Efficient Algorithms for Learning Depth-2 Neural Networks with General ReLU Activations

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present polynomial time and sample efficient algorithms for learning an unknown depth-2 feedforward neural network with *general* ReLU activations, under mild non-degeneracy assumptions. In particular, we consider learning an unknown network of the form $f(x) = a^\mathsf{T}\sigma(W^\mathsf{T}x + b)$, where x is drawn from the Gaussian distribution, and $\sigma(t) \coloneqq \max(t, 0)$ is the ReLU activation. Prior works for learning networks with ReLU activations assume that the bias b is zero. In order to deal with the presence of the bias terms, our proposed algorithm consists of robustly decomposing multiple higher order tensors arising from the Hermite expansion of the function $f(x)$. Using these ideas we also establish identifiability of the network parameters under minimal assumptions.

1 Introduction

The empirical success of deep learning in recent years has led to a flurry of recent works exploring various theoretical aspects of deep learning such as learning, optimization and generalization. A fundamental question in the theory of deep learning is to identify conditions under which one can design provably time-efficient and sample-efficient learning algorithms for neural networks. Perhaps surprisingly, even for the simplest case of a depth-2 feedforward neural network, the learning question remains unresolved. In this work we make progress on this front by studying the problem of learning an unknown neural network of the form

$$y = f(x) = a^\mathsf{T}\sigma(W^\mathsf{T}x + b). \tag{1}$$

We are given access to a finite amount of samples of the form (x_i, y_i) drawn i.i.d. from the data distribution, where each x_i is comes from the standard Gaussian distribution $N(0, I)$, and $y_i = f(x_i)$. The goal is to design an algorithm that outputs an approximation of the function f up to an arbitrary error measured in the expected mean squared metric (squared ℓ_2 loss). An efficient learning algorithm has running time and sample complexity that are polynomial in the different problem parameters such as the input dimensionality, number of hidden units, and the desired error.

Without any further assumptions on the depth-2 network, efficient learning algorithms are unlikely. The recent work of [DKKZ20] provides evidence by proving exponential statistical query lower bounds (even when x is Gaussian) that rule out a broad class of algorithms.

Several recent works have designed efficient algorithms for depth-2 neural networks in the special setting when the bias term $b = 0$. One prominent line of work [GLM17, BJW19] give polynomial time algorithms under the non-degeneracy assumption that the matrix W has full-rank. Another body of work relaxes the full-rank assumption by designing algorithm that incur an exponential dependence on the number of hidden units [DKKZ20, CKM20, GK19]. There is little existing literature on learning neural networks in the presence of the bias term. A notable exception is an approach based on computing "score functions" [JSA15] that applies to certain activations with bias and requires

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

various assumptions that are not satisfied by the ReLU function. The diminished expressivity of neural networks without the bias terms leads to the following compelling question:

Can we design polynomial time algorithms even in the presence of bias terms in the ReLU units?

We answer the question affirmatively by designing efficient algorithms for depth-2 neural networks with general ReLU activations, under the assumption that $W_{d\times m}$ has linearly independent columns (hence $m \leq d$). In fact, our algorithms can be extended to work under much weaker assumptions on W, that allow for $m \gg d$ ($m \leq O(d^\ell)$ for any constant $\ell \geq 1$) in a natural smoothed analysis setting considered in prior works [GKLW18] (see Theorem 3.1 and Corollary 3.2). An important consequence of our techniques is the fact that the network parameters are identifiable as long as no two columns of W are parallel, and all the $\{a_i\}$ are non-zero.

Conceptual and technical challenges with bias terms. Similar to prior works [GKLW18, JSA15], our techniques rely on the use of tensor decomposition algorithms to recover the parameters of the network. In the absence of any bias, it can be shown that the 4th Hermite coefficient of the function $f(x)$ takes the form $\hat{f}_4 = \sum_{i=1} a_i w_i^{\otimes 4}$ where w_i are the columns of W and $a = (a_1, a_2, \ldots,)$. When the columns of W are linearly independent, existing algorithms for tensor decompositions in the full-rank setting can be used to recover the parameters [Har70, JGKA19]. However, when bias terms are present, there are several challenges that we highlight below.

In the presence of biases the kth Hermite expansion of $f(x)$ takes the form $\hat{f}_k = \sum_{i=1}^m a_i g_k(b_i) w_i^{\otimes k}$, where g_k is a function that may vanish on some of the unknown b_i parameters. This creates a hurdle in recovering the corresponding w_i. A simple example where the above approach fails is when some of the $b_i = \pm 1$, since the corresponding rank-1 terms vanish from the decomposition of $\hat{f}_4$. To overcome this obstacle, we first show give a precise expression for the function $g_k(b_i)$ involving the $(k-2)$th Hermite polynomial (see Lemma 3.4). We then design an algorithm that decomposes multiple tensors obtained from Hermite coefficients to recover the parameters. We use various properties of Hermite polynomials to analyze the algorithm e.g., the separation of roots of consecutive Hermite polynomials is used to argue that each w_i is recovered from decomposing at least one of the tensors.

Secondly, in the presence of the bias terms, recovery of all the parameters may not even be possible from polynomially many samples. For instance, consider a particular hidden node with output $\sigma(w_i^\top x + b_i)$. If b_i is a large positive number then it behaves like a linear function (always active). Hence if multiple b_is are large positive numbers then one can only hope to recover a linear combination of their corresponding weights and biases. On the other hand if b_i is a large negative constant then the activation is 0 except with probability exponentially small in $|b_i|$. We cannot afford a sample complexity that is exponential in the magnitude of the parameters. We handle this by proceeding in two stages. In the first stage we recover the network parameters of all the "good" terms, i.e., hidden units with biases of small magnitude. To handle the "bad" terms (large magnitude bias) we show that a linear functions is a good approximation to the residual function comprising of the bad terms.

There are several other technical challenges that arise in the analysis sketched above, when there are sampling errors due to using only a polynomial amount of data (for example, the tensors obtained from $\hat{f}_k$ may have some rank-1 terms that are small but not negligible, that may affect the robust recovery guarantees for tensor decompositions). We obtain our robust guarantees by leveraging many useful properties of Hermite polynomials, and a careful analysis of how the errors propagate.

The rest of the paper is organized as follows. We present preliminaries in Section 1.1 followed by related work in Section 2. We then formally present and discuss our main results in Section 3. In Section 4 we present our main algorithm and analysis in the population setting, i.e., under the assumption that one has access to infinite data from the distribution. We then present the finite sample extension of our algorithm in Section 5 that achieves polynomial runtime and sample complexity.

1.1 Model Setup and Preliminaries

We consider the supervised learning problem with input $x \in \mathbb{R}^d$ drawn from a standard d-dimensional Gaussian distribution $\mathcal{N}(0, I_{d\times d})$ and labels y generated by a neural network $y = f(x) = a^\mathsf{T}\sigma(W^\mathsf{T}x + b)$, where $a, b \in \mathbb{R}^m$, $W \in \mathbb{R}^{d\times m}$ and σ is the element-wise ReLU activation function, i.e., $\sigma(t) = \max(t, 0)$. We denote the column vectors of W as $w_i \in \mathbb{R}^d$ with $i \in [m]$ and a_i as the i'th element of vector a, similarly for b and x. We pose a constraint on magnitudes of a, b, W

such that they are all B-bounded for some $1 \leq B \leq \text{poly}(m, d)$, i.e. $\|a\|_\infty, \|b\|_\infty, \|W\|_\infty \leq B$, and $\min_{i\in[m]} |a_i| \geq 1/B$. Furthermore, we assume $\|w_i\|_2 = 1$ without loss of generality. If w_i are not unit vectors, we can always scale a_i and b_i to $\|w_i\|a_i$ and $\frac{b_i}{\|w_i\|}$ respectively so that w_i are normalized. We will denote by $\Phi()$ the cumulative density function (CDF) of the standard Gaussian distribution. Finally, for a matrix M, we will use $s_k(M)$ to denote the kth largest singular value of M.

Hermite polynomials and Hermite coefficients. The d-dimensional or d-variate Hermite polynomials of the kth order evaluated at a point x represented by the tensor $He_k(x) \in (\mathbb{R}^d)^{\otimes k}$ with $k \in \mathbb{N}$ (here $\mathbb{N}$ includes 0) form a complete orthogonal basis for the weighted $L^2(\mathbb{R}^d, \gamma)$ space with inner product defined as $\mathbb{E}[f(x)g(x)] = \int_{\mathbb{R}^d} f(x)g(x)\gamma(x)dx$ for two functions f, g, where $\gamma(x) = \exp(-\|x\|^2/2)/\sqrt{2\pi}$. We can thus define the k'th Hermite coefficient of $f(x)$ by $\hat{f}_k = \mathbb{E}[f(x)He_k(x)]$, similar to how the Fourier coefficients are constructed. Throughout the context we wll also use $He_k^\alpha(x) \in \mathbb{R}$ to denote a specific entry of $He_k(x)$ with $\alpha \in [d]^k$ as a multi-index if $d > 1$. More details and facts about Hermite polynomials are included in Appendix A.1.

Tensor decomposition. The tensor product $u \otimes v \otimes w \in \mathbb{R}^{d_1} \otimes \mathbb{R}^{d_2} \otimes \mathbb{R}^{d_3}$ of vectors $u \in \mathbb{R}^{d_1}, v \in \mathbb{R}^{d_2}, w \in \mathbb{R}^{d_3}$ is a rank-1 tensor. Similarly, we will use $u^{\otimes \ell} \in (\mathbb{R}^d)^{\otimes \ell}$ to denote the tensor product of u with itself ℓ times. An order-t tensor $T \in \mathbb{R}^{d_1} \otimes \mathbb{R}^{d_2} \otimes \cdots \otimes \mathbb{R}^{d_t}$ is represented using a t-way array $\mathbb{R}^{d_1 \times d_2 \times \cdots \times d_t}$ that has t modes corresponding to the t different indices. Given two matrices U, V with k columns each given by $U = (u_i : i \in [k])$ and $V = (v_u : i \in [k])$, the *Khatri-Rao product* $M = U \odot V$ is a matrix formed by the ith column being $u_i \otimes v_i$. We will also use $U^{\odot 2} = U \odot U$ (and similarly for higher orders). A claim about preserving the full-column-rank property (and analogously minimum singular value) under the Khatri-Rao product is included in the Appendix A.2.

Flattening or Reshaping: Given an order-t tensor T, for $t_1, t_2, t_3 \geq 0$ such that $t_1 + t_2 + t_3 = t$ define $T' = \text{flatten}(T, t_1, t_2, t_3)$ as the order-3 tensor $T' \in \mathbb{R}^{d^{t_1} \times d^{t_2} \times d^{t_3}}$, obtained by flattening and combining the first t_1 modes, the next t_2 and last t_3 modes respectively. When $t_3 = 0$, the output is a matrix in $\mathbb{R}^{d^{t_1} \times d^{t_2}}$.

Tensor decompositions of order 3 and above, unlike matrix decompositions (which are of order 2) are known to be unique under mild conditions. While tensor decompositions are NP-hard in the worst-case, polynomial time algorithms for tensor decompositions are known under certain non-degeneracy conditions (see e.g., [JGKA19, Vij20]). In particular, Jennrich's algorithm [Har70] provides polynomial time guarantees for recovering all the rank-1 terms of a decomposition of a tensor $T = \sum_{i=1}^k u_i \otimes v_i \otimes w_i$, when the $\{u_i : i \in [k]\}$ are linearly independent, the $\{v_i : i \in [k]\}$ are linearly independent, and no two of the vectors $\{w_i : i \in [k]\}$ are parallel. This algorithm and its guarantee can also be made robust to some noise (of an inverse polynomial magnitude), when measured in Frobenius norm. See Appendix A.2 for details.

2 Related Work

By now there is a vast literature exploring various aspects of deep learning from a theoretical perspective. Here we discuss the works most relevant in the context of our results. As discussed earlier, the recent works of [GLM17, BJW19, GKLW18] provide polynomial time algorithms for learning depth-2 feedforward ReLU networks under the assumption that the input distribution is Gaussian and that the matrix W is full rank. However, these works do not extend to the case of non-zero bias. The work of [JSA15] proposed a general approach based on tensor decompositions for learning an unknown depth-2 neural network that could also handle the presence of the bias terms. The tensor used in the work of [JSA15] is formed by taking the weighted average of a "score" function evaluated on each data point. In this way their approach generalizes to a large class of distributions provided one has access to the score function. However, for most data distributions computing the score function is a hard task itself. When the input distribution is Gaussian, then the score functions correspond to the Hermite coefficients of the target function $f(x)$ and can be evaluated efficiently. However, the analysis in [JSA15] does not extend to the case of ReLU activations for several reasons. Their technique needs certain smoothness and symmetry assumptions on the activations that do not hold for ReLU. These assumptions also ensure that all the terms in the appropriate tensor are non-zero. We do not make such assumptions, and tackle one of the main challenges by showing that one can indeed recover a good approximation to the network by analyzing multiple higher order tensors. Furthermore, the authors in [JSA15] assume that the biases, and the spectral norm of W are

both bounded by a constant. As a result they do not handle the case of biases of large magnitude where some of the ReLU units mostly function as linear functions (with high probability).

There have also been works on designing learning algorithms for neural networks without assumptions on the linear independence of columns of W. These results incur an exponential dependence on either the input dimensionality or the number of parameters in the unknown network [DKKZ20, CKM20]. In particular, the result of [CKM20] provides a learning algorithm for arbitrary depth neural networks under the Gaussian distribution with an "FPT" guarantee; its running time is polynomial in the dimension, but exponential in the number of ReLU units. Given the recent statistical query lower bounds on learning deep neural networks [DKKZ20, GGJ+20], getting a fully polynomial time algorithm without any assumptions is a challenging open problem, even under Gaussian marginals.

Polynomial time algorithms with fewer assumptions and beyond depth-2 can be designed if the activation functions in the first hidden layer are sigmoid functions [GK19]. Finally, there is also extensive literature on analyzing the convergence properties of gradient descent and stochastic gradient descent for neural networks. The results in this setting implicitly or explicitly assume that the target function is well approximated in the Neural Tangent Kernel (NTK) space of an unknown network. Under this assumption these results show that gradient descent on massively overparameterized neural networks can learn the target [APVZ14, Dan17, DFS16, AZLS19, DZPS18, ADH+19, LXS+19, CB18, JGH18].

3 Main Results

There are two related but different goals that we consider in learning the ReLU network:

- *Approximating f:* Output a ReLU network $\widetilde{f}(x) = \widetilde{a}^\top \sigma(\widetilde{W}^\top x + \widetilde{b})$ that ε-approximates f for some $\varepsilon > 0$ i.e., $\forall x \in \mathbb{R}^n, |f(x) - \widetilde{f}(x)| \le \varepsilon \|x\|_2 + \varepsilon$.[1]
- *Parameter recovery:* Output $\widetilde{W}, \widetilde{a}, \widetilde{b}$, such that each parameter is ε-close (up to permuting the m co-ordinates of $a, b \in \mathbb{R}^m$ and reordering the corresponding columns of W).

We remark that the second goal is harder and implies the first; in particular, when $\varepsilon = 0$, the second goal corresponds to identifiability of the model. However in some cases, parameter recovery may be impossible to achieve with polynomially many samples even though we can achieve the goal of approximating f. As we have seen earlier, given N samples if $b_i \gg \sqrt{\log N}$, then $\sigma(w_i^\top x + b_i)$ will be indistinguishable from the linear function $w_i^\top x + b$ w.h.p.; hence if there are multiple such $b_i \in [m]$ with large magnitude, the best we can hope to do is recover the sum of all those linear terms. Our first result shows that this is the only obstacle when we are in the *full-rank* or *undercomplete* setting i.e., $\{w_i : i \in [m]\}$ are linearly independent (in a robust sense). Moreover even when there are b_i of large magnitude, we can learn a function $\tilde{f}$ that approximates f.

In fact, our algorithm and guarantees are more general, and can operate under the much milder condition that $\{w_i^{\otimes \ell} : i \in [m]\}$ are linearly independent for any constant $\ell \ge 1$; the setting when $\ell > 1$ corresponds to what is often called the *overcomplete setting*. In what follows, for any constant $\ell \in \mathbb{N}$ we use $\mathrm{poly}_\ell(n_1, n_2, \dots)$ to denote a polynomial dependency on $n_1, n_2, \dots$, and potentially exponential dependence on ℓ.

Theorem 3.1. *Suppose $\ell \in \mathbb{N}$ be a constant, and $\varepsilon > 0$. If we are given N i.i.d. samples as described above from a ReLU network $f(x) = a^\top \sigma(W^\top x + b)$ that is B-bounded and if W satisfies $s_m(W^{\odot \ell}) \ge 1/\mathrm{poly}(m, d)$, then there is an algorithm that given $N \ge \mathrm{poly}_\ell(m, d, 1/\varepsilon, B)$ runs in $\mathrm{poly}_\ell(N, m, d)$ time and with high probability finds a ReLU network $\widetilde{f}(x) = \widetilde{a}^\top \sigma(\widetilde{W}^\top x + \widetilde{b})$ that ε-approximates f. Furthermore there are constants $c = c(\ell) > 0, c' > 0$ such that for all $i \in [m]$ with $|b_i| < c\sqrt{\log(1/(\varepsilon \cdot mdB))}$ we have that $\|w_i - \widetilde{w}_i\|_2 + |a_i - \widetilde{a}_i| + |b_i - \widetilde{b}_i| < c'\varepsilon/(mB)$.*

The special case of $\ell = 1$ in Theorem 3.1 corresponds to the *full-rank setting*. Here the condition on the least singular value $s_m(W)$ necessitates that $m \le d$. But for $\ell > 1$, we only require that the set of vectors $w_1^{\otimes \ell}, w_2^{\otimes \ell}, \dots, w_m^{\otimes \ell}$ are linearly independent (in a robust sense), which one can expect for much larger values of m typically. The following corollary formalizes this in the smoothed analysis framework of Spielman and Teng [ST04], which is a popular paradigm for reasoning

[1] An approximate $\widetilde{f}$ also implies a small expected squared loss under the data distribution (see Section C.3).

about non-worst-case instances [Rou20]. Combining the above theorem with existing results on smoothed analysis [BCPV19] implies polynomial time guarantees for non-degenerate instances with $m = O(d^\ell)$ for any constant $\ell > 0$. Below, $\widehat{W}$ denotes the columns of W are τ-smoothed i.e., randomly perturbed with standard Gaussian of average length τ that is at least inverse polynomial (See Section D for the formal smoothed analysis model and result).

Corollary 3.2 (Smoothed Analysis)**.** *Suppose $\ell \in \mathbb{N}$ and $\varepsilon > 0$ are constants in the smoothed analysis model with smoothing parameter $\tau > 0$, and also assume the ReLU network $f(x) = a^\top \sigma(\widehat{W}^\top x + b)$ is B-bounded with $m \leq 0.99\binom{d+\ell-1}{\ell}$. Then there is an algorithm that given $N \geq \mathrm{poly}_\ell(m, d, 1/\varepsilon, B, 1/\tau)$ samples runs in $\mathrm{poly}_\ell(N, m, d)$ time and with high probability finds a ReLU network $\widetilde{f}(x) = \widetilde{a}^\top \sigma(\widetilde{W}^\top x + \widetilde{b})$ that ε-approximates f i.e., $\forall x \in \mathbb{R}^d$, $|f(x) - \widetilde{f}(x)| \leq \varepsilon \|x\|_2 + \varepsilon$. Furthermore there are constants $c, c' > 0$ such that for all $i \in [m]$ such that $|b_i| < c\sqrt{\log(1/(\varepsilon \cdot mdB))}$ we have that $\|w_i - \widetilde{w}_i\|_2 + |a_i - \widetilde{a}_i| + |b_i - \widetilde{b}_i| < c'\varepsilon/(mB)$.*

While our algorithm and analysis give guarantees that are robust to sampling errors and inverse polynomial error, even the non-robust analysis has implications and, implies identifiability of the model as long as no two rows of W are parallel. Note that in general identifiability may not imply any finite sample complexity bounds.

Theorem 3.3 (Identifiability)**.** *Suppose we are given samples from a ReLU network $f(x) = a^\top \sigma(W^\top x + b)$ where all $\min_{i \in [m]} |a_i| > 0$ and no two columns of W are parallel to each other. Then the parameters of the model W, a, b are uniquely identified from samples.*

Our results rely on the precise expressions for higher order Hermite coefficients of $f(x)$ given below.

Lemma 3.4. *Let $\hat{f}_k = \mathbb{E}_{x \sim \mathcal{N}(0,I)}[f(x) He_k(x)]$ (with $k \in \mathbb{N}$) be the k'th Hermite coefficient (this is an order-k tensor) of $f(x) = a^\top \sigma(W^\top x + b)$. Then*

$$\hat{f}_0 = \sum_{i=1}^{m} a_i \Big(b_i \Phi(b_i) + \frac{\exp(-\frac{b_i^2}{2})}{\sqrt{2\pi}} \Big) \ , \ \hat{f}_1 = \sum_{i=1}^{m} a_i \Phi(b_i) w_i \tag{2}$$

$$\forall k \geq 2, \ \ \hat{f}_k = \sum_{i=1}^{m} (-1)^k \cdot a_i \cdot He_{k-2}(b_i) \cdot \frac{\exp(\frac{-b_i^2}{2})}{\sqrt{2\pi}} \cdot w_i^{\otimes k} \tag{3}$$

We prove this by considering higher order derivatives and using properties of Hermite polynomials. A key property we use here is that the k'th derivative of a standard Gaussian function is itself multiplied by the k'th Hermite polynomial (with sign flipped for odd k). This significantly simplifies the expression for the coefficient $g(b_i)$ of $w_i^{\otimes k}$.We remark that the above lemma may also be used to give an expression for the training objective for depth-2 ReLU networks, analogous to the result of [GLM17] for ReLU activations with no bias, that provides an expression as a combination of tensor decomposition problems of increasing order. See Appendix B.1 for details.

4 Non-robust Algorithm and Analysis

Our algorithms for learning the parameters of $f(x) = a^\top \sigma(W^\top x + b)$ decompose tensors obtained from the Hermite coefficients $\{\hat{f}_t \in (\mathbb{R}^d)^{\otimes t}\}$ of the function f. In this section, we design an algorithm assuming that we have access to all the necessary Hermite coefficients exactly (no noise or sampling errors). This will illustrate the basic algorithmic idea and the identifiability result. However with polynomial samples, we can only hope to estimate these quantities up to inverse polynomial accuracy. In Section 5 we describe how we deal with the challenges that arise from errors.

Theorem 4.1. *Suppose the parameters $\{(a_i, b_i, w_i) : i \in [m]\}$ satisfies:*
(i) *no two $\{w_i : i \in [m]\}$ are linearly dependent and,* (ii) *for a constant $\ell \in \mathbb{N}$, $\{w_i^{\otimes \ell} : i \in [m]\}$ are linearly independent,* (iii) *$a_i \neq 0$ for all $i \in [m]$. Then given $\{\hat{f}_t : 0 \leq t \leq 2\ell + 2\}$ exactly, Algorithm 1 recovers (with probability 1) the unknown parameters a, b and W in $\mathrm{poly}_\ell(m, d)$ time.*

We now describe the algorithm for general $\ell \geq 1$ (this specializes to the full-rank setting for $\ell = 1$).

Subroutine Alg. 2 finds the unknown parameters $a_1, \ldots, a_m \in \mathbb{R}$ and $b_1, \ldots, b_m \in \mathbb{R}$ given $w_1, w_2, \ldots, w_m$. While Alg. 1 changes a little when we have errors in the estimates, the subroutine Alg. 2 remains the same even for the robust version of the algorithm.

Algorithm 1: for order ℓ: recover a, b, W given $\{\hat{f}_t : 0 \le t \le 2\ell+2\}$

Input: $\hat{f}_\ell, \hat{f}_{\ell+1}, \hat{f}_{\ell+2}, \hat{f}_{\ell+3}, \hat{f}_{2\ell+1}, \hat{f}_{2\ell+2}$;

1. Let $T' = \text{flatten}(\hat{f}_{2\ell+1}, \ell, \ell, 1) \in \mathbb{R}^{d^\ell \times d^\ell \times d}$ and $T'' = \text{flatten}(\hat{f}_{2\ell+2}, \ell, \ell, 2) \in \mathbb{R}^{d^\ell \times d^\ell \times d^2}$ be order-3 tensors obtained by flattening $\hat{f}_{2\ell+1}$ and $\hat{f}_{2\ell+2}$.
2. Set $k' = \text{rank}(\text{flatten}(\hat{f}_{2\ell+1}, \ell, \ell+1, 0))$. Run Jennrich's algorithm [Har70] on T' to recover rank-1 terms $\{\alpha'_i u_i^{\otimes \ell} \otimes u_i^{\otimes \ell} \otimes u_i \mid i \in [k']\}$, where $\forall i \in [k'],\ u_i \in \mathbb{S}^{d-1}$ and $\alpha'_i \in \mathbb{R}$.
3. Set $k'' = \text{rank}(\text{flatten}(\hat{f}_{2\ell+2}, \ell, \ell+1, 0))$. Run Jennrich's algorithm [Har70] on T'' to recover rank-1 terms $\{\alpha''_i v_i^{\otimes \ell} \otimes v_i^{\otimes \ell} \otimes v_i^{\otimes 2} \mid i \in [k'']\}$, where $\forall i \in [k''],\ v_i \in \mathbb{S}^{d-1}$ and $\alpha''_i \in \mathbb{R}$.
4. Remove duplicates from $\{u_1, u_2, \ldots, u_{k'}\} \cup \{v_1, v_2, \ldots, v_{k''}\}$ to get $\widetilde{w}_1, \widetilde{w}_2, \ldots, \widetilde{w}_m$.
5. Run subroutine RECOVERSCALARS$(m, \ell, \{\widetilde{w}_i : i \in [m]\}, \hat{f}_\ell, \hat{f}_{\ell+1}, \hat{f}_{\ell+2}, \hat{f}_{\ell+3})$ to get $\{\widetilde{a}_i, \widetilde{b}_i : i \in [m]\}$.

Result: Output $\{\widetilde{w}_i, \widetilde{a}_i, \widetilde{b}_i : i \in [m]\}$

Algorithm 2: Subroutine RECOVERSCALARS to recover a, b given W.

Input: $m, \ell, (\widetilde{w}_i : i \in [m])$, and tensors $T_\ell, T_{\ell+1}, T_{\ell+2}, T_{\ell+3}$ which are tensors of orders $\ell, \ell+1, \ell+2, \ell+3$ respectively;

if $\ell = 1$ **then**

1. For $j \in \{1, 2, 3\}$, solve the system of linear equation $\sum_{i=1}^m \zeta_j(i) \mathsf{vec}(\widetilde{w}_i^{\otimes j}) = \mathsf{vec}(T_j)$ to recover unknowns $\{\zeta_j(i) \mid i \in [m]\}$;
2. For each $i \in [m]$, set $b_i = -\frac{\zeta_3(i) + 2\zeta_1(i)}{\zeta_2(i)}$, and $a_i = \zeta_2(i) \cdot \sqrt{2\pi} e^{b_i^2/2}$.

else

1. For $j \in \{\ell, \ell+1, \ell+2, \ell+3\}$, solve the system of linear equation $\sum_{i=1}^m \zeta_j(i) \mathsf{vec}(\widetilde{w}_i^{\otimes j}) = \mathsf{vec}(T_j)$ to recover unknowns $\{\zeta_j(i) \mid i \in [m]\}$;
2. For each $i \in [m]$, $q_i := \text{argmax}_{j \in \{\ell+1, \ell+2\}} |\zeta_j(i)|$, set $\widetilde{b}_i = -\frac{\zeta_{q+1}(i) + q \cdot \zeta_{q-1}(i)}{\gamma_q(i)}$, and $\widetilde{a}_i = \sqrt{2\pi}(-1)^q \zeta_q(i) e^{b_i^2/2} / He_q(b_i)$, as described in (6) (Lemma 4.4).

end

Result: Output $(\widetilde{a}_i, \widetilde{b}_i : i \in [m])$

Algorithm 1 decomposes two different tensors obtained from consecutive Hermite coefficients $\hat{f}_{2\ell+1}, \hat{f}_{2\ell+2}$ to obtain the $\{w_i\}$. This is because the bias b_i could make the coefficient of the ith term in the decomposition 0 (e.g., $He_{2\ell-1}(b_i) = 0$ for $\hat{f}_{2\ell+1}$); hence w_i cannot be recovered by decomposing $\hat{f}_{2\ell+1}$. Hence $\hat{f}_{2\ell+1}$ can degenerate to a rank $m' < m$ tensor, and Jennrich's algorithm will return only $m' < m$ eigenvectors that correspond to non-zero eigenvalues. The following lemma addresses this issue by showing that two consecutive Hermite polynomials can not both take small values at any point $x \in \mathbb{R}$. This implies a separation between roots of consecutive Hermite polynomials, and establishes a "robust" version that will be useful in Section 5.

Lemma 4.2 (Separation of Roots)**.** *For all* $k \in \mathbb{N}, x \in \mathbb{R}$, $\max\{|He_k(x)|, |He_{k+1}(x)|\} \ge \sqrt{k!/2}$.

Proof. First, $\forall k \in \mathbb{N}$, by Turán's inequality [Tur50] we have

$$He_{k+1}^2(x) - He_k(x) He_{k+2}(x) = k! \cdot \sum_{i=0}^{k} \frac{He_i(x)^2}{i!} > 0 \tag{4}$$

Set $\varepsilon = \sqrt{k!/2}$ and assume for contradiction that $|He_{k+1}(x)|, |He_{k+2}(x)| < \varepsilon$. The LHS of (4) is at most $\varepsilon^2 + \varepsilon |He_k(x)|$, and the RHS of (4) is at least

$$k! \cdot (1 + x^2 + \ldots + \frac{He_k(x)^2}{k!}) > k! \cdot (1 + \frac{He_k(x)^2}{k!}) \tag{5}$$

Therefore, if $|He_k(x)| = t$, combining both sides we get $\varepsilon^2 + \varepsilon t \ge k! + t^2$. This implies on the one hand that $\varepsilon t \ge k!/2$, and on the other hand that $\varepsilon t \ge t^2$. However for our choices of $\varepsilon = \sqrt{k!/2}$, no value of t is feasible. This yields the required contradiction. $\square$

The following Claim shows that Jennrich's algorithm for decomposing a tensor successfully recovers all the terms whose appropriate $He_k(b_i) \neq 0$. This claim along with Lemma 4.2 shows that Steps 2-3 of Algorithm 1 successfully recovers all the $\{w_i\}$.

Claim 4.3. *Let $\ell_1, \ell_2 \geq \ell$ and $T \in \mathbb{R}^{d^{\ell_1} \times d^{\ell_2} \times d^{\ell_3}}$ have a decomposition $T = \sum_{i=1}^m \alpha_i w_i^{\otimes \ell_1} \otimes w_i^{\otimes \ell_2} \otimes w_i^{\otimes \ell_3}$, with $\{w_i^{\otimes \ell} : i \in [m]\}$ being linearly independent. Consider matrix $M = \textit{flatten}(T, \ell_1, \ell_2 + \ell_3, 0) \in \mathbb{R}^{d^{\ell_1} \times d^{\ell_2 + \ell_3}}$, and let $r := rank(M)$. Then Jennrich's algorithm applied with rank r runs in $\mathrm{poly}_{\ell_1+\ell_2+\ell_3}(m, d)$ time recovers (w.p. 1) the rank-1 terms corresponding to $\{i \in [m] : |\alpha_i| > 0\}$.*

Proof. Let $Q = \{i \in [m] : |\alpha_i| > 0\}$. Firstly $r = rank(M) = |Q|$, since M has a decomposition

$$M = \sum_{i \in Q} \alpha_i \left(w_i^{\otimes \ell_1}\right)\left(w_i^{\otimes \ell_2 + \ell_3}\right)^\top = M_1 \mathrm{diag}(\alpha_Q) M_2^\top$$

where $M_1, M_2, \mathrm{diag}(\alpha_Q)$ all have full column rank $|Q|$. Secondly, from assumption (ii) of Theorem 4.1 and Claim A.4 applied with $\ell_1, \ell_2 \geq \ell$, we have that $\{w_i^{\otimes \ell'} : i \in [m]\}$ are linearly independent for every $\ell' \geq \ell$. Hence the the factor matrices $U = (w_i^{\otimes \ell_1} : i \in Q)$ and $V = (w_i^{\otimes \ell_2} : i \in Q)$ also have full column rank. Similarly from (iii) no two vectors in $\{\alpha_i w_i^{\otimes \ell_3} : i \in Q\}$ are parallel. Hence, they satisfy the conditions of Jennrich's algorithm. Since there is no error in the tensor, Jennrich's algorithm (Theorem A.5) succeeds with probability 1 (see [Vij20]). □

The following lemma is useful for recovering $\{a_i, b_i\}_{i \in [m]}$ once we have the $\{w_i\}_{i \in [m]}$, and uses various properties of Hermite polynomials along with Lemma 3.4 and Lemma 4.2.

Lemma 4.4. *Suppose $k \in \mathbb{N}, k \geq 2$. Suppose for some unknowns $\beta, z \in \mathbb{R}$ with $\beta \neq 0$, we are given values of $\gamma_j = (-1)^j \beta He_j(z)\ \forall j \in \{k, k+1, k+2, k+3\}$. Then z, β are uniquely determined by*

$$\textit{For } q := \operatorname*{argmax}_{j \in \{k+1, k+2\}} |\gamma_j|, \ \textit{ set } \ z = -\frac{\gamma_{q+1} + q \cdot \gamma_{q-1}}{\gamma_q}, \quad \beta = (-1)^q \frac{\gamma_q}{He_q(z)} \tag{6}$$

Proof. We use the following fact about Hermite polynomials:

$$He_{r+1}(z) = z He_r(z) - r \cdot He_{r-1}(z). \tag{7}$$

From Lemma 4.2, we know that $\max\{|He_{k+1}(z)|, |He_{k+2}(z)|\} > 0$ and hence $\gamma_q \neq 0$. Substituting in the recurrence (7) with $r = q$,

$$z = \frac{He_{q+1}(z) + q \cdot He_{q-1}(z)}{He_q(z)} = \frac{\beta He_{q+1}(z) + q \cdot \beta He_{q-1}(z)}{\beta He_q(z)} = -\frac{\gamma_{q+1} + q \cdot \gamma_{q-1}}{\gamma_q},$$

as required. The β value is also recovered since $\gamma_q = (-1)^q \beta He_q(z)$.

□

A robust version of this lemma (see Section C.2.2) will be important in the robust analysis of Section 5. The following claim applies the above lemma for each $i \in [m]$ with $z = b_i$ and $\beta = a_i e^{-b_i^2/2}/\sqrt{2\pi}$, to show that Step 5 of the algorithm recovers the correct $\{(a_i, b_i)\}_{i \in [m]}$ given the $\{w_i\}_{i \in [m]}$.

Claim 4.5. *If $\{\widetilde{w}_i : i \in [m]\} = \{w_i : i \in [m]\}$, Step 5 of Alg. 1 recovers $\{(a_i, b_i, w_i) : i \in [m]\}$.*

Proof. We first prove for $\ell \geq 2$. For each of the $j \in \{\ell, \ell+1, \ell+2, \ell+2\}$, we have from Lemma 3.4

$$\hat{f}_j = \sum_{i=1}^m (-1)^j \cdot a_i \cdot He_{j-2}(b_i) \cdot \frac{1}{\sqrt{2\pi}} \exp(-b_i^2/2) \cdot w_i^{\otimes j}.$$

Moreover the vectors $\{w_i^{\otimes j} : i \in [m]\}$ are linearly independent by assumption (and from Claim A.4 for $j > \ell$). Hence the linear system for each j has a unique solution

$$\forall \ell \leq j \leq \ell + 3, \ \forall i \in [m], \ \text{ we have } \zeta_j(i) = a_i \cdot \frac{(-1)^j}{\sqrt{2\pi}} \exp(-b_i^2/2)\ He_{j-2}(b_i).$$

Lemma 4.4 applied with $\beta = \frac{1}{\sqrt{2\pi}} a_i e^{-b_i^2/2}$ and $\gamma_{j-2} = (-1)^j He_{j-2}(b_i)$ (note $\beta \neq 0$) proves that Alg. 2 recovers a_i, b_i. For $\ell = 1$, we note $\forall z \in \mathbb{R},\ He_0(z) = 1$, and apply a similar argument. □

We now complete the proof of the non-robust analysis for any constant $\ell \geq 0$.

Proof of Theorem 3.1. The proof follows by combining Claim 4.3 and Claim 4.5, along with Lemma 4.2. Let $Q_1 = \{i : |He_{2\ell-1}(b_i)| > 0\}$ and $Q_2 = \{i : |He_{2\ell+1}(b_i)| > 0\}$. From Claim 4.3, Step 2 recovers all the rank-1 terms in Q_1 with probability 1; hence we obtain in particular $\{w_i \in \mathbb{S}^{d-1} \mid i \in Q_1\}$. Similarly, in Step 3 we recover w.p. 1 all the $\{w_i \in \mathbb{S}^{d-1} \mid i \in Q_2\}$.

From Lemma 4.2, we know that no $x \in \mathbb{R}$ is a simultaneous root of $He_{2\ell+1}(x), He_{2\ell+2}(x)$. Hence $Q_1 \cup Q_2 = \{1, 2, \ldots, m\}$. Thus we recover all $\{w_i \in \mathbb{S}^{d-1} : i \in [m]\}$ in Step 4. Finally using Claim 4.5, we recover for each $i \in [m]$, the $a_i, b_i \in \mathbb{R}$ corresponding to w_i.

□

Proof of Identifiability (Theorem 3.3): Theorem 3.3 follows by verifying that the conditions of Theorem 4.1 hold for $\ell = m$. Conditions (i) and (iii) follow from the conditions of Theorem 3.3.

We now verify condition (ii). For a matrix U with columns $\{u_i : i \in [m]\}$, the $\mathsf{krank}(U)$ (denoting the Kruskal-rank) is at least k iff every k of the m columns of U are linearly independent. Note that $\mathsf{krank}(U) \leq \mathsf{rank}(U) \leq m$. The krank increases under the Khatri-Rao product.

Fact 4.6 (Lemma A.4 of [BCV14]). *For two matrices U, V with m columns,* $\mathsf{krank}(U \odot V) = \min(\mathsf{krank}(U) + \mathsf{krank}(V) - 1, m)$.

Let $U = W^\top$ (with ith column w_i). Note that since no two columns are parallel, $\mathsf{krank}(U) \geq 2$. By applying the above fact on matrix $M = U^{\odot m}$ with ith column $w_i^{\otimes m}$, we get that $\mathsf{krank}(M) = m$, as required. Hence, Theorem 4.1 can be applied to recover all the unknown parameters. □

5 Robustness Analysis

In the previous section we showed that given oracle access to $\{\hat{f}_k\}$, we can recover the exact network parameters a, b, W. In reality, we can only access polynomially many samples in polynomial time, and we will have sampling errors when estimating $\{\hat{f}_k\}$. Therefore, given data generated from the target network $(x_1, y_1), ..., (x_n, y_n)$, we will approximate $\hat{f}_k$ through the empirical estimator

$$T_k = \frac{1}{n} \sum_{i=1}^{n} y_i He_k(x_i) \tag{8}$$

Observe that T_k is an unbiased estimator for $\hat{f}_k$. We first show using standard concentration bounds that for any $\eta > 0$, with $N \geq \mathrm{poly}(d^k, m, B, 1/\eta)$ samples, the empirical estimates with high probability satisfies $\forall \ell \in [k]$, $\|\xi_\ell\|_F := \|T_\ell - \hat{f}_\ell\|_F \leq \eta$ (see Appendix C.1). Hence for any constant k, with polynomial samples, we can obtain with high probability, estimates for the tensors $\{T_0, T_1, \ldots, T_k\}$ that are accurate up to any desired inverse-polynomial error.

We use Algorithm 3 described below and prove that it achieves the guarantees in Theorem 3.1. In fact, the analysis just assumes that $\|T_k - \hat{f}_k\|_F$ are upper bounded up to an amount that is inverse polynomial in the different parameters (this could also include other sources of error) to approximate the 2-layer ReLU network that approximates f up to desired inverse polynomial error ε.
The error parameters $\eta_0, \eta_1, \eta_2, \eta_3$ can be set with appropriate polynomial dependencies on $\varepsilon, d^\ell, m, B, s_m(W^{\odot \ell})$ to obtain the recovery guarantees in Theorem 3.1. See Section C.4 for details.

Overview of Analysis. The error in the tensors T_k introduces additional challenges that we described in Section 1. The analysis is technical and long, but we now briefly describe the main components.

(i) Recall from Section 1, that when there are errors, it may not even be possible to recover the parameters of some ReLU units! In particular when the bias b_i is large in magnitude, the ReLU unit will be indistinguishable from a simple linear function. It will contribute negligibly to any of the higher order Hermite coefficients, and hence will be impossible to recover them individually (especially if there are multiple such units). For a desired recovery error $\varepsilon > 0$, the m hidden ReLU units are split into groups (for analysis)

$$G = \{i \in [m] \mid |b_i| < O\big(\sqrt{\log(1/(\varepsilon \cdot mdB))}\big)\}, \text{ and } P = \{1, 2, \ldots, m\} \setminus G.$$

Algorithm 3: for order ℓ: recover a, b, W given estimates $\{T_0, \ldots, T_{2\ell+2}\}$.

Input: Estimates $T_0, \ldots, T_{2\ell+2}$ for $\hat{f}_0, \hat{f}_1, \ldots, \hat{f}_{2\ell+2}$;
Parameters: $\eta_0, \eta_1, \eta_2, \eta_3 > 0$.;

1. Let order-3 tensors $T' = \text{flatten}(T_{2\ell+1}, \ell, \ell, 1) \in \mathbb{R}^{d^\ell \times d^\ell \times d}$ and let $T'' = \text{flatten}(T_{2\ell+2}, \ell, \ell, 2) \in \mathbb{R}^{d^\ell \times d^\ell \times d^2}$.
2. Set $k' = \max_{r \le m} s_r(\text{flatten}(T_{2\ell+1}, \ell, \ell+1, 0)) > \eta_1$. Run Jennrich's algorithm on T' to recover rank-1 terms $\{\alpha'_i u_i^{\otimes \ell} \otimes u_i^{\otimes \ell} \otimes u_i \mid i \in [k']\}$, where $\forall i \in [k'],\ u_i \in \mathbb{S}^{d-1}$ and $\alpha'_i \in \mathbb{R}$.
3. Set $k'' = \max_{r \le m} s_r(\text{flatten}(T_{2\ell+2}, \ell, \ell+2, 0)) > \eta_1$. Run Jennrich's algorithm on T'' to recover rank-1 terms $\{\alpha''_i v_i^{\otimes \ell} \otimes v_i^{\otimes \ell} \otimes v_i^{\otimes 2} \mid i \in [k'']\}$, where $\forall i \in [k''],\ v_i \in \mathbb{S}^{d-1}$ and $\alpha''_i \in \mathbb{R}$.
4. Remove all the rank-1 terms in steps 2 and 3 with Frobenius norm $< \eta_2$ i.e., α''_i or $\alpha'_i < \eta_2$. Also remove all duplicates (within η_3 in ℓ_2 distance) from $\{u_1, u_2, \ldots, u_{k'}\} \cup \{v_1, v_2, \ldots, v_{k''}\}$ to get $\widetilde{w}_1, \widetilde{w}_2, \ldots, \widetilde{w}_{m'}$.
5. Run the subroutine $\textsc{RecoverScalars}(\ell, \{\widetilde{w}_i : i \in [m']\}, T_\ell, T_{\ell+1}, T_{\ell+2}, T_{\ell+3})$ (i.e., Alg. 2) to get $\{\widetilde{a}_i, \widetilde{b}_i : i \in [m']\}$.
6. Set $\widetilde{w}_0 := T_1 - \sum_{i=1}^{m'} \widetilde{a}_i \Phi(\widetilde{b}_i) \widetilde{w}_i$, and $\widetilde{b}_0 := T_0 - \sum_{i=1}^{m'} \widetilde{a}_i (\widetilde{b}_i \Phi(\widetilde{b}_i) + \exp(-\widetilde{b}_i^2/2)/\sqrt{2\pi})$.

Result: Output $\{\widetilde{w}_i, \widetilde{a}_i, \widetilde{b}_i : 1 \le i \le m'\}$ and $\{\widetilde{w}_0, \widetilde{b}_0\}$

We aim to recover all of the parameters of the units corresponding to G. For the terms in P, we will learn a linear function that approximates the total contribution from all the terms in P.

(ii) The tensor decomposition steps (steps 2-3) are simpler in the no-noise setting: the parameter w_i of the ith ReLU unit can be recovered as long its bias b_i is not a root of $He_{2\ell-1}$. When there is noise, there could be terms $i \in [m]$ for which b_i are not roots of $He_{k-2}(x)$, and yet their signal can get swamped by the sampling error in the tensor. We can only hope to recover those $k \le m$ components whose corresponding coefficient is above some chosen threshold η_1 (the other terms are considered as part of the error tensor). However a technical issue that arises is that the robust recovery guarantees for tensor decomposition algorithms lose polynomial factors in different parameters including the least singular value (s_k) of the factor matrices. Hence, for each of step 2 and 3, we argue that recovery is possible only if the coefficient of the corresponding term is significantly large, and this may give reasonable estimates for only a subset of these m terms with coefficients $> \varepsilon$.

(iii) When decomposing two consecutive tensors $T_{2\ell+1}$ and $T_{2\ell+2}$, we use Lemma 4.2 to argue that each $i \in G$ will have a large coefficient in at least one of these two tensors. Hence we can stitch together accurate estimates $\{\widetilde{w}_i : i \in G\}$. This will in turn be used to recover a_i, b_i for $i \in G$, with properties of Hermite polynomials used to ensure that the errors do not propagate badly.

(iv) We argue that the other ReLU units in $P = [m] \setminus G$ can be approximated altogether using a linear function. This is obtained by subtracting from estimates T_0, T_1 with the corresponding terms from G.

For details of the analysis, please see Section C for Theorem 3.1, and Section D for Corollary 3.2.

6 Conclusion

In this paper, we designed polynomial time algorithms for learning depth-2 neural networks with general ReLU activations (with non-zero bias terms), and gave provable guarantees under mild non-degeneracy conditions. The results of this work are theoretical in nature, in trying to understand whether efficient algorithms exist for learning ReLU networks; hence we believe they do not have any adverse societal impact. We addressed multiple challenges for learning such ReLU network with non-zero bias terms throughout our analyses, that may be more broadly useful in handling bias terms in the ReLU activations. We also proved identifiability under minimal assumptions and adopted the framework of smoothed analysis to establish beyond-worst-case guarantees. The major open direction is to provide similar guarantees for networks of higher depth.

References

[ADH+19] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

[APVZ14] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning polynomials with neural networks. In *International conference on machine learning*, pages 1908–1916. PMLR, 2014.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[BCMV14] A. Bhaskara, M. Charikar, A. Moitra, and A. Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 594–603, New York, NY, USA, 2014. Association for Computing Machinery.

[BCPV19] Aditya Bhaskara, Aidao Chen, Aidan Perreault, and Aravindan Vijayaraghavan. Smoothed analysis in unsupervised learning via decoupling. In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2019.

[BCV14] Aditya Bhaskara, Moses Charikar, and Aravindan Vijayaraghavan. Uniqueness of tensor decompositions with applications to polynomial identifiability. In Maria Florina Balcan, Vitaly Feldman, and Csaba Szepesvári, editors, *Proceedings of The 27th Conference on Learning Theory*, volume 35 of *Proceedings of Machine Learning Research*, pages 742–778, Barcelona, Spain, 2014. PMLR.

[Bha97] Rajendra Bhatia. *Matrix Analysis*, volume 169. Springer, 1997.

[BJW19] Ainesh Bakshi, Rajesh Jayaram, and David P Woodruff. Learning two layer rectified neural networks in polynomial time. In *Conference on Learning Theory*, pages 195–268. PMLR, 2019.

[CB18] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *arXiv preprint arXiv:1805.09545*, 2018.

[CKM20] Sitan Chen, Adam R Klivans, and Raghu Meka. Learning deep relu networks is fixed-parameter tractable. *arXiv preprint arXiv:2009.13512*, 2020.

[Dan17] Amit Daniely. Sgd learns the conjugate kernel class of the network. *arXiv preprint arXiv:1702.08503*, 2017.

[DFS16] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. *arXiv preprint arXiv:1602.05897*, 2016.

[DKKZ20] Ilias Diakonikolas, Daniel M Kane, Vasilis Kontonis, and Nikos Zarifis. Algorithms and sq lower bounds for pac learning one-hidden-layer relu networks. In *Conference on Learning Theory*, pages 1514–1539. PMLR, 2020.

[DZPS18] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

[GGJ+20] Surbhi Goel, Aravind Gollakota, Zhihan Jin, Sushrut Karmalkar, and Adam Klivans. Superpolynomial lower bounds for learning one-layer neural networks using gradient descent. In *International Conference on Machine Learning*, pages 3587–3596. PMLR, 2020.

[GK19] Surbhi Goel and Adam R Klivans. Learning neural networks with two nonlinear layers in polynomial time. In *Conference on Learning Theory*, pages 1470–1499. PMLR, 2019.

[GKLW18] Rong Ge, Rohith Kuditipudi, Zhize Li, and Xiang Wang. Learning two-layer neural networks with symmetric inputs. *arXiv preprint arXiv:1810.06793*, 2018.

[GLM17] Rong Ge, Jason D Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. *arXiv preprint arXiv:1711.00501*, 2017.

[GVX14] Navin Goyal, Santosh Vempala, and Ying Xiao. Fourier pca and robust tensor decomposition. STOC '14, page 584–593, New York, NY, USA, 2014. Association for Computing Machinery.

[Har70] R. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-model factor analysis. 1970.

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[JGKA19] Majid Janzamin, Rong Ge, Jean Kossaifi, and Animashree Anandkumar. Spectral learning on matrices and tensors. *Foundations and Trends in Machine Learning*, 12, 11 2019.

[JSA15] 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.

[LXS+19] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.

[Moi18] Ankur Moitra. *Algorithmic Aspects of Machine Learning*. Cambridge University Press, 2018.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, USA, 2014.

[Rou20] Tim Roughgarden. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

[ST04] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[Tur50] Paul Turán. On the zeros of the polynomials of legendre. *Časopis pro pěstování matematiky a fysiky*, 75(3):113–122, 1950.

[Vij20] Aravindan Vijayaraghavan. *Beyond the Worst-Case Analysis of Algorithms*, chapter Efficient Tensor Decomposition. In [Rou20], 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Some of the proofs are in the Supplementary material or Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]
(b) Did you mention the license of the assets? [N/A]
(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]