *Journal of the ACM (JACM)*, 62(2):13, 2015.

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

Wikipedia. Multinomial theorem — Wikipedia, the free encyclopedia, 2016. [Online; accessed 30-October-2016].

Qiuyi Zhang, Rina Panigrahy, and Sushant Sachdeva. Electron-proton dynamics in deep learning. *CoRR*, abs/1702.00458, 2017. URL http://arxiv.org/abs/1702.00458.

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

Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *ICML*, volume 70, pages 4140–4149. JMLR.org, 2017. URL http://jmlr.org/proceedings/papers/v70/.

Appendix A. Background

A.1. Learning Models

We consider two learning models in our paper, the standard Probably Approximately Correct (PAC) learning model and a relaxation of the standard model, the Probabilistic Concept (p-concept) learning model. For completeness, we define the two models and refer the reader to Valiant (1984); Kearns and Schapire (1990) for a detailed explanation.

Definition 19 (PAC Learning (Valiant (1984))) *We say that a concept class $\mathcal{C} \subseteq \{0,1\}^{\mathcal{X}}$ is Probably Approximately Correct (PAC) learnable, if there exists an algorithm $\mathcal{A}$ such that for every $c \in \mathcal{C}, \delta, \epsilon > 0$ and $\mathcal{D}$ over $\mathcal{X}$, if $\mathcal{A}$ is given access to examples drawn from $\mathcal{D}$ and labeled according to c, $\mathcal{A}$ outputs a hypothesis $h : \mathcal{X} \to \{0,1\}$, such that with probability at least $1-\delta$,*

$$Pr_{\boldsymbol{x}\sim\mathcal{D}}[h(\boldsymbol{x}) \neq c(\boldsymbol{x})] \leq \epsilon. \tag{1}$$

Furthermore, we say that $\mathcal{C}$ is efficiently PAC learnable to error ϵ *if $\mathcal{A}$ can output an h satisfying the above with running time and sample complexity polynomial in n, $1/\epsilon$, and $1/\delta$.*

Definition 20 (p-concept Learning (Kearns and Schapire (1990))) *We say that a concept class $\mathcal{C} \subseteq \mathcal{Y}^{\mathcal{X}}$ is Probabilistic Concept (p-concept) learnable, if there exists an algorithm $\mathcal{A}$ such that for every $\delta, \epsilon > 0$, $c \in \mathcal{C}$ and distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ with $\mathbb{E}[y|\boldsymbol{x}] = c(\boldsymbol{x})$ we have that $\mathcal{A}$, given access to examples drawn from $\mathcal{D}$, outputs a hypothesis $h : \mathcal{X} \to \mathcal{Y}$, such that with probability at least $1-\delta$,*

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[(h(\boldsymbol{x}) - c(\boldsymbol{x}))^2] \leq \epsilon. \tag{2}$$

Furthermore, we say that $\mathcal{C}$ is efficiently p-concept learnable to error ϵ *if $\mathcal{A}$ can output an h satisfying the above with running time and sample complexity polynomial in n, $1/\epsilon$, and $1/\delta$.*

Here we focus on square loss for p-concept since an efficient algorithm for square-loss implies efficient algorithms of various other standard losses.

A.2. Generalization Bounds

The following standard generalization bound based on Rademacher complexity is useful for our analysis. For a background on Rademacher complexity, we refer the readers to Bartlett and Mendelson (2002).

Theorem 21 (Bartlett and Mendelson (2002)) *Let $\mathcal{D}$ be a distribution over $\mathcal{X} \times \mathcal{Y}$ and let $\mathcal{L} : \mathcal{Y}' \times \mathcal{Y}$ (where $\mathcal{Y} \subseteq \mathcal{Y}' \subseteq \mathbb{R}$) be a b-bounded loss function that is L-Lipschitz in its first argument. Let $\mathcal{F} \subseteq (\mathcal{Y}')^{\mathcal{X}}$ and for any $f \in \mathcal{F}$, let $\mathcal{L}(f;\mathcal{D}) := \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathcal{L}(f(\boldsymbol{x}), y)]$ and $\widehat{\mathcal{L}}(f;S) := \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(f(\boldsymbol{x_i}), y_i)$, where $S = ((\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m)) \sim \mathcal{D}^m$. 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:*

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

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

For a linear concept class, the Rademacher complexity can be bounded as follows.

Theorem 22 (Kakade et al. (2009)) *Let $\mathcal{X}$ be a subset of a Hilbert space equipped with inner product $\langle\cdot,\cdot\rangle$ such that for each $\boldsymbol{x} \in \mathcal{X}$, $\langle \boldsymbol{x}, \boldsymbol{x}\rangle \le X^2$, and let $\mathcal{W} = \{\boldsymbol{x} \mapsto \langle \boldsymbol{x}, \boldsymbol{w}\rangle \mid \langle \boldsymbol{w}, \boldsymbol{w}\rangle \le W^2\}$ be a class of linear functions. Then it holds that*

$$\mathcal{R}_{\boldsymbol{m}}(\mathcal{W}) \le X \cdot W \cdot \sqrt{\frac{1}{m}}.$$

The following result is useful for bounding the Rademacher complexity of a smooth function of a concept class.

Theorem 23 (Bartlett and Mendelson (2002); Ledoux and Talagrand (1991)) *Let $\phi : \mathbb{R} \to \mathbb{R}$ be L_ϕ-Lipschitz and suppose that $\phi(0) = 0$. Let $\mathcal{Y} \subseteq \mathbb{R}$, and for a function $f \in \mathcal{Y}^{\mathcal{X}}$. Finally, for $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$, let $\phi \circ \mathcal{F} = \{\phi \circ f \colon f \in \mathcal{F}\}$. It holds that $\mathcal{R}_{\boldsymbol{m}}(\phi \circ \mathcal{F}) \le 2 \cdot L_\phi \cdot \mathcal{R}_{\boldsymbol{m}}(\mathcal{F})$.*

A.3. Kernel Methods

Here we define two kernels and a few of their properties that we will use for our analysis. First, we define a variant of the polynomial kernel, the *multinomial kernel* due to Goel et al. (2016):

Definition 24 (Multinomial Kernel (Goel et al. (2016))) *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 $\psi_d(\boldsymbol{x})$ corresponding to tuple $(k_1, \ldots, k_j)$ equals $\boldsymbol{x}_{k_1} \cdots \boldsymbol{x}_{k_j}$. (When $j = 0$ we have an empty tuple and the corresponding entry is 1.) Define kernel $\mathcal{MK}_d$ as follows:*

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

Also define $\mathcal{H}_{\mathcal{MK}_d}$ to be the corresponding RKHS.

It is easy to see that the multinomial kernel is efficiently computable. A multivariate polynomial p of degree d can be represented as an element $\mathbf{v} \in \mathcal{H}_{\mathcal{MK}_d}$. Also, every $\mathbf{v} \in \mathcal{H}_{\mathcal{MK}_d}$ can be interpreted as a multivariate polynomial of degree d such that

$$p(\mathbf{x}) = \langle \mathbf{v}, \psi_d(\mathbf{x})\rangle = \sum_{\substack{(i_1,\ldots,i_n)\in\{0,\ldots,d\}^n \\ i_i+\cdots+i_n \le d}} \beta(i_1, \ldots, i_n) \mathbf{x}_1^{i_1} \cdots \mathbf{x}_n^{i_n}.$$

where coefficient $\beta(i_1, \ldots, i_n)$ is as follows,

$$\beta(i_1, \ldots, i_n) = \sum_{\substack{k_1,\ldots,k_j \in [n]^j \\ j \in \{0,\ldots,d\}}} \mathbf{v}(k_1, \ldots, k_j).$$

Here, $\mathbf{v}(\cdot)$ is used to index the corresponding entry in $\mathbf{v}$.

The following lemma is due to Goel et al. (2016), following an argument of Shalev-Shwartz et al. (2011):

Lemma 25 *Let $p(t) = \sum_{i=0}^{d} \beta_i t^i$ be a given univariate polynomial with $\sum_{i=1}^{d} \beta_i^2 \leq B^2$. For $\boldsymbol{w}$ such that $||\boldsymbol{w}|| \leq 1$, the polynomial $p(\boldsymbol{w} \cdot \boldsymbol{x})$ equals $\langle p_{\boldsymbol{w}}, \psi(\boldsymbol{x}) \rangle$ for some $p_{\boldsymbol{w}} \in \mathcal{H}_{MK_d}$ with $||p_{\boldsymbol{w}}|| \leq B$.*

Remark. Observe that we can normalize the multinomial feature map such that $\forall \mathbf{x} \in \mathcal{X}, \mathcal{MK}_d(\mathbf{x}, \mathbf{x}) \leq 1$ for bounded space $\mathcal{X}$. More formally, $\max_{\mathbf{x} \in \mathcal{X}} \mathcal{MK}_d(\mathbf{x}, \mathbf{x}) = \max_{\mathbf{x} \in \mathcal{X}} \sum_{j=0}^{d} ||\mathbf{x}||^j \leq \sum_{j=0}^{d} X^j$ where $X = \max_{\mathbf{x} \in \mathcal{X}} ||\mathbf{x}||$, hence we can normalize using this value. Subsequently, in the above, $||p_{\mathbf{w}}||$ will need to be multiplied by the same value. For $\mathcal{X} = \mathbb{S}^{n-1}$, the scaling factor is $d+1$ (Goel et al. (2016)). Throughout the paper, we will assume the kernel to be normalized as discussed.

For our results on Multiple Instance Learning, we make use of the following known kernel defined over sets of vectors:

Definition 26 (Mean Map Kernel (Smola et al. (2007))) *Let $\mathcal{X}^*$ denote the Kleene closure of $\mathcal{X}$.*

The mean map kernel $\mathcal{K}_{\mathsf{mean}} \colon \mathcal{X}^* \times \mathcal{X}^* \to \mathbb{R}$ of kernel $\mathcal{K} \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with feature vector $\psi \colon \mathcal{X} \to \mathcal{Z}$ is defined as,

$$\mathcal{K}_{\mathsf{mean}}(S, T) = \frac{1}{|S||T|} \sum_{\boldsymbol{s} \in S, \boldsymbol{t} \in T} \mathcal{K}(\boldsymbol{s}, \boldsymbol{t}).$$

The feature map $\psi_{\mathsf{mean}} \colon \mathcal{X}^* \to \mathcal{Z}$ corresponding to this kernel is given by

$$\psi_{\mathsf{mean}}(S) = \frac{1}{|S|} \sum_{\boldsymbol{s} \in S} \psi(\boldsymbol{s}).$$

Also define $\mathcal{H}_{\mathsf{mean}}$ to be the corresponding RKHS.

Fact. *If $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \leq M$ then $\forall S, S' \in \mathcal{X}^*, \mathcal{K}_{\mathsf{mean}}(S, S') \leq M$.*

A.4. Approximation Theory

We will make use of a variety of tools from approximation theory to obtain specific embeddings of function classes into a RKHS. The following lemma for approximating the Boolean sign function was given by Daniely (2015):

Lemma 27 *Let $a, \gamma, \tau > 0$. There exists a polynomial p of degree $O\left(\frac{1}{\gamma} \cdot \log\left(\frac{1}{\tau}\right)\right)$ such that*

- *For $x \in [a, a]$, $|p(x)| < 1 + \tau$.*
- *For $x \in [a, a] \backslash [\gamma \cdot a, \gamma \cdot a]$, $|p(x) \mathsf{sign}(x)| < \tau$.*

The above lemma assumes sign takes on values $\{\pm 1\}$, but a simple linear transformation also works for $\{0, 1\}$.

Goel et al. (2016) showed that activation functions sigmoid: $\sigma_{sig}(a) = \frac{1}{1+e^{-a}}$ and ReLU: $\sigma_{relu}(a) = \max(0, a)$ can be (ϵ, B)-uniformly approximated by the multinomial kernel for B dependent on ϵ, more formally they showed the following:

Lemma 28 (Approximating a Single Hidden Unit) *We have,*

1. ***Sigmoid**: For all $\boldsymbol{a} \in \mathbb{S}^{n-1}$ there exists a corresponding $\boldsymbol{v} \in \mathcal{H}_{\mathcal{MK}_d}$ for $d = O(\log(1/\epsilon))$, such that*
$$\forall\, \boldsymbol{x} \in \mathbb{S}^{n-1}, |\sigma_{sig}(\boldsymbol{a} \cdot \boldsymbol{x}) - \langle \boldsymbol{v}, \psi(\boldsymbol{x})\rangle| \le \epsilon.$$
Further, $||\boldsymbol{v}|| \le (1/\epsilon)^{O(1)}$. This implies that σ_{sig} is $(\epsilon, (1/\epsilon)^{O(1)})$-uniformly approximated by $\mathcal{MK}_d$.

2. ***ReLU**: For all $\boldsymbol{a} \in \mathbb{S}^{n-1}$ there exists a corresponding $\boldsymbol{v} \in \mathcal{H}_{\mathcal{MK}_d}$ for $d = O(1/\epsilon)$, such that*
$$\forall\, \boldsymbol{x} \in \mathbb{S}^{n-1}, |\sigma_{relu}(\boldsymbol{a} \cdot \boldsymbol{x}) - \langle \boldsymbol{v}, \psi(\boldsymbol{x})\rangle| \le \epsilon.$$
Further, $||\boldsymbol{v}|| \le 2^{O(1/\epsilon)}$. This implies that σ_{relu} is $(\epsilon, 2^{O(1/\epsilon)})$-uniformly approximated by $\mathcal{MK}_d$.

Finally we state the following lemmas that bound the sum of squares of coefficients of a univariate polynomial:

Lemma 29 (Sherstov (2012)) *Let $p(t) = \sum_{i=0}^{d} \beta_i t^i$ be a univariate polynomial of degree d. Let M be such that $\max\limits_{t\in[-1,1]} |p(t)| \le M$. Then $\sum\limits_{i=0}^{d} \beta_i^2 \le (d+1)\cdot(4e)^{2d}\cdot M^2$.*

Lemma 30 *[Fact 3 (Klivans and Servedio (2008))] Let $p(t) = \sum_{i=0}^{d} \beta_i t^i$ be a univariate polynomial of degree d such that $|\beta_i| \le M$ for all $i \in [d]$. For any $r \in \mathbb{Z}^+$ consider $p^r(t) = \left(\sum_{i=0}^{d} \beta_i t^i\right)^r = \sum_{i=0}^{dr} \eta_i t^i$ then, $\sum\limits_{i=0}^{dr} \eta_i^2 \le (Md)^{2r}$.*

Proof We have $p^r(t) = \sum_{i_1,\cdots,i_r\in[d]} \beta_{i_1}\cdots\beta_{i_r} t^{i_1+\cdots+i_r}$. It follows that $\left(\sum_{i=0}^{d} \beta_i t^i\right)^r = \sum_{i=0}^{dr} \eta_i t^i$ is bounded above by
$$\left(\sum_{i_1,\cdots,i_r\in[d]} |\beta_{i_1}\cdots\beta_{i_r}|\right)^2 \le M^{2r}\left(\sum_{i_1,\cdots,i_r\in[d]} 1\right)^2 = (Md)^{2r}.$$
■

A.5. Properties of KM Algorithm and Projection Operator

The KM algorithm learns sparse approximations to boolean functions given query access to the underlying function. The following lemmas about the KM algorithm are important to our analysis.

Lemma 31 (Kushilevitz and Mansour (1993)) *Given an oracle for $P : \{-1,1\}^n \to \mathbb{R}$, $\mathsf{KM}(P,\theta)$ returns $Q : \{-1,1\}^n \to \mathbb{R}$ with $|\mathsf{supp}(Q)| \le O(L_2(P)^2\theta^{-2})$ and $L_\infty(P-Q) \le \theta$. The running time is $\mathsf{poly}(n, \theta^{-1}, L_2(P))$.*

Lemma 32 (Kushilevitz and Mansour (1993)) *If P has $L_1(P) \le k$, then $\mathsf{KM}\left(P, \frac{\epsilon^2}{2k}\right)$ returns Q s.t. $||P-Q||_2 \le \epsilon$.*

Gopalan et al. (2008) show that proj_K is simple and easy to compute for sparse polynomials. We use the following lemmas by Gopalan et al. (2008) about the projection operator in our analysis.

Lemma 33 (Gopalan et al. (2008)) *Let P, P' be such that $L_\infty(P-P') \leq \epsilon$. Then $L_\infty(\mathsf{proj}_K(P)-\mathsf{proj}_K(P')) \leq 2\epsilon$.*

Lemma 34 (Gopalan et al. (2008)) *Let P, P' be such that $L_\infty(P-P') \leq \epsilon$. Then $||\mathsf{proj}_K(P)-\mathsf{proj}_K(P')||_2 \leq 2\sqrt{\epsilon k}$.*

Appendix B. Omitted Proofs

Proof of Theorem 1. Define $\mathbf{v^t} = \sum_{i=1}^m \alpha_i^t \psi(\mathbf{x_i})$ implying $h^t(\mathbf{x}) = u(\langle \mathbf{v^t}, \psi(\mathbf{x})\rangle)$. Let $\Delta := \frac{1}{m}\sum_{i=1}^m (y_i - u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle) + \xi(\mathbf{x_i}))\psi(\mathbf{x_i})$ and $\Delta^t := \frac{1}{m}\sum_{i=1}^m (y_i - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))\psi(\mathbf{x_i})$. Aslo define $\rho := \frac{1}{m}\sum_{i=1}^m \xi(\mathbf{x_i})^2$. We will use the following lemma:

Lemma 35 *At iteration t in Alphatron, suppose $||\boldsymbol{v^t} - \boldsymbol{v}|| \leq B$ for $B > 1$, then if $||\Delta|| \leq \eta < 1$, then*

$$||\boldsymbol{v^t} - \boldsymbol{v}||^2 - ||\boldsymbol{v^{t+1}} - \boldsymbol{v}||^2 \geq \lambda\left(\left(\frac{2}{L} - \lambda\right)\widehat{\varepsilon}(h^t) - 2\sqrt{\rho} - 2B\eta - \lambda\eta^2 - 2\lambda\eta\right).$$

Proof Expanding the left hand side of the equation above, we have

$$||\mathbf{v^t} - \mathbf{v}||^2 - ||\mathbf{v^{t+1}} - \mathbf{v}||^2 \tag{3}$$
$$= 2\lambda\langle \mathbf{v} - \mathbf{v^t}, \Delta^t\rangle - \lambda^2||\Delta^t||^2 \tag{4}$$
$$= 2\lambda\langle \mathbf{v} - \mathbf{v^t}, \Delta^t - \Delta\rangle + 2\lambda\langle \mathbf{v} - \mathbf{v^t}, \Delta\rangle - \lambda^2||\Delta^t||^2 \tag{5}$$
$$\geq \frac{2\lambda}{m}\sum_{i=1}^m (u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))\langle \mathbf{v} - \mathbf{v^t}, \psi(\mathbf{x_i})\rangle - 2\lambda B||\Delta|| - \lambda^2||\Delta^t||^2 \tag{6}$$
$$= \frac{2\lambda}{m}\sum_{i=1}^m (u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i}) - \langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle)$$
$$- \frac{2\lambda}{m}\sum_{i=1}^m (u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))\xi(\mathbf{x_i}) - 2\lambda B||\Delta|| - \lambda^2||\Delta^t||^2 \tag{7}$$
$$\geq \frac{2\lambda}{Lm}\sum_{i=1}^m (u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))^2$$
$$- \frac{2\lambda}{m}\sum_{i=1}^m |u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle)||\xi(\mathbf{x_i})| - 2\lambda B||\Delta|| - \lambda^2||\Delta^t||^2 \tag{8}$$
$$\geq \frac{2\lambda}{L}\widehat{\varepsilon}(h^t) - 2\lambda\sqrt{\rho} - 2\lambda B\eta - \lambda^2||\Delta^t||^2 \tag{9}$$

Here (4) follows from substituting the expression of $\mathbf{v^{t+1}}$, (6) follows from bounding $||\mathbf{v^t} - \mathbf{v}|| \leq B$ and, (8) follows from u being monotone and L-Lipschitz, that is, $(u(a) - u(b))(a - b) \geq \frac{1}{L}(u(a) - u(b))^2$. (9) follows from the definition of $\widehat{\varepsilon}(h^t)$, the second term follows from the fact that u is $[0, 1]$ and $\frac{1}{m}\sum_{i=1}^m \sum_{i=1}^m |\xi(\mathbf{x_i})| \leq \sqrt{\frac{1}{m}\sum_{i=1}^m \sum_{i=1}^m \xi(\mathbf{x_i})^2} = \sqrt{\rho}$ and the third term is bounded using the assumption $||\Delta|| \leq \eta$.

We now bound $||\Delta^t||$ as follows.

$$||\Delta^t||^2 = \left|\left|\frac{1}{m}\sum_{i=1}^m (y_i - u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) + u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))\psi(\mathbf{x_i})\right|\right|^2 \tag{10}$$

$$\leq ||\Delta||^2 + \left|\left|\frac{1}{m}\sum_{i=1}^m (u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))\psi(\mathbf{x_i})\right|\right|^2$$

$$+ 2||\Delta|| \left|\left|\frac{1}{m}\sum_{i=1}^m (u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})) - u(\langle \mathbf{v^t}, \psi(\mathbf{x_i})\rangle))\psi(\mathbf{x_i})\right|\right| \tag{11}$$

$$\leq \eta^2 + \widehat{\varepsilon}(h^t) + 2\eta \tag{12}$$

Here (13) follows by expanding the square and (12) follows by applying Jensen's inequality to show that for all $\mathbf{a}, \mathbf{b} \in \mathbb{R}^m$ and vectors $\mathbf{v_i}$ for $i \in \{1, \cdots, m\}$, $\left|\left|\frac{1}{m}\sum_{i=1}^m (\mathbf{a_i} - \mathbf{b_i})\mathbf{v_i}\right|\right|^2 \leq \frac{1}{m}\sum_{i=1}^m (\mathbf{a_i} - \mathbf{b_i})^2 ||\mathbf{v_i}||^2$ and subsequently using the fact that $||\psi(\mathbf{x_i})|| \leq 1$. Combining (9) and (12) gives us the result. ■

By definition we have that $(y_i - u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})))\psi(\mathbf{x_i})$ are zero mean iid random variables with norm bounded by 1. Using Hoeffding's inequality (and the fact that that the $\mathbf{x_i}$'s are independent draws), with probability $1 - \delta$ we have

$$||\Delta|| = \left|\left|\frac{1}{m}\sum_{i=1}^m (y_i - u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})))\psi(\mathbf{x_i})\right|\right| \leq \frac{1}{\sqrt{m}}\left(1 + \sqrt{2\log(1/\delta)}\right).$$

Similarly, we can bound ρ using Hoeffding's inequality to get,

$$\rho \leq \epsilon + O\left(M^2\sqrt{\frac{\log(1/\delta)}{m}}\right) \implies \sqrt{\rho} \leq \sqrt{\epsilon} + O\left(M\sqrt[4]{\frac{\log(1/\delta)}{m}}\right)$$

Now using the previous lemma with $\lambda = 1/L$ and $\eta = \frac{1}{\sqrt{m}}\left(1 + \sqrt{2\log(1/\delta)}\right)$, we have

$$||\mathbf{v^t} - \mathbf{v}||^2 - ||\mathbf{v^{t+1}} - \mathbf{v}||^2 \geq \frac{1}{L}\left(\frac{\widehat{\varepsilon}(h^t)}{L} - 2\sqrt{\rho} - 2B\eta - \frac{\eta^2}{L} - \frac{2\eta}{L}\right).$$

Thus, for each iteration t of Alphatron, one of the following two cases needs to be satisfied,

Case 1: $||\mathbf{v^t} - \mathbf{v}||^2 - ||\mathbf{v^{t+1}} - \mathbf{v}||^2 \geq \frac{B\eta}{L}$

Case 2: $\widehat{\varepsilon}(h^t) \leq 3BL\eta + 2L\sqrt{\rho} + \eta^2 + 2\eta = O\left(L\epsilon + LM\sqrt[4]{\frac{\log(1/\delta)}{m}} + BL\sqrt{\frac{\log(1/\delta)}{m}}\right)$.

Let t^* be the first iteration where Case 2 holds. We need to show that such an iteration exists. Assume the contradictory, that is, Case 2 fails for each iteration. Since $||\mathbf{v^0} - \mathbf{v}||^2 \leq B^2$, however, in at most $\frac{BL}{\eta}$ iterations Case 1 will be violated and Case 2 will have to be true. If $\frac{BL}{\eta} \leq T$ then t^* exists such that

$$\widehat{\varepsilon}(h^{t^*}) \leq O\left(L\sqrt{\epsilon} + LM\sqrt[4]{\frac{\log(1/\delta)}{m}} + BL\sqrt{\frac{\log(1/\delta)}{m}}\right).$$

We need to bound $\varepsilon(h^{t^*})$ in terms of $\widehat{\varepsilon}(h^{t^*})$. Define $\mathcal{F} = \{\mathbf{x} \to u(\langle \mathbf{z}, \psi(\mathbf{x})\rangle) : ||\mathbf{z}|| \leq 2B\}$, and $\mathcal{Z} = \{\mathbf{x} \to f(\mathbf{x}) - u(\langle \mathbf{v}, \psi(\mathbf{x})\rangle + \xi(\mathbf{x})) : f \in \mathcal{F}\}$. Using Theorem 22 and 23 we have $\mathcal{R}_m(\mathcal{F}) = O(BL\sqrt{1/m})$. By definition of Rademacher complexity, we have

$$\begin{aligned}
\mathcal{R}_m(\mathcal{Z}) &= \mathbb{E}_{\mathbf{x_i},\sigma_i}\left[\sup_{z\in\mathcal{Z}}\left(\frac{2}{m}\sum_{i=1}^{m}\sigma_i z(\mathbf{x_i})\right)\right] \\
&= \mathbb{E}_{\mathbf{x_i},\sigma_i}\left[\sup_{f\in\mathcal{F}}\left(\frac{2}{m}\sum_{i=1}^{m}\sigma_i (f(\mathbf{x_i}) - u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i})))\right)\right] \\
&= \mathbb{E}_{\mathbf{x_i},\sigma_i}\left[\sup_{f\in\mathcal{F}}\left(\frac{2}{m}\sum_{i=1}^{m}\sigma_i f(\mathbf{x_i})\right)\right] - \mathbb{E}_{\mathbf{x_i}}\left[\frac{2}{m}\sum_{i=1}^{m}\mathbb{E}_{\sigma_i}[\sigma_i] u(\langle \mathbf{v}, \psi(\mathbf{x_i})\rangle + \xi(\mathbf{x_i}))\right] \\
&= \mathcal{R}_m(\mathcal{F})
\end{aligned}$$

Here, $\sigma_i \in \{\pm 1\}$ are iid Rademacher variables hence $\forall\, i, E[\sigma_i] = 0$ and $\mathbf{x_i}$ are drawn iid from $\mathcal{D}$.

Recall that $h^{t^*}(\mathbf{x}) = u(\langle \mathbf{v^T}, \psi(\mathbf{x})\rangle)$ is an element of $\mathcal{F}$ as $||\mathbf{v^T} - \mathbf{v}||^2 \leq B^2$ (case 1 is satisfied in iteration $t^* - 1$) and $||\mathbf{v}|| \leq B$. A direct application of Theorem 21 on $\mathcal{Z}$ with loss function $\mathcal{L}(a, \cdot) = a^2$, gives us, with probability $1 - \delta$,

$$\varepsilon(h^{t^*}) \leq \widehat{\varepsilon}(h^{t^*}) + O\left(BL\sqrt{\frac{1}{m}} + \sqrt{\frac{\log(1/\delta)}{m}}\right) = O\left(L\sqrt{\epsilon} + LM\sqrt[4]{\frac{\log(1/\delta)}{m}} + BL\sqrt{\frac{\log(1/\delta)}{m}}\right).$$

The last step is to show that we can indeed find a hypothesis satisfying the above guarantee. Since for all h, $\varepsilon(h)$ is up to constants equal to $err(h)$ we can do so by choosing the hypothesis with the minimum err using a fresh sample set of size $O(\log(T/\delta)/\epsilon_0^2) \leq N$. This holds as given the sample size, by Chernoff bound using the fact that $\widehat{\varepsilon}(h^t)$ is bounded in $[0, 1]$, each h^t for $t \leq T$ will have empirical error within ϵ_0 of the true error with probability $1 - \delta/T$ and hence all will simultaneously satisfy this with probability $1 - \delta$. Setting $\epsilon_0 = 1/\sqrt{m}$ will give us the required bound.

Proof of Theorem 6. We first extend the approximation guarantees to linear combinations of function classes using the following lemma.

Lemma 36 *If for all $i \in [k]$, f_i is (ϵ, B)-uniformly approximated in kernel $\mathcal{K}$ then $\sum_{i=1}^{k} \boldsymbol{a}_i f_i(\boldsymbol{x})$ for $\boldsymbol{a} \in \mathbb{R}^k$ such that $||\boldsymbol{a}||_1 \leq W$ is $(\epsilon W, WB)$-uniformly approximated in kernel $\mathcal{K}$.*

Proof We have for each $i \in [k]$, $\forall \mathbf{x} \in \mathcal{X}, |f_i(\mathbf{x}) - \langle \mathbf{v_i}, \psi(\mathbf{x})\rangle| \leq \epsilon$ for some $\mathbf{v_i} \in \mathcal{H}$ such that $||\mathbf{v_i}|| \leq B$. Consider $\mathbf{v} = \sum_{i=1}^{k} \mathbf{a}_i \mathbf{v_i}$. We have $\forall \mathbf{x} \in \mathcal{X}$,

$$\left|\sum_{i=1}^{k} a_i f_i(\mathbf{x}) - \langle \mathbf{v}, \psi(\mathbf{x})\rangle\right| = \left|\sum_{i=1}^{k} \mathbf{a}_i (f_i(\mathbf{x}) - \langle \mathbf{v_i}, \psi(\mathbf{x})\rangle)\right| \leq \sum_{i=1}^{k} |a_i||f_i(\mathbf{x}) - \langle \mathbf{v_i}, \psi(\mathbf{x})\rangle| \leq \epsilon ||\mathbf{a}||_1 = \epsilon W.$$

Also $||\mathbf{v}|| = \left|\left|\sum_{i=1}^{k} \mathbf{a}_i \mathbf{v_i}\right|\right| \leq \sum_{i=1}^{k} |\mathbf{a}_i|||\mathbf{v_i}|| \leq WB$. Thus $\mathbf{v}$ satisfies the required approximation. ■

Combining Lemmas 28 and 36 we have that $\mathcal{N}_1$ for activation function σ_{sig} is $(\epsilon_0\sqrt{k}, (\sqrt{k}/\epsilon_0)^C)$-uniformly approximated by some kernel $\mathcal{MK}_d$ with $d = O(\log(1/\epsilon_0))$ and sufficiently large constant $C > 0$. Thus by Theorem 5, we have that there exists an algorithm that outputs a hypothesis h

such that, with probability $1-\delta$,

$$\mathbb{E}_{\mathbf{x},y\sim\mathcal{D}}(h(\mathbf{x})-\mathcal{N}_2(\mathbf{x}))^2 \leq C'L\left(\epsilon_0\sqrt{k}+\left(\frac{\sqrt{k}}{\epsilon_0}\right)^C\cdot\sqrt{\frac{\log(1/\delta)}{m}}\right)$$

for some constants $C'>0$. Setting $\epsilon_0=\frac{\epsilon}{2\sqrt{k}C'L}$ and $m=\left(\frac{2kCL}{\epsilon}\right)^{2C}\cdot\left(\frac{4\log(1/\delta)}{\epsilon^2}\right)$ gives us the required result (the claimed bounds on running time also follow directly from Theorem 5).

The guarantees can be extended to ReLUs using the same outline but a worse dependence on the parameters.

Theorem 37 *Consider samples $(\boldsymbol{x_i},y_i)_{i=1}^m$ drawn iid from distribution $\mathcal{D}$ on $\mathbb{S}^{n-1}\times[0,1]$ such that $E[y|\boldsymbol{x}]=\mathcal{N}_2(\boldsymbol{x})$ with $\sigma':\mathbb{R}\to[0,1]$ is a known L-Lipschitz non-decreasing function and $\sigma=\sigma_{relu}$ is the ReLU function. There exists an algorithm that outputs a hypothesis h such that with probability $1-\delta$,*

$$\mathbb{E}_{\boldsymbol{x},y\sim\mathcal{D}}\left[(h(\boldsymbol{x})-\mathcal{N}_2(\boldsymbol{x}))^2\right]\leq\epsilon$$

for $m=2^{O(kL/\epsilon)}\cdot\log(1/\delta)$. The algorithm runs in time polynomial in m and n.

Proof of Theorem 7. Let $\varepsilon(h)=\mathbb{E}_{x\in\{-1,1\}^n}[(u(P(x))-u(h(x)))^2]=||u\circ P-u\circ h||_2^2$. Similar to Alphatron, we will show that with each iteration t we will move closer to the true solution as long as $\varepsilon(P_t)$ is large.

Lemma 38 *For a suitable choice of θ, $||P_t-P||_2^2-||P_{t+1}-P||_2^2\geq\frac{2\lambda}{L}(\varepsilon(P_t)-L\lambda)$.*

Proof Let us define the following polynomials for $t\leq T$, $Q'_t=P_{t-1}+\lambda(u\circ P-u\circ P_{t-1})$ and $Q_t=\mathsf{proj}_K(Q'_t)$. For all $t\leq T$,

$$\begin{aligned}P'_t-Q'_t&=(P_{t-1}+\lambda\mathsf{KM}(u\circ P-u\circ P_{t-1},\theta))-(P_{t-1}+\lambda(u\circ P-u\circ P_{t-1}))\\&=\lambda(\mathsf{KM}(u\circ P-u\circ P_{t-1},\theta)-(u\circ P-u\circ P_{t-1})).\end{aligned}$$

From Lemma 31, $L_\infty(\mathsf{KM}(u\circ P-u\circ P_{t-1},\theta)-(u\circ P-u\circ P_{t-1}))\leq\theta$ implying $L_\infty(P'_t-Q'_t)\leq\lambda\theta\leq\theta$ since $\lambda\leq1$.

Using Lemma 34, we have $||\mathsf{proj}_K(P'_t)-\mathsf{proj}_K(Q'_t)||_2\leq2\sqrt{\theta k}$. Since $P_t=\mathsf{KM}(\mathsf{proj}_K(P'_t),\theta)$ and $L_1(\mathsf{proj}_K(P'_t))\leq k$, using Lemma 32, $||P_t-\mathsf{proj}_K(P'_t)||_2\leq\sqrt{2\theta k}$. Using Triangle inequality, we get,

$$||P_t-Q_t||_2\leq||P_t-\mathsf{proj}_K(P'_t)||_2+||\mathsf{proj}_K(P'_t)-\mathsf{proj}_K(Q'_t)||_2<4\sqrt{\theta k}.$$

Observe that $||Q_t-P||_2=L_2(Q_t-P)\leq L_1(Q_t-P)\leq L_1(Q_t)+L_1(P)\leq2k$. Combining these two observations, we have

$$||P_t-P||_2^2\leq(||P_t-Q_t||_2+||Q_t-P||_2)^2\leq||Q_t-P||^2+16k\sqrt{\theta k}+16\theta k\leq||Q_t-P||^2+Ck\sqrt{\theta k}$$

for large enough constant $C>0$. Therefore,

$$\begin{aligned}&||P_t-P||_2^2-||P_{t+1}-P||_2^2\\&\geq||P_t-P||_2^2-||Q_{t+1}-P||^2-Ck\sqrt{\theta k}\qquad(13)\end{aligned}$$

$$\begin{aligned}
&\geq ||P_t - P||_2^2 - ||Q'_{t+1} - P||^2 - Ck\sqrt{\theta k} && (14)\\
&= ||P_t - P||_2^2 - ||P_t - P + \lambda(u \circ P - u \circ P_t)||_2^2 - Ck\sqrt{\theta k} && (15)\\
&= -2\lambda(u \circ P - u \circ P_t) \cdot (P_t - P) - \lambda^2 ||u \circ P - u \circ P_t||_2^2 - Ck\sqrt{\theta k} && (16)\\
&\geq \frac{2\lambda}{L} \cdot \varepsilon(P_t) - \lambda^2 - Ck\sqrt{\theta k}.
\end{aligned}$$

Here, (13) follows from the triangle inequality and (B), (14) follows from projecting to a convex set reducing the distance to points in the convex set, (16) follows from u being monotone, L-Lipschitz with output bounded in $[0, 1]$. Setting θ such that $Ck\sqrt{\theta k} \leq \lambda^2$ gives the required result. ■

As long as $\varepsilon(P_t) \geq 2L\lambda$, we have $||P_t - P||_2^2 - ||P_{t+1} - P||_2^2 \geq 2\lambda^2$. Since $||P_0 - P||_2^2 = ||P||_2^2 \leq L_1(P)^2 \leq k^2$, after $T = \frac{k^2}{2\lambda^2}$, there must be some $r \leq T$ such that $||P_r - P||_2^2 \geq 2\lambda^2$ does not hold, at this iteration, $\varepsilon(P_t) \leq 2L\lambda = \epsilon$ for $\lambda = \frac{\epsilon}{2L}$. The last step of choosing the best P_t would give us the required hypothesis (similar to Alphatron). Observe that each iteration of KMtron runs in time poly$(n, k, L, 1/\epsilon)$ (Lemma 31) and KMtron is run for poly$(k, L, 1/\epsilon)$ iterations giving us the required runtime.

Proof of Corollary 9. Let $\{T_i\}_{i=1}^s$ be the ANDs corresponding to each term of the DNF. Let $T = \sum_{i=1}^s T_i$. By definition of f, if $f(x) = 1$ then $T(x) \geq 1$ and if $f(x) = 0$ then $T(x) = 0$. Observe that $L_1(T) \leq \sum_{i=1}^s L_1(T_i) \leq s$ using the well known fact that AND has L_1 bounded by 1.

Consider the following u,

$$u(a) = \begin{cases} 0 & a \leq 0 \\ a & 0 < a < 1 \\ 1 & a \geq 1 \end{cases}$$

Observe that u is 1-Lipschitz. It is easy to see that $f(x) = u(T(x))$ on $\{-1, 1\}^n$. Hence, given query access to f is the same as query access to $u \circ T$ over $\{-1, 1\}^n$.

Since $L_1(T)$ is bounded, we can apply Theorem 7 for the given u. We get that in time poly$(n, s, 1/\epsilon)$, KMtron outputs a polynomial P such that $\mathbb{E}_{x \in \{-1,1\}^n}[(u(T(x)) - u(P(x)))^2] \leq \epsilon$. Recall that $u \circ P$ may be real-valued as KMtron learns with square loss. To convert it to a boolean function with 0-1 loss bounded, we use the following lemma with f and $g = u \circ P$.

Lemma 39 *Let $f : \mathbb{R}^n \to \{0, 1\}$ be a boolean function and $g : \mathbb{R}^n \to [0, 1]$ be a real-valued function. If $\mathbb{E}[(f(\boldsymbol{x}) - g(\boldsymbol{x}))^2] \leq \epsilon$ then $Pr[h(\boldsymbol{x}) \neq f(\boldsymbol{x})] \leq 4\epsilon$ where $h : \mathbb{R}^n \to \{0, 1\}$ is such that it is 1 when $g(\boldsymbol{x}) \geq 1/2$ and 0 otherwise.*

Proof Using Markov's inequality, we have $Pr[(g(\mathbf{x}) - f(\mathbf{x}))^2 \geq 1/4] \leq 4\epsilon$. For $\mathbf{x}$, if $(g(\mathbf{x}) - f(\mathbf{x}))^2 < 1/4$ then $|g(\mathbf{x}) - f(\mathbf{x})| < 1/2$. Since $f(\mathbf{x}) \in \{0, 1\}$ then clearly $h(\mathbf{x}) = f(\mathbf{x})$. Thus, $Pr[h(\mathbf{x}) \neq f(\mathbf{x})] \leq 4\epsilon$. ■

Scaling ϵ appropriately gives us the result.

Proof of Theorem 12. Consider polynomial $P(\mathbf{x}) = \sum_{S:|S|\le d} \widehat{f}(S)\chi_S(\mathbf{x})$. We have,

$$\mathbb{E}_{\mathcal{D}}[(f(\mathbf{x}) - P(\mathbf{x}))^2] = \mathbb{E}_{\mathcal{D}}\left[\left(\sum_{S:|S|>d} \widehat{f}(S)\chi_S(\mathbf{x})\right)^2\right] = \sum_{S:|S|>d} \widehat{f}(S)^2 \le \epsilon^2.$$

We also know that $\widehat{f}(S) \le M$ (since $|f(\mathbf{x})| \le M$) for all $S \subseteq [n]$, thus $|f(\mathbf{x}) - P(\mathbf{x})| \le |f(\mathbf{x})| + |P(\mathbf{x})| = O(n^d M)$ for all $\mathbf{x} \in \{-1,1\}^n$. Also observe that

$$\sum_{S:|S|\le d} \widehat{f}(S)^2 \le \sum_S \widehat{f}(S)^2 = \frac{1}{2^n} \sum_{\mathbf{x}\in\{-1,1\}^n} f(\mathbf{x}) \le M$$

where the equality follows from Parseval's Theorem. This implies that f is $(\epsilon, \sqrt{M}, n^d M)$-approximated by kernel $\mathcal{K}$ and RKHS $\mathcal{H}$ with feature map ψ of all monomials of degree $\le d$. This kernel takes $O(n^d)$ time to compute. Now, we apply Theorem 3 after renormalizing the kernel to get the required result.

Proof of Lemma 13. For all $S \subseteq [n]$, we have $\widehat{f}(S) = \sum_{i=1}^k a_i \widehat{f}_i(S)$. Let ϵ and d be as in the lemma, we have,

$$\begin{aligned}
\sum_{S:|S|>d} \widehat{f}(S)^2 &= \sum_{S:|S|>d} \left(\sum_{i=1}^k a_i \widehat{f}_i(S)\right)^2 \\
&\le \sum_{S:|S|>d} \left(\sum_{i=1}^k a_i^2\right)\left(\sum_{j=1}^k \widehat{f}_j(S)^2\right) \\
&\le \left(\sum_{i=1}^k a_i^2\right)\left(\sum_{i=1}^k \sum_{S:|S|>d_j} \widehat{f}_j(S)^2\right) \\
&= \left(\sum_{i=1}^k a_i^2\right)\left(\sum_{i=1}^k \epsilon_i^2\right) = \epsilon^2.
\end{aligned}$$

Here the first inequality follows from Cauchy-Schwarz, the second follows from rearranging the sum and using the fact that $\{S : |S| > d\} \subseteq \{S : |S| > d_j\}$ and the third follows from the Fourier concentration of each f_i.

Proof of Theorem 14. We use Lemma 27 to show the existence of polynomial p of degree $d = O\left(\frac{1}{\rho} \cdot \log\left(\frac{1}{\epsilon_0}\right)\right)$ such that for $a \in [1,1]$, $|p(a)| < 1+\epsilon_0$ and for $a \in [1,1]\backslash[-\rho,\rho]$, $|p(a)\mathsf{sign}(a)| < \epsilon_0$.

Since for each i, $\rho \le |\mathbf{w_i} \cdot \mathbf{x}| \le 1$, we have $|p(\mathbf{w_i} \cdot \mathbf{x})\mathsf{sign}(\mathbf{w_i} \cdot \mathbf{x})| \le \epsilon_0$ such that p is bounded in $[-1,1]$ by $1+\epsilon_0$. From Lemma 29 and 25, we have that for each i, $p(\mathbf{w_i} \cdot \mathbf{x}) = \langle \mathbf{v_i}, \psi_d(\mathbf{x})\rangle$ such that $||\mathbf{v_i}|| = \left(\frac{1}{\epsilon_0}\right)^{O(1/\rho)}$ where ψ_d is the feature vector corresponding to the multinomial kernel of degree d. Using Lemma 36, we have that $\sum_{i=1}^t \mathbf{a}_i h_i(\mathbf{x})$ is $\left(\epsilon_0 A, A\left(\frac{1}{\epsilon_0}\right)^{O(1/\rho)}\right)$-uniformly approximated by $\mathcal{MK}_d$.

Subsequently, applying Theorem 5, we get that there exists an algorithm that outputs a hypothesis h such that with probability $1-\delta$,

$$\varepsilon(h) \le CLA\left(\epsilon_0 + \left(\frac{1}{\epsilon_0}\right)^{C'/\rho} \cdot \sqrt{\frac{\log(1/\delta)}{m}}\right)$$

for some constants $C, C' > 0$. Setting $\epsilon_0 = \frac{\epsilon}{2CLA}$ and $m = \left(\frac{2CLA}{\epsilon}\right)^{2C'/\rho} \cdot \left(\frac{4\log(1/\delta)}{\epsilon^2}\right)$ to gives us the required result.

Proof of Corollary 15. Let $T(\mathbf{x}) = \frac{1}{t}\sum_{i=1}^{t} \mathbf{a}_i h_i(\mathbf{x})$. Consider the following u,

$$u(a) = \begin{cases} 0 & a \le 1 - \frac{1}{t} \\ a & 1 - \frac{1}{t} < a < 1 \\ 1 & a \ge 1 \end{cases}$$

Observe that u is $1/t$-Lipschitz. It is easy to see that $f_{\mathsf{AND}}(\mathbf{x}) = u(T(\mathbf{x})$. Using Theorem 14 for $L = 1$ and $A = 1$, we know that there exists an algorithm that runs in time poly $\left(n, \left(\frac{t}{\epsilon}\right)^{(C/\rho)}, \log(1/\delta)\right)$ and outputs a hypothesis h such that $\mathbb{E}_{\mathbf{x},y\sim\mathcal{D}}\left[\left(h(\mathbf{x}) - u\left(\sum_{i=1}^{t} \mathbf{a}_i h_i(\mathbf{x})\right)\right)^2\right] \le \epsilon$. Since h is a real-valued function, we can use $\mathsf{sign}(h)$ to get a 0-1 loss bound as in the proof of Corollary 9 to get the required result.

Proof of Theorem 43. The following lemma is useful for our analysis.

Lemma 40 *Let c be the instance labeling function mapping $\mathcal{X}$ to $\mathbb{R}$ such that c is (ϵ, B)-uniformly approximated by kernel $\mathcal{K}$ and feature vector ψ. Then the function $f : \mathfrak{B} \to \mathbb{R}$ given by $f(\beta) = \frac{1}{|\beta|} \cdot \sum_{\mathbf{x}\in\beta} c(\mathbf{x})$ is (ϵ, B)-uniformly approximated by the mean map kernel of $\mathcal{K}$.*

Proof We have that $\forall \mathbf{x} \in \mathcal{X}, |c(\mathbf{x}) - \langle v, \psi(\mathbf{x})\rangle| \le \epsilon$ for v such that $||\mathbf{v}|| \le B$. Let $\mathcal{K}_{\mathsf{mean}}$ be the mean map kernel of $\mathcal{K}$ and ψ_{mean} be the corresponding vector. We will show that $\mathbf{v}$ (ϵ, B)-approximates f in $\mathcal{K}_{\mathsf{mean}}$. This follows from the following,

$$\begin{aligned} |f(\beta) - \langle \mathbf{v}, \psi_{\mathsf{mean}}(\beta)\rangle| &= \left|\frac{1}{|\beta|} \cdot \sum_{\mathbf{x}\in\beta} c(\mathbf{x}) - \frac{1}{|\beta|} \cdot \sum_{\mathbf{x}\in\beta} \langle \mathbf{v}, \psi(\mathbf{x})\rangle\right| \\ &\le \frac{1}{|\beta|} \cdot \sum_{\mathbf{x}\in\beta} |c(\mathbf{x}) - \langle \mathbf{v}, \psi(\mathbf{x})\rangle| \le \epsilon. \end{aligned}$$

■

Consider $c \in \mathcal{C}$ that is $(\epsilon/CL, B)$-uniformly approximated by some kernel $\mathcal{K}$ for large enough constant $C > 0$ (to be chosen later). Using Lemma 40 we know that $f(\beta) = \frac{1}{|\beta|} \cdot \sum_{\mathbf{x}\in\beta} c(\mathbf{x})$ is $(\epsilon/CL, B)$-uniformly approximated by the mean map kernel $\mathcal{K}_{\mathsf{mean}}$ of $\mathcal{K}$. Applying Theorem 5, we get that with probability $1-\delta$,

$$\mathbb{E}_{\beta\sim\mathcal{D}}\left[\left(h(\beta) - u\left(\frac{1}{|\beta|} \cdot \sum_{\mathbf{x}\in\beta} c(\mathbf{x})\right)\right)^2\right] \le \frac{C'}{C}\epsilon.$$

for sufficiently large constant $C' > 0$. Choosing $C \ge C'$ gives the result.

Proof of Theorem 16. A one-layer neural network with k hidden units and σ_{relu} can compute a $\log k$-term DNF exactly. We also know that $\log k$-DNF computes $\log \log k$-juntas. Thus we have that the network computes a $\log \log r$-junta. However, the network learns with respect to square loss. We can then use Lemma 39 for $0/1$ loss closeness.

Proof of Theorem 17. Similar to the ReLU case, k-hidden unit sigmoidal network can approximately compute a $\log k$-DNF. Using the fact that sigmoid can approximately represent a conjunction within error $O(e^{-L})$, it can approximately compute a $\log r$-size DNF within error $O(re^{-L})$. Let g denote this approximate function. Then we have for all x, $|f(x) - g(x)| \le O(re^{-L})$. We will show that for sufficiently large polynomial L, the distribution $(x, f(x))$ (say $\mathcal{D}_1$) is indistinguishable from (x, y) where $y \in \{0, 1\}$ such that $\mathbb{E}[y|x] = g(x)$ (say $\mathcal{D}_2$). Note that g can be greater than 1 but it can be easily handles by subtracting a small constant term of $O(re^{-L})$. Subsequently, we will use the algorithm for learning g to learn f up to small error.

Observe that both $\mathcal{D}_1$ and $\mathcal{D}_2$ have same marginal distribution on x and the label differs with probability $\le O(re^{-L})$. If $L = Cn$ for large enough constant $C > 0$ then $O(re^{-L}) \sim e^{-\Omega(n)}$ (exponentially small) as long as r is polynomial in n. Thus no polynomial time algorithm can distinguish between $\mathcal{D}_1$ and $\mathcal{D}_2$. Thus we can learn g within ϵ in squared loss. Since g and f are point-wise close with exponentially small error, using Lemma 39, we can small get 0-1 error.

Appendix C. Related work on Hardness of Learning Neural Networks

Using the fact that the difference of two ReLUs can compute a halfspace, Livni et al. (2014) proved that neural networks with two nonlinear layers (with say all ReLU activations) can compute an intersection of halfspaces, a function class that is known to be cryptographically hard (Klivans and Sherstov (2009)). Networks with one nonlinear layer of sigmoids (with polynomially large weights) feeding into the σ_{OR} output gate can also compute intersections of halfspaces. Note that we *can* efficiently learn such networks if the sigmoids in the first layer have unit norm.

Livni et al. (2014) proved the hardness of learning one nonlinear-layer neural networks but in a model where the learner sees the sign of the real-valued output of the network (i.e., they implicitly allow the nonlinear output activation of sign). Here we prove hardness results for learning one nonlinear layer of ReLUs in the realizable model (no noise, and the learner sees the output of the network rather than its sign).

Appendix D. Interpretability of the Hypothesis

Our learning algorithm outputs the hidden layer as a multivariate polynomial. Given inputs $\mathbf{x_1}, \cdots, \mathbf{x_m}$, the hypothesis output by our algorithm Alphatron is of the form $h(\mathbf{x}) = u(\sum_{i=1}^m \alpha_i^* \mathcal{MK}_d(\mathbf{x}, \mathbf{x_i})) = u(\langle \mathbf{v}, \psi_d(\mathbf{x}) \rangle)$ where $\mathbf{v} = \sum_{i=1}^m \alpha_i^* \psi_d(\mathbf{x_i})$ and d is dependent on required approximation. As seen in the preliminaries, $\langle \mathbf{v}, \psi_d(\mathbf{x}) \rangle$ can be expressed as a polynomial and the coefficients can be computed as follows,

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

Here, we follow the notation from Goel et al. (2016); M maps ordered tuple $(k_1, \ldots, k_j) \in [n]^j$ for $j \in [d]$ to tuple $(i_1, \ldots, i_n) \in \{0, \ldots, d\}^n$ such that $x_{k_1} \cdots x_{k_j} = x_1^{i_1} \cdots x_n^{i_n}$ and C maps ordered tuple $(i_1, \ldots, i_n) \in \{0, \ldots, d\}^n$ to the number of distinct orderings of the i_j's for $j \in \{0, \ldots, n\}$. The function C can be computed from the multinomial theorem (cf. Wikipedia (2016)). Thus, the coefficients of the polynomial can be efficiently indexed. Informally, each coefficient can be interpreted as the correlation between the target function and the product of features appearing in the coefficient's monomial.

Heuristics for Extending to Deep Nets. The invention of alternatives to gradient descent is an important goal in deep learning (there are simple examples of shallow convolutional nets where stochastic gradient descent is outperformed by other algorithms (Goel et al. (2018))). While our theoretical results only hold for networks with two nonlinear layers, there are several plausible heuristics for applying our method to deep nets by combining our approach with backpropagation. The simplest one is to simply train all but the last layer of a network using gradient descent and then apply Alphatron to learn the weights of the last one or two layers. Our initial experiments have shown this to work at least as well as traditional backpropagation on the whole network (the improvements decay with the depth of the net).

Other works have tried to combine kernel methods with deep learning through multilayer kernel machines(Cho and Saul (2009)) and convolutional kernel networks (Mairal et al. (2014)). A more ambitious approach that has been independently studied by research in applied machine learning is to treat the kernel function as an activation (Scardapane et al. (2017)), that is, $\sigma(a) = \sum_{i=1}^{D} \alpha_i \mathcal{K}(a, d_i)$ where d_i fixed dictionary elements and α_i are trained. Figuring out the "right" heuristic to combine Alphatron with traditional neural network training is an intriguing open problem and the subject of forthcoming work by the authors.

Appendix E. Multiple Instance Learning

Multiple Instance Learning (MIL) is a generalization of supervised classification in which a label is assigned to a *bag*, that is, a set of instances, instead of an individual instance (Dietterich et al. (1997)). The bag label is induced by the labels of the instances in it. The goal we focus on in this work is to label future bags of instances correctly, with high probability. We refer the reader to Amores (2013); Herrera et al. (2016) for an in-depth study of MIL. In this section we apply the previously developed ideas to MIL and give the first provable learning results for concrete schemes that do not rely on unproven assumptions.

Comparison to Previous Work. Under the standard MI assumption, various results are known in the PAC learning setting. Blum and Kalai (1998) showed a simple reduction from PAC learning MIL to PAC learning with one-sided noise under the assumption that the instances in each bag were drawn independently from a distribution. Sabato and Tishby (2012) removed the independence assumption and gave sample complexity bounds for learning future bags. All the above results require the existence of an algorithm for PAC learning with one-sided noise, which is itself a challenging problem and not known to exist for even simple concept classes.

In this work, we do not assume instances within each bag are independently distributed, and we do not require the existence of PAC learning algorithms for one-sided noise. Instead, we give efficient algorithms for labeling future bags when the class labeling instances is an unknown halfspace with a margin or an unknown depth-two neural network. We succeed with respect to general monotone, smooth combining functions.

Notation. Let us denote the space of instances as $\mathcal{X}$ and the space of bags as $\mathfrak{B} \subseteq \mathcal{X}^*$. Let N be an upper bound on the size of the bags, that is, $N = \max_{\beta \in \mathfrak{B}} |\beta|$. Let the instance labeling function be $c : \mathcal{X} \to \mathbb{R}$ and the bag labeling function be f_{bag}. We assume a distribution $\mathcal{D}$ over the bags and allow the instances within the bag to be dependent on each other. We consider two variants of the relationship between the instance and bag labeling functions and corresponding learning models.

Probabilistic MIL. We generalize the deterministic model to allow the labeling function to induce a probability distribution over the labels. This assumption seems more intuitive and less restrictive than the deterministic case as it allows for noise in the labels.

Definition 41 (Probabilistic MI Assumption) *Given combining function $u : \mathbb{R} \to [0, 1]$, for bag β, $f_{\mathsf{bag}}(\beta)$ is a random variable such that $Pr[f_{\mathsf{bag}}(\beta) = 1] = u\left(\frac{1}{|\beta|} \cdot \sum_{\boldsymbol{x} \in \beta} c(\boldsymbol{x})\right)$ where c is the instance labeling function.*

Definition 42 (Probabilistic MIL) *The concept class $\mathcal{C}$ is (ϵ, δ)-Probabilistic MIL for u with sample complexity M and running time T if under the probabilistic MI assumption for u, there exists an algorithm $\mathcal{A}$ such that for all $c \in \mathcal{C}$ as the instance labeling function and any distribution $\mathcal{D}$ on $\mathfrak{B}$, $\mathcal{A}$ draws at most M iid bags and runs in time at most T to return a bag-labeling hypothesis h such that with probability $1 - \delta$,*

$$\mathbb{E}_{\beta \sim \mathcal{D}}\left[\left(h(\beta) - u\left(\frac{1}{|\beta|} \cdot \sum_{\boldsymbol{x} \in \beta} c(\boldsymbol{x})\right)\right)^2\right] \leq \epsilon.$$

The following is our main theorem of learning in the Probabilistic MIL setting.

Theorem 43 *The concept class $\mathcal{C}$ is (ϵ, δ)-Probabilistic MIL for monotone L-Lipschitz u with sample complexity $(\frac{BL}{\epsilon})^2 \cdot \log(1/\delta)$ and running time $\mathsf{poly}(n, B, L, 1/\epsilon, \log(1/\delta))$ if all $c \in \mathcal{C}$ are $(\epsilon/CL, B)$-uniformly approximated by some kernel $\mathcal{K}$ for large enough constant $C > 0$.*

Combining Theorem 43 with Theorems 6 and 14 we can show the following polynomial time Probabilistic MIL results.

Corollary 44 *For any monotone L-Lipschitz function u, the concept class of sigmoids over $\mathbb{S}^{n-1}$ are (ϵ, δ)-Probabilistic MIL with sample complexity and running time $\mathsf{poly}(n, L, 1/\epsilon, \log(1/\delta))$.*

Corollary 45 *For any monotone L-Lipschitz function u, the concept class of halfspaces with a constant margin over $\mathbb{S}^{n-1}$ are (ϵ, δ)-Probabilistic MIL with sample complexity and running time $\mathsf{poly}(n, L, 1/\epsilon, \log(1/\delta))$.*