On the Connection Between Learning Two-Layer Neural Networks and Tensor Decomposition

Marco Mondelli
Stanford University

Andrea Montanari
Stanford University

Abstract

We establish connections between the problem of learning a two-layer neural network and tensor decomposition. We consider a model with feature vectors $\boldsymbol{x} \in \mathbb{R}^d$, r hidden units with weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ and output $y \in \mathbb{R}$, i.e., $y = \sum_{i=1}^r \sigma(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x})$, with activation functions given by low-degree polynomials. In particular, if $\sigma(x) = a_0 + a_1 x + a_3 x^3$, we prove that no polynomial-time algorithm can outperform the trivial predictor that assigns to each example the response variable $\mathbb{E}(y)$, when $d^{3/2} \ll r \ll d^2$. Our conclusion holds for a 'natural data distribution', namely standard Gaussian feature vectors $\boldsymbol{x}$, and output distributed according to a two-layer neural network with random isotropic weights, and under a certain complexity-theoretic assumption on tensor decomposition. Roughly speaking, we assume that no polynomial-time algorithm can substantially outperform current methods for tensor decomposition based on the sum-of-squares hierarchy.

We also prove generalizations of this statement for higher degree polynomial activations, and non-random weight vectors. Remarkably, several existing algorithms for learning two-layer networks with rigorous guarantees are based on tensor decomposition. Our results support the idea that this is indeed the core computational difficulty in learning such networks, under the stated generative model for the data. As a side result, we show that under this model learning the network requires accurate learning of its weights, a property that does not hold in a more general setting.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

1 Introduction and Main Results

Let $\{(\boldsymbol{x}_i, y_i)\}_{1\le i\le n}$ be n data points where, for each i, $\boldsymbol{x}_i \in \mathbb{R}^d$ is a feature vector and $y_i \in \mathbb{R}$ is a response variable or label. The simplest neural network attempts to fit these data using the model

$$\hat{y}(\boldsymbol{x}; \hat{\boldsymbol{w}}) = \sum_{i=1}^r \sigma(\langle \boldsymbol{x}, \hat{\boldsymbol{w}}_i\rangle)\,. \tag{1}$$

Here $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function, and $\hat{\boldsymbol{w}} = (\hat{\boldsymbol{w}}_i)_{i\le r}$, where $\hat{\boldsymbol{w}}_1, \dots, \hat{\boldsymbol{w}}_r \in \mathbb{R}^d$ are model parameters (weight vectors). In the following, we will often omit the argument $\hat{\boldsymbol{w}}$ from $\hat{y}$. Let us emphasize that this is a deliberately oversimplified neural network model: (*i*) It only includes one hidden layer of r units (neurons); (*ii*) The output unit is linear (it takes a linear combination of the hidden units); (*iii*) The hidden units have no offset or output weight. Since our main results are negative (computational hardness), we are not too concerned with such simplifications. For instance, it is unlikely that adding a non-linear output unit can reduce the problem hardness.

Throughout this paper, we will assume the data to be i.i.d. with common distribution $\mathcal{D}$, namely $(\boldsymbol{x}_i, y_i) \sim \mathcal{D}$. A rapidly growing literature develops algorithms and rigorous guarantees to learn such a model, see e.g. [Janzamin et al., 2015, Soudry and Carmon, 2016, Soltanolkotabi et al., 2019, Safran and Shamir, 2016, Freeman and Bruna, 2017, Ge et al., 2018, Zhong et al., 2017] and the brief overview in Section 1.1. These papers analyze the landscape of empirical risk minimization for the model (1), or its variants. Under suitable assumptions on the data distribution $\mathcal{D}$ (as well as the parameters d, r, n) they develop algorithms that are guaranteed to recover the weights $\hat{\boldsymbol{w}}_1, \dots, \hat{\boldsymbol{w}}_r$ with small training error.

In this paper we consider the complementary question, and use a reduction from tensor decomposition to provide evidence that –in certain regimes, and for certain data distributions $\mathcal{D}$– the model (1) cannot be learnt in polynomial time. Let us emphasize two important aspects of our results. First, our impossibility results

are entirely computational, and do not depend on the data distribution $\mathcal{D}$. Indeed, they hold even if we have access to an infinite sample (more accurately, they hold under a stronger model that allows us to compute expectations with respect to $\mathcal{D}$). There is nothing surprising in this. Think of minimizing an empirical risk function, which is an average over n samples. If its expectation is a very complex function, then the problem remains hard irrespective of the number of samples n. Second, earlier work has proven computational hardness for simpler problems than the neural network (1). For instance, [Daniely, 2016] proves hardness for learning a single linear classifier. However these proofs are based on the construction of special distributions $\mathcal{D}$ that are are unknown to the learner. Here instead we consider a 'natural' class of distributions $\mathcal{D}$ that is in fact normally assumed in works estabilishing positive guarantees. This point of view is similar to the one recently developed in [Shamir, 2018] although our methods and results are quite different.

As mentioned above, our results are conditional on a complexity-theoretic assumption for tensor decomposition, i.e. the problem of recovering the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ given access to the k-th order tensor $\boldsymbol{T}^{(k)} = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes k}$. We state this assumption explicitly below, for the case of tensors of order $k = 3$.

Conjecture 1 (ϵ-Hardness of 3-Tensor Decomposition)**.** *The following holds for some $\epsilon_0 > 0$, and all $\delta > 0$. Define a distribution $\mathcal{W}_{d,r}$ over the weights $\boldsymbol{w} = (\boldsymbol{w}_i)_{1\le i\le r} \in (\mathbb{R}^d)^r$, by letting*

$$\boldsymbol{w}_i = \frac{\boldsymbol{g}_i - \frac{1}{r}\sum_{j=1}^r \boldsymbol{g}_j}{\left\|\boldsymbol{g}_i - \frac{1}{r}\sum_{j=1}^r \boldsymbol{g}_j\right\|}, \qquad \forall\, i \in [r], \tag{2}$$

where $\{\boldsymbol{g}_i\}_{1\le i\le r} \sim_{\text{i.i.d.}} \mathsf{N}(\boldsymbol{0}_d, \boldsymbol{I}_d/d)$. Set $\boldsymbol{T}(\boldsymbol{w}_1, \dots, \boldsymbol{w}_r) = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes 3}$. Assume $r = r(d) \ge d^{(3/2)+\delta}$ and $\epsilon < \epsilon_0$. Then, there is no algorithm $\mathcal{A}$ that, given as input $\boldsymbol{T}(\boldsymbol{w})$, with $\boldsymbol{w} = (\boldsymbol{w}_i)_{1\le i\le r} \sim \mathcal{W}_{d,r}$ fulfills the following two properties:

(P1) *$\mathcal{A}$ outputs $\{\hat{\boldsymbol{w}}_i\}_{1\le i\le r}$ of unit norm s.t., with probability at least $1/2$, for some $i, j \in [r]$, $|\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_j\rangle| \ge \epsilon$.*

(P2) *$\mathcal{A}$ has complexity bounded by a polynomial in d.*

Tensor decomposition has been studied by a number of authors, and the best known algorithms are based on (or match the guarantees of) the sum-of-squares (SoS) hierarchy [Hopkins et al., 2016, Ma et al., 2016, Schramm and Weitz, 2015]. The above assumption amounts to conjecturing that no algorithm can beat SoS for this problem[1]. We limit ourselves to noticing that SoS appears to capture computational boundaries in a number of similar statistical problems [Barak and Steurer, 2014, Hopkins et al., 2015, Barak et al., 2015, Barak et al., 2016, Hopkins et al., 2017].

Theorem 1. *Let $\sigma(x) = a_0 + a_1 x + a_3 x^3$ for some $a_0, a_1, a_3 \in \mathbb{R}$ and denote by $\mathcal{N}(d, r)$ the set of functions $\hat{y}(\,\cdot\,; \hat{\boldsymbol{w}}) : \mathbb{R}^d \to \mathbb{R}$ of the form (1) where $\|\hat{\boldsymbol{w}}_1\|_2 = \cdots = \|\hat{\boldsymbol{w}}_r\|_2 = 1$. Assume $r = r(d)$ to be such that $d^{(3/2)+\delta} \le r \le d^{2-\delta}$ for some $\delta > 0$. Then, under Conjecture 1, there exists $\eta(r, d) \to 0$ as $d \to \infty$ such that the following holds. Let $\boldsymbol{w} = (\boldsymbol{w}_j)_{j\le r} \sim \mathcal{W}_{d,r}$ be random weights, see Eq. (2). Consider data $\{(\boldsymbol{x}_i, y_i)\}_{i\le n}$ with common distribution $\mathcal{D}$ defined by $\boldsymbol{x}_i \sim \mathsf{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$ and $y_i = y(\boldsymbol{x}_i) = \hat{y}(\boldsymbol{x}_i; \boldsymbol{w})$, with $\hat{y}(\boldsymbol{x}; \boldsymbol{w})$ given by (1). In particular,*

$$\min_{\hat{y}(\,\cdot\,)\in\mathcal{N}(d,r)} \mathbb{E}_{\mathcal{D}}\left\{|y(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})|^2\right\} = 0\,. \tag{3}$$

However, for any polynomial-time algorithm $\mathcal{P}$ that takes as input $\{(\boldsymbol{x}_i, y_i)\}_{i\le n}$ and returns a function $\hat{y}_{\mathcal{P}} \in \mathcal{N}(d, r)$, we have that

$$\mathbb{E}_{\mathcal{D}}\left\{|y(\boldsymbol{x}) - \hat{y}_{\mathcal{P}}(\boldsymbol{x})|^2\right\} \ge \mathrm{Var}\left\{y(\boldsymbol{x})\right\}\,(1 - \eta(r, d))\,, \tag{4}$$

with high probability with respect to $\boldsymbol{w} \sim \mathcal{W}_{d,r}$.

Remark 1: The right-hand side of (4) is the risk of a trivial model that always predicts y with its expectation. Hence, Theorem 1 implies that, under the data distribution $\mathcal{D}$, no polynomial algorithm can predict the response better than a trivial predictor that assigns to each example the same response $\mathbb{E}(y)$. Notice that this lower bound is independent of n, and in fact we prove it under a more powerful model, whereby the algorithm $\mathcal{P}$ is given access to an oracle that computes expectations with respect to $\mathcal{D}$. On the other hand, under unbounded computation, it is possible to find a neural network of the form (1), with zero test error.

Remark 2: A large part of the theoretical literature adopts the same model of the above theorem, namely random Gaussian features $\boldsymbol{x} \sim \mathsf{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$, and data generated according to a two-layer network with random weights, see e.g. [Janzamin et al., 2015, Ge et al., 2018, Zhong et al., 2017, Soltanolkotabi et al., 2019]. Our theorem implies that, within the assumptions of these papers, $r \ll d^{3/2}$ is a computational hardness barrier (under the stated conjecture on tensor decomposition). Note that several of these papers use tensor decomposition procedures as a key subroutine (typically to initialize the weights before gradient descent). Theorem 1 implies that the appearance of tensor decomposition in these algorithms is a consequence of a fundamental connection between the two problems.

[1] We assume that it is impossible to estimate even a single component of $\boldsymbol{T}$. This is motivated by the remark that in all existing algorithmic approaches for tensor decomposition, the problems of learning a single component and of learning all components are either both solvable or both unsolvable, e.g., see [Ma et al., 2016, Schramm and Steurer, 2017]. It is also easy to see that they are equivalent if we demand *exact* reconstruction of the weights.

In the rest of this introduction we provide a brief overview of related work. We then present our technical contributions. In Section 3, we show that, if we cannot estimate the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ accurately, then the error $\mathbb{E}\{|y(\boldsymbol{x})-\hat{y}(\boldsymbol{x})|^2\}$, typically called generalization error[2], of the predictor $\hat{y}(\boldsymbol{x})$ is close to that of a trivial predictor. We prove this result in two separate settings: for deterministic and for random weights $(\boldsymbol{w}_i)_{i\le r}$. In Section 4, we present reductions from the problem of tensor decomposition to the problem of estimating the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ in the two-layer neural network model. By combining these two results, in Section 5 we present reductions from the problem of tensor decomposition to the problem of learning a two-layer neural network with small generalization error. These results generalize Theorem 1 in two directions: we consider non-random weights, and a broader set of polynomial activation functions $\sigma(\,\cdot\,)$. Finally, in Section 6, we present numerical experiments supporting our theoretical findings.

In summary, we consider a popular model for theoretical research (random two-layer neural network with Gaussian feature vectors) and show that: (*i*) learning in this model requires accurate weight estimation; and (*ii*) the latter requires solving a tensor decomposition problem, which is computationally expensive. A promising direction of research would be to understand whether these conclusions can be avoided by considering different generative models.

1.1 Related Work

Several recent papers provide recovery guarantees for neural network models, and what follows is a necessarily incomplete overview. In [Arora et al., 2014], the weights are assumed to be sparse and random, and the proposed algorithm learns almost all the models in this class with polynomial sample and computational complexity. In [Brutzkus and Globerson, 2017], the authors consider a two-layer neural network with convolutional structure, no overlap[3], and ReLU activation function. It is shown that learning is NP-complete in the worst case, but gradient descent converges to the global optimum in polynomial time when the input distribution is Gaussian. A similar positive result, i.e., convergence to the global optimum of gradient descent with polynomial complexity and Gaussian input, is proved in [Tian, 2017]. In this work, the author considers a two-layer neural network model of the form (1), where σ is a ReLU activation function and the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ are orthogonal (which implies that $r\le d$). However, [Tian, 2017] requires a good initialization and does not discuss initialization methods. In [Panigrahy et al., 2018], the authors design an activation function that guarantees provable learning, but the proposed algorithm runs in $d^{O(d)}$. In [Sedghi and Anandkumar, 2015], the subspace spanned by the weight matrix is provably recovered with a tensor decomposition algorithm, and the weights can also be recovered under an additional sparsity assumption. The works [Brutzkus and Globerson, 2017, Tian, 2017, Sedghi and Anandkumar, 2015] consider only the population risk and do not give bounds on the sample complexity. The paper [Janzamin et al., 2015] presents a tensor based algorithm that learns a two-layer neural network with sample complexity of order $d^3\cdot\mathrm{poly}(r)/\varepsilon^2$, where ε is the precision. In [Zhong et al., 2017], a tensor initialization algorithm is combined with gradient descent to obtain a procedure with sample complexity of order $d\cdot\mathrm{poly}(r)\cdot\log(1/\varepsilon)$ and computational complexity $n\cdot d\cdot\mathrm{poly}(r)\cdot\log(1/\varepsilon)$, where n is the number of samples and it is assumed that $r\le d$. The connection between tensors and neural networks is also studied in [Ge et al., 2018].

As mentioned above, several hardness results are available for training neural networks or even simple linear classifiers [Blum and Rivest, 1989, Bartlett and Ben-David, 1999, Kuhlmann, 2000, Šíma, 2002, Daniely, 2016]. However, these results are either worst-case or rely on special constructions of the distribution $\mathcal{D}$. In contrast here, we consider a class of distributions that has been frequently studied in the algorithms literature, in order to estabilish rigorous guarantees. Similar in spirit to our results is the recent work [Shamir, 2018] which considers data generated according to the model (1) with smooth distributions of the feature vectors $\boldsymbol{x}$, and periodic activation functions (while we consider low-degree polynomials). Apart from technical differences in the model, our results are different and complementary to the ones of [Shamir, 2018]. While [Shamir, 2018] analyzes a specific class of 'approximate gradient' algoritms, we prove a general average-case hardness result, conditional on a complexity-theoretic assumption. Notice that proving average-case hardness under weak assumptions (e.g., P not equal to NP) is extremely difficult, and has been achieved only in a handful of cases. Recently, significant progress in understanding the hardness of statistical problems has been achieved by making stronger assumptions, as in this work. For instance, several results have been obtained conditional on hardness of the planted clique problem [Berthet and Rigollet, 2013, Brennan et al., 2018].

[2] The term 'generalization error' is often used interchangeably with 'risk' and it refers to the expected loss of a prediction rule also in the realizable case, see [Bousquet and Elisseeff, 2002, Zhong et al., 2017] and [Shalev-Shwartz and Ben-David, 2014, pp. 34-35].

[3] The filter of the convolutional neural network is applied to non-overlapping parts of the input vector.

2 Preliminaries

2.1 Notation and System Model

Let $[n]$ be a shorthand for $\{1, \dots, n\}$. Let $\mathbf{0}_n$ and $\mathbf{1}_n$ denote the vector consisting of n 0s and n 1s, respectively, and let $\boldsymbol{I}_n$ denote the $n \times n$ identity matrix. Given a vector $\boldsymbol{x} \in \mathbb{R}^n$, we let $x(i)$ be its i-th element, where $i \in [n]$, and $\|\boldsymbol{x}\|$ be its ℓ_2 norm. Given a matrix $\boldsymbol{A}$, we let $\boldsymbol{A}^\mathsf{T}$ be its transpose, $\mathrm{Tr}(\boldsymbol{A})$ be its trace, $\|\boldsymbol{A}\|_F$ be its Frobenius norm, and $\|\boldsymbol{A}\|_{\mathrm{op}}$ be its operator norm. We use $\boldsymbol{A} \otimes \boldsymbol{B}$ to denote the Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\boldsymbol{A}^{\otimes k}$ as a shorthand for $\boldsymbol{A} \otimes \cdots \otimes \boldsymbol{A}$, where $\boldsymbol{A}$ appears k times. We also set $\boldsymbol{A}^{\otimes 0} = 1$. Given two k-th order tensors $\boldsymbol{x}, \boldsymbol{y} \in (\mathbb{R}^d)^{\otimes k}$, we let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i_1,\dots,i_k=1}^{d} x(i_1, \dots, i_k) \cdot y(i_1, \dots, i_k)$ be their scalar product. Given a k-th order tensor $\boldsymbol{x} \in (\mathbb{R}^d)^{\otimes k}$, we let $\|\boldsymbol{x}\|_F = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ be its Frobenius norm. Given an integer k, we denote by $\mathrm{par}(k)$ its parity, i.e., we set $\mathrm{par}(k)$ to 0 if k is even and to 1 if k is odd. Given a polynomial f, we denote by $\deg(f)$ its degree. If f is either even or odd, we denote by $\mathrm{par}(f)$ its parity, i.e., we set $\mathrm{par}(f)$ to 0 if f is even and to 1 if f is odd. Given a function σ in the weighted L^2 space[4] $L^2(\mathbb{R}, e^{-x^2/2})$, we denote by $\hat{\sigma}_k$ its k-th Hermite coefficient. It is helpful to write explicitly the formulas to compute $\hat{\sigma}_1$ and $\hat{\sigma}_2$:

$$\begin{aligned} \hat{\sigma}_1 &= \mathbb{E}_{G \sim \mathsf{N}(0,1)} \left\{ G \cdot \sigma(G) \right\}, \\ \hat{\sigma}_2 &= \frac{1}{\sqrt{2}} \mathbb{E}_{G \sim \mathsf{N}(0,1)} \left\{ (G^2 - 1) \sigma(G) \right\}. \end{aligned} \tag{5}$$

Throughout the paper, we consider a two-layer neural network with input dimension d and r hidden nodes with weights $\boldsymbol{w} = (\boldsymbol{w}_i)_{1 \le i \le r} \in (\mathbb{R}^d)^r$. We denote the input by $\boldsymbol{x} \in \mathbb{R}^d$ and the output by $y(\boldsymbol{x}; \boldsymbol{w}) \in \mathbb{R}$, which is defined by

$$y(\boldsymbol{x}; \boldsymbol{w}) = \sum_{i=1}^{r} \sigma(\langle \boldsymbol{x}, \boldsymbol{w}_i \rangle). \tag{6}$$

We will often omit the argument $\boldsymbol{w}$ from y. Given n samples from the neural network, we obtain the estimates $\{\hat{\boldsymbol{w}}_i\}_{1 \le i \le r}$ on the weights $\{\boldsymbol{w}_i\}_{1 \le i \le r}$, which allows us to construct $\hat{y}(\boldsymbol{x})$ given by (1). Two error metrics can be considered. A stronger requirement is to learn accurately (up to a permutation) the weights. More formally, we require that the *estimation error* $\min_\pi \sum_{i=1}^{r} \left\| \boldsymbol{w}_i - \hat{\boldsymbol{w}}_{\pi(i)} \right\|^2$ is small, where the minimization is with respect to all permutations $\pi : [r] \to [r]$. If we assume that the vectors $\{\boldsymbol{w}_i\}_{1 \le i \le n}$ and $\{\hat{\boldsymbol{w}}_i\}_{1 \le i \le n}$ have unit norm, then this quantity is small if and only if $\max_\pi \sum_{i=1}^{r} \left\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_{\pi(i)} \right\rangle$ is large. A weaker requirement is to predict accurately the output of the network. More formally, we require that the *generalization error* $\mathbb{E}\left\{ |y(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})|^2 \right\}$ is small, where the expectation is with respect to the distribution of $\boldsymbol{x}$. Our results of Section 3 prove that these two requirements are equivalent when $\boldsymbol{x}$ is Gaussian: if the stronger requirement does not hold, i.e., the estimates $\{\hat{\boldsymbol{w}}_i\}_{1 \le i \le r}$ have small correlation with the weights $\{\boldsymbol{w}_i\}_{1 \le i \le r}$, then also the weaker requirement does not hold, i.e., the generalization error is large.

[4] $L^2(\mathbb{R}, e^{-x^2/2}) = \left\{ \sigma : \int_{\mathbb{R}} |\sigma(x)|^2 e^{-x^2/2} \, \mathrm{d}x < \infty \right\}$.

2.2 Tensor Decomposition

Tensors are arrays of numbers indicized by multiple integers. As for the problem of learning a neural network, many problems involving tensors (e.g., the computation of the rank or the spectral norm) are NP-hard in the worst case [Håstad, 1990, Hillar and Lim, 2013]. However, recent work has focused on provably efficient algorithms by making assumptions on the input and allowing for approximations [Anandkumar et al., 2015, Anandkumar et al., 2017, Ge and Ma, 2015, Hopkins et al., 2015, Hopkins et al., 2016, Barak et al., 2015, Ma et al., 2016, Schramm and Steurer, 2017]. The typical setting for the problem of tensor decomposition is as follows. Let $\boldsymbol{w}_1, \dots, \boldsymbol{w}_r \in \mathbb{R}^d$ be vectors of unit norm and, for $k \ge 3$, define the k-th order tensor $\boldsymbol{T}^{(k)} = \sum_{i=1}^{r} \boldsymbol{w}_i^{\otimes k}$. Given a subset of tensors $\{\boldsymbol{T}^{(k)}\}_{3 \le k \le \ell}$, the goal is to recover $\{\boldsymbol{w}_i\}_{i \le r}$. A classical algorithm based on matrix diagonalization [Harshman, 1970, De Lathauwer et al., 1996] solves tensor decomposition when $\{\boldsymbol{w}_i\}_{i \le r}$ are linearly independent and $\ell \ge 3$. The requirement that $\{\boldsymbol{w}_i\}_{i \le r}$ are linearly independent implies that $r \le d$. Recent works have focused on the overcomplete case, where $r > d$, and on a more general setup where the vectors $\{\boldsymbol{w}_i\}_{i \le r}$ are smoothed, i.e., randomly perturbed [Bhaskara et al., 2014, Ma et al., 2016]. The best algorithms are based on (or match the guarantees of) the SoS hierarchy and some of these results are reviewed below.

Random vectors. Assume that $\{\boldsymbol{w}_i\}_{i \le r}$ are chosen i.i.d. from the unit sphere in $\mathbb{R}^d$. Then, with high probability, tensor decomposition can be solved given $\boldsymbol{T}^{(3)}$ and r as large as $d^{3/2}$ (up to logarithmic factors), see Theorem 1.2 in [Ma et al., 2016].

Separated unit vectors. Assume that $\{\boldsymbol{w}_i\}_{i \le r}$ have at most δ-correlation, i.e., for any $i, j \in [r]$ with $i \ne j$, $|\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle| \le \delta$. Then, tensor decomposition can be solved given the tensors of order up to $\log r / \log(1/\delta)$ [Schramm and Weitz, 2015].

General unit vectors. Here, $\{\boldsymbol{w}_i\}_{i \le r}$ can be any vectors in $\mathbb{R}^d$. Then, tensor decomposition can be approximated given the tensors of order up to $\mathrm{poly}(1/\varepsilon)$, where ε denotes the Hausdorff distance between the original set of weights and the set of estimates, see Theorem 1.6 in [Ma et al., 2016].

3 Lower Bounds on Generalization Error

In our results, we consider a more general predictor $\hat{y}(\boldsymbol{x})$ given by

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^{R} \sigma(\langle \boldsymbol{x}, \hat{\boldsymbol{w}}_i \rangle), \tag{7}$$

i.e., we allow the number R of estimated weights to be different from the number r of unknown weights. Our first theorem holds when the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ are separated and isotropic, and our second theorem when the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ are random.

3.1 Separated Isotropic Weights

We make the following assumptions on $\{\boldsymbol{w}_i\}_{1\le i\le r}$.

(A1) *Unit norm*: $\|\boldsymbol{w}_i\| = 1, \qquad \forall\, i \in [r]$.

(A2) *At most δ-correlation*: $|\langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle| \le \delta, \qquad \forall\, i, j \in [r],$ with $i \neq j$.

(A3) *Mean $\eta_{\rm avg}$-close to zero*: $\|\sum_{i=1}^{r} \boldsymbol{w}_i\|^2 \le \eta_{\rm avg} \cdot r$.

(A4) *Covariance $\eta_{\rm var}$-close to scaled identity*: $\big\|\sum_{i=1}^{r} \boldsymbol{w}_i \boldsymbol{w}_i^{\mathsf{T}} - \frac{r}{d}\boldsymbol{I}_d\big\|_{\rm op} \le \eta_{\rm var} \cdot r/d$.

It is simple to produce weight vectors that satisfy these assumptions. If the weight matrix is the identity, then the assumptions hold with $\delta = 0$, $\eta_{\rm avg} = 1$, and $\eta_{\rm var} = 0$. If we center and rescale the weights by a factor $\sqrt{d/(d-1)}$, the assumptions hold with $\delta = \frac{d+1}{d(d-1)} \approx 1/d$, $\eta_{\rm avg} = 0$, and $\eta_{\rm var} = 2$. For $r = d+1$, we can take $\widetilde{\boldsymbol{W}}$ to be Haar distributed conditional on $\widetilde{\boldsymbol{W}}^{\mathsf{T}}\mathbf{1}_{d+1} = \mathbf{0}_d$, and let $\{\boldsymbol{w}_i\}_{1\le i\le r}$ be $\sqrt{(d+1)/d}$ times the rows of $\widetilde{\boldsymbol{W}}$ (these are just the rotations of the vertices of the standard simplex). Then, the assumptions hold with $\delta = 1/d$ and $\eta_{\rm avg} = \eta_{\rm var} = 0$. For $r > d+1$, we concatenate $r/(d+1)$ of these matrices. By doing so, we still have that $\eta_{\rm avg} = \eta_{\rm var} = 0$. We expect δ to be small (say of order $1/\sqrt{d}$).

The result below, whose proof is contained in Appendix A of the supplementary material, considers the case of a Gaussian input distribution and rules out a scenario in which the weights are not estimated well, but the generalization error is still small.

Theorem 2 (Lower Bound on Generalization Error for Separated Isotropic Weights)**.** *Consider a two-layer neural network with input dimension d, r hidden nodes, and activation function $\sigma \in L^2(\mathbb{R}, e^{-x^2/2})$. Assume that the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ satisfy the assumptions* (A1)-(A4) *for positive $\delta, \eta_{\rm avg}$ and $\eta_{\rm var}$ such that $1 - \delta\cdot(1+\eta_{\rm var})\cdot r/d \ge 0$. Let $y(\boldsymbol{x})$ and $\hat{y}(\boldsymbol{x})$ be defined in* (6) *and* (7)*. Assume that the estimated weights $\{\hat{\boldsymbol{w}}_i\}_{1\le i\le R}$ satisfy the assumption* (A1) *and have at most ϵ-correlation with the ground-truth weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$, i.e., for some $\epsilon > 0$, $|\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_j\rangle| \le \epsilon$, for all $i \in [r]$ and $j \in [R]$. Then, the following lower bound on the generalization error holds:*

$$\mathbb{E}\big\{|y(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})|^2\big\} \tag{8}$$
$$\ge \left(\min_{a,b\in\mathbb{R}} \mathbb{E}\left\{\Big|y(\boldsymbol{x}) - \big(a + b\,\|\boldsymbol{x}\|^2\big)\Big|^2\right\} - c_1\right)(1-c_2)\,,$$

where the expectation is with respect to $\boldsymbol{x} \sim \mathsf{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and the terms c_1 and c_2 are given by

$$c_1 = 2\hat{\sigma}_1^2\,\eta_{\rm avg}\,r + 2\hat{\sigma}_2^2\,\eta_{\rm var}^2\,\frac{r^2}{d},\ c_2 = \frac{2\epsilon\,(1+\eta_{\rm var})\,\dfrac{R}{d}}{1-\delta\,(1+\eta_{\rm var})\,\dfrac{r}{d}}, \tag{9}$$

with $\hat{\sigma}_1$ and $\hat{\sigma}_2$ defined in (5)*. If we also assume that σ is even, then* (8) *holds with c_1 and c_2 given by*

$$c_1 = 2\hat{\sigma}_2^2\,\eta_{\rm var}^2\,\frac{r^2}{d},\ \ c_2 = \frac{2\epsilon^2\,(1+\eta_{\rm var})\,\dfrac{R}{d}}{1-\delta^2\,(1+\eta_{\rm var})\,\dfrac{r}{d}}. \tag{10}$$

Some remarks are of order. First, note that the generalization error $\min_{a,b\in\mathbb{R}} \mathbb{E}\left\{\left|y(\boldsymbol{x}) - \left(a + b\,\|\boldsymbol{x}\|^2\right)\right|^2\right\}$ is that of a trivial predictor having access only to the norm of the input. Hence, if the weights are not estimated well, then the generalization error is close to that of a predictor that does not really use the input. Second, the assumption that the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ and $\{\hat{\boldsymbol{w}}_i\}_{1\le i\le R}$ have unit norm mainly serves to simplify the proof. On the contrary, the assumption that the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ are roughly isotropic is crucial. Indeed, if either (A3) or (A4) do not hold, then it might be possible to learn the mean vector or the covariance matrix of the weights, which could reduce the generalization error for activation functions that have a non-zero linear or quadratic component. Indeed, consider the following example: $\sigma(x) = x$, $\{\boldsymbol{w}_i\}_{i\le r}$ arbitrary, and $\hat{\boldsymbol{w}}_i = \overline{\boldsymbol{w}} \equiv \sum_{i=1}^{r} \boldsymbol{w}_i/r$ for all i. Clearly, the weights are not estimated correctly. However, the generalization error is 0 for any input $\boldsymbol{x} \in \mathbb{R}^d$ (and is superior to the one of the trivial predictor).

Let us evaluate the bound for some natural choices of the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$. Recall that, if σ is even (odd), then $\hat{\sigma}_k = 0$ for k odd (even). If the matrix of the weights is equal to the identity matrix and σ is even, then the generalization error of the neural network is close to that of a trivial predictor, namely, the neural network does not generalize well, as long as $\epsilon^2 \cdot R/d$ is small. Suppose now that we center and rescale the weights and that we pick σ odd. Then, the neural network does not generalize well as long as $\epsilon \cdot R/d$ is small. If the weights are the rescaled rows of $r/(d+1)$

matrices $\widetilde{\boldsymbol{W}}$, where $\widetilde{\boldsymbol{W}}$ is Haar distributed conditional on $\widetilde{\boldsymbol{W}}^\mathsf{T}\mathbf{1}_{d+1} = \mathbf{0}_d$, then, for any σ, the neural network does not generalize well as long as $\epsilon \cdot R/d$ and $\delta \cdot r/d$ are small. Furthermore, when σ is even, we only require that $\epsilon^2 \cdot R/d$ and $\delta^2 \cdot r/d$ are small.

3.2 Random Weights

We assume that the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ have the following form:

$$\boldsymbol{w}_i = \frac{\boldsymbol{g}_i - \frac{1}{r}\sum_{j=1}^r \boldsymbol{g}_j}{\left\|\boldsymbol{g}_i - \frac{1}{r}\sum_{j=1}^r \boldsymbol{g}_j\right\|}, \qquad \forall\, i \in [r], \tag{11}$$

where $\{\boldsymbol{g}_i\}_{1\le i\le r} \sim_{\text{i.i.d.}} \mathsf{N}(\mathbf{0}_d, \boldsymbol{I}_d/d)$. The result below, whose proof is contained in Appendix B of the supplementary material, is similar in spirit to Theorem 2 and it applies to a setting with random weights.

Theorem 3 (Lower Bound on Generalization Error for Random Weights)**.** *Consider a two-layer neural network with input dimension d, r hidden nodes, and activation function $\sigma \in L^2(\mathbb{R}, e^{-x^2/2})$ such that $\hat\sigma_2 = 0$, where $\hat\sigma_2$ is defined in* (5)*. Assume that the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$ have the form* (11)*. Let $y(\boldsymbol{x})$ and $\hat y(\boldsymbol{x})$ be defined in* (6) *and* (7)*. For some $\epsilon \in (0,1)$, define*

$$\hat{\mathcal{S}}_\epsilon = \{\{\hat{\boldsymbol{w}}_i\}_{i\le R} : \|\hat{\boldsymbol{w}}_i\| = 1\ \forall\, i, |\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_j\rangle| \le \epsilon\ \forall\, i, j\}. \tag{12}$$

As $r, d \to \infty$, assume that

$$\epsilon = o(1), \qquad r = o(d^2/(\log d)^2). \tag{13}$$

Then, for a sequence of vanishing constants $\eta(r,d) = o(1)$, with high probability with respect to $\boldsymbol{w} = (\boldsymbol{w}_i)_{i\le r}$,

$$\sup_{\{\hat{\boldsymbol{w}}_i\}_{1\le i\le R}\in\hat{\mathcal{S}}_\epsilon} \mathbb{E}\big\{|y(\boldsymbol{x}) - \hat y(\boldsymbol{x})|^2\big\} \tag{14}$$
$$\ge \left(\mathrm{Var}\{y(\boldsymbol{x})\} - r\cdot\eta(r,d)\right)\left(1 - \frac{R}{r}\cdot\eta(r,d)\right),$$

where the expectation and the variance are with respect to $\boldsymbol{x} \sim \mathsf{N}(\mathbf{0}_d, \boldsymbol{I}_d)$.

Some remarks are of order. First, note that $\langle \boldsymbol{x}, \boldsymbol{w}_i\rangle$ is of order 1, hence the term $\mathrm{Var}\{y(\boldsymbol{x})\}$ is of order r. Consequently, in the limit $r, d \to \infty$, the term $r\,\eta(r,d)$ is negligible compared to $\mathrm{Var}\{y(\boldsymbol{x})\}$. Second, the hypothesis that $\hat\sigma_2 = 0$ can be removed at the cost of a less tight lower bound. For general σ, we have that

$$\sup_{\{\hat{\boldsymbol{w}}_i\}_{1\le i\le R}\in\hat{\mathcal{S}}_\epsilon} \mathbb{E}\big\{|y(\boldsymbol{x}) - \hat y(\boldsymbol{x})|^2\big\}$$
$$\ge \left(\min_{\substack{a\in\mathbb{R}\\ \boldsymbol{A}\in\mathbb{R}^{d\times d}}} \mathbb{E}\left\{|y(\boldsymbol{x}) - (a + \langle \boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}\rangle)|^2\right\} - r\cdot\eta(r,d)\right) \cdot\left(1 - \frac{R}{r}\cdot\eta(r,d)\right). \tag{15}$$

Third, Theorem 3 covers regimes different from those of Theorem 2. Indeed, the result of this section guarantees that the generalization error of the neural network is close to that of a trivial predictor for any σ such that $\hat\sigma_2 = 0$ and for r up to d^2 (modulo logarithmic factors), unless the weights are estimated 'better than random', namely with a non-vanishing correlation. We also allow predictors with a number of nodes R that can be larger than the number of nodes r of the original neural network, as long as R and r are of the same order.

The key technical step in the proof is upper bounding the third-order correlation $\sum_{i\le r, j\le R}\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_j\rangle^3/R$ uniformly over all estimates such that $\max_{i,j}|\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_j\rangle| \le \epsilon$. A naive bound would be $r\epsilon^3$, while using the approximate isotropicity of the $\boldsymbol{w}_i$ yields an upper bound of order $\epsilon\max(1, r/d)$. For $\epsilon \approx 1/\sqrt{d}$ this would vanish only in the regime $r \ll d^{3/2}$. In order to obtain a non-trivial result for $r \gg d^{3/2}$, we use the randomness of the $\boldsymbol{w}_i$, together with an epsilon-net argument and several ad-hoc estimates, which eventually yields that $\sum_{i\le r, j\le R}\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_j\rangle^3/R = o(1)$.

4 Learning a Neural Network and Tensor Decomposition

We present reductions from tensor decomposition to the problem of learning the weights of a two-layer neural network. Note that no assumption on the input distribution is necessary. Before giving the statement, let us formally define what we mean when we say that it is algorithmically hard to learn the weights $\{\boldsymbol{w}_i\}_{1\le i\le r}$.

Definition 1 (ϵ-Hardness of Learning)**.** *A weight-learning problem is defined by a triple $(\epsilon, \mathcal{S}, f)$, where $\epsilon \in (0,1)$, $\mathcal{S}$ is a set of possible weights*

$$\mathcal{S} \subseteq \{\{\boldsymbol{w}_i\}_{1\le i\le r} : \|\boldsymbol{w}_i\| = 1,\ \forall\, i \in [r]\}, \tag{16}$$

and $f : \mathcal{S} \to \mathcal{I}$ is a function, where $\mathcal{I}$ denotes a set of inputs. We always assume that r and the size of $\mathcal{I}$ are bounded by polynomials in d. We say that the problem $(\epsilon, \mathcal{S}, f)$ is hard *(or, the problem is* ϵ-hard*) if there is no algorithm $\mathcal{A}$ that, given as input $f(\boldsymbol{w}_1, \dots, \boldsymbol{w}_r)$, fulfills the following two properties:*

(P1) *$\mathcal{A}$ outputs $\{\hat{\boldsymbol{w}}_i\}_{1\le i\le R}$ of unit norm such that, for some $i \in [r]$ and $j \in [R]$, $|\langle \boldsymbol{w}_i, \hat{\boldsymbol{w}}_j\rangle| \ge \epsilon$;*

(P2) *$\mathcal{A}$ has complexity which is polynomial in d.*

The result below, whose proof is contained in Appendix C of the supplementary material, provides a reduction for activation functions that are polynomials whose degree is *at most* the order of the tensor to be decomposed.

Theorem 4 (Learning a Neural Network and Tensor Decomposition)**.** *Fix an integer $\ell \ge 3$ and let $y(\boldsymbol{x})$*

be defined in (6)*, where* σ *is the activation function. For* $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$*, let* $\mathcal{P}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ *be the problem of learning* $\{\boldsymbol{w}_i\}_{1 \le i \le r}$ *given as input* $\{\boldsymbol{x}_j\}_{1 \le j \le n}$ *and* $\{y(\boldsymbol{x}_j)\}_{1 \le j \le n}$*. Then, the following results hold.*

1) Assume that, given as input the tensor $\boldsymbol{T}^{(\ell)} = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes \ell}$*, the problem of learning* $\{\boldsymbol{w}_i\}_{1 \le i \le r} \in \mathcal{S}$ *is* ϵ*-hard in the sense of Definition 1 for some* $\epsilon > 0$*. Let the activation function* σ *be a polynomial with* $\deg(\sigma) \le \ell$ *and* $\mathrm{par}(\sigma) = \mathrm{par}(\ell)$*. Then, for any* $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$*, the problem* $\mathcal{P}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ *is* ϵ*-hard in the sense of Definition 1.*

2) Assume that, given as input the tensors $\boldsymbol{T}^{(\ell)} = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes \ell}$ *and* $\boldsymbol{T}^{(\ell+1)} = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes \ell+1}$*, the problem of learning* $\{\boldsymbol{w}_i\}_{1 \le i \le r} \in \mathcal{S}$ *is* ϵ*-hard in the sense of Definition 1 for some* $\epsilon > 0$*. Let the activation function* σ *be a polynomial with* $\deg(\sigma) \le \ell + 1$*. Then, for any* $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$*, the problem* $\mathcal{P}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ *is* ϵ*-hard in the sense of Definition 1.*

In words, learning a two-layer neural network whose activation function is a polynomial of degree ℓ and assigned parity (i.e., either even or odd) is as hard as solving tensor decomposition given the tensor of order ℓ with the same parity. Furthermore, learning a two-layer neural network whose activation function is a polynomial of degree $\ell+1$ (without any assumption on its parity) is as hard as solving tensor decomposition given the tensors of order ℓ and $\ell+1$. In Appendix D of the supplementary material, we consider a slightly different model of two-layer neural network with an additive error term. By doing so, we can prove a reduction with activation functions that are polynomials with degree *larger* than the order of the tensor.

5 Generalization Error and Tensor Decomposition

We present reductions from tensor decomposition to the problem of finding a predictor of a two-layer neural network with small generalization error. As in Section 4, no assumption is necessary on the distribution of the samples given as input to the learning algorithm. However, when taking the expectation to compute the generalization error, we assume that $\boldsymbol{x} \sim \mathsf{N}(\boldsymbol{0}, \boldsymbol{I}_d)$. The corollary below considers the case of separated and isotropic weights and its proof is readily obtained by combining the results of Theorem 2 and 4.

Corollary 1 (Generalization Error and Tensor Decomposition for Separated Isotropic Weights)**.** *Fix an integer* $\ell \ge 3$*, and, for positive* $\delta, \eta_{\rm avg}$ *and* $\eta_{\rm var}$ *such that* $1 - \delta \cdot (1 + \eta_{\rm var}) \cdot r/d \ge 0$*, let*

$$\mathcal{S}' \subseteq \{\{\boldsymbol{w}_i\}_{1 \le i \le r} : \textit{assumptions } (\mathsf{A1})\text{-}(\mathsf{A4}) \textit{ hold}\}. \tag{17}$$

We have the following results.

1) Assume that, given the tensor $\boldsymbol{T}^{(\ell)} = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes \ell}$*, the problem of learning* $\{\boldsymbol{w}_i\}_{1 \le i \le r} \in \mathcal{S}'$ *is* ϵ*-hard in the sense of Definition 1 for some* $\epsilon > 0$*. Let* $y(\boldsymbol{x})$ *be defined in* (6)*, where* σ *is a polynomial with* $\deg(\sigma) \le \ell$ *and* $\mathrm{par}(\sigma) = \mathrm{par}(\ell)$*. Then, for any* $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$ *and for any polynomial algorithm that, given as input* $\{\boldsymbol{x}_j\}_{1 \le j \le n}$ *and* $\{y(\boldsymbol{x}_j)\}_{1 \le j \le n}$*, outputs* $\{\hat{\boldsymbol{w}}_i\}_{1 \le i \le R}$ *of unit norm, we have that*

$$\begin{aligned} &\mathbb{E}\{|y(\boldsymbol{x}) - \hat{y}(\boldsymbol{x})|^2\} \\ &\ge \left(\min_{a,b \in \mathbb{R}} \mathbb{E}\left\{ \left| y(\boldsymbol{x}) - \left(a + b\,\|\boldsymbol{x}\|^2\right) \right|^2 \right\} - c_1 \right) (1 - c_2)\,, \end{aligned} \tag{18}$$

where $\hat{y}(\boldsymbol{x})$ *is defined in* (7)*, the expectation is with respect to* $\boldsymbol{x} \sim \mathsf{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$ *and the terms* c_1 *and* c_2 *are given by* (9) *with* $\hat{\sigma}_1$ *and* $\hat{\sigma}_2$ *defined in* (5)*. If we also assume that* σ *is even, then* (18) *holds with* c_1 *and* c_2 *given by* (10)*.*

2) Assume that, given the tensors $\boldsymbol{T}^{(\ell)} = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes \ell}$ *and* $\boldsymbol{T}^{(\ell+1)} = \sum_{i=1}^r \boldsymbol{w}_i^{\otimes \ell+1}$*, the problem of learning* $\{\boldsymbol{w}_i\}_{1 \le i \le r} \in \mathcal{S}'$ *is* ϵ*-hard in the sense of Definition 1 for some* $\epsilon > 0$*. Let* $y(\boldsymbol{x})$ *be defined in* (6)*, where* σ *is a polynomial with* $\deg(\sigma) \le \ell + 1$*. Then, for any* $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$ *and for any polynomial algorithm that, given as input* $\{\boldsymbol{x}_j\}_{1 \le j \le n}$ *and* $\{y(\boldsymbol{x}_j)\}_{1 \le j \le n}$*, outputs* $\{\hat{\boldsymbol{w}}_i\}_{1 \le i \le R}$ *of unit norm, we have that* (18) *holds, where* $\hat{y}(\boldsymbol{x})$ *is defined in* (7)*, the expectation is with respect to* $\boldsymbol{x} \sim \mathsf{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$ *and the terms* c_1 *and* c_2 *are given by* (9)*. Furthermore, if* σ *is even, then the terms* c_1 *and* c_2 *are given by* (10)*.*

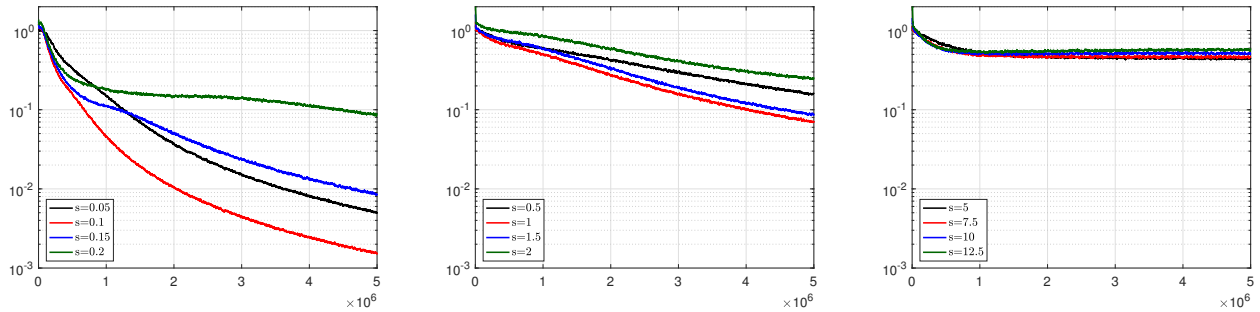
A reduction for the case of random weights is contained in Theorem 1, stated in Section 1. Its proof follows by combining the result of Theorem 3 with $R = r$ with the same proof of Theorem 4.

Let us now summarize briefly some implications of our results. The discussion in Section 2.2 suggests that tensor decomposition is hard in the following cases: if the weights are random vectors, given $\boldsymbol{T}^{(3)}$ and for $r \gg d^{3/2}$; if the weights are separated unit vectors, given $\{\boldsymbol{T}^{(k)}\}_{3 \le k \le \ell}$, for fixed ℓ and for $r \gg d$. By setting $R = r$, in our paper we consider a model similar to that of [Tian, 2017, Sedghi and Anandkumar, 2015, Janzamin et al., 2015, Zhong et al., 2017]. Our results suggest that it will be difficult to extend those recovery schemes to several interesting regimes:

1) $d^{3/2} \ll r \ll d^2$ for random weights and activation function $\sigma(x) = a_0 + a_1 x + a_3 x^3$ for some $a_0, a_1, a_3 \in \mathbb{R}$.

2) $d \ll r \ll d/\epsilon$ for separated isotropic weights and polynomial activation function;

3) $d \ll r \ll d/\epsilon^2$ for separated isotropic weights and even polynomial activation function.



(a) Generalization error, $r = 50$. (b) Generalization error, $r = 350$. (c) Generalization error, $r = 2500$.

(d) Estimation error, $r = 50$. (e) Estimation error, $r = 350$. (f) Estimation error, $r = 2500$.

Figure 1: Performance of stochastic gradient descent with Gaussian feature vectors and Haar distributed weights for different numbers of hidden nodes r. Different curves correspond to a different step size s.

6 Numerical Experiments

We consider a two-layer neural network with input dimension $d = 50$ and r hidden nodes, with $r \in \{50, 350, 2500\}$. The activation function σ is equal to $\tanh(5x/2)$. The weights $\{\boldsymbol{w}_i\}_{1 \le i \le r}$ are obtained by concatenating r/d random unitary matrices of size $d \times d$ that are independent and identically distributed according to the Haar measure. In particular, each of these matrices is obtained from the SVD of a matrix whose entries that are $\sim_{\text{i.i.d.}} \mathsf{N}(0,1)$. Then, the weights are centered by subtracting their empirical mean. We generate $n = 5 \cdot 10^6$ samples $\{(\boldsymbol{x}_j, y(\boldsymbol{x}_j))\}_{1 \le j \le n}$, where $\{\boldsymbol{x}_j\}_{1 \le j \le n} \sim_{\text{i.i.d.}} \mathsf{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ and $y(\boldsymbol{x})$ is given by (6). We perform n iterations of stochastic gradient descent with a fixed step size s. We also perform Polyak-Ruppert averaging, i.e., the algorithm outputs at step $j \in [n]$ the average of the estimates obtained so far, in order to smooth the performance of the algorithm. Let $\hat{y}_j(\boldsymbol{x})$ be the predictor given by (1), where $\{\hat{\boldsymbol{w}}_i\}_{1 \le i \le r}$ are the weights outputted by the algorithm at step $j \in [n]$.

The results are presented in Figure 1, where we plot two different performance metrics. On top, we have the generalization error $(y(\boldsymbol{x}_j) - \hat{y}_j(\boldsymbol{x}_j))^2/(y(\boldsymbol{x}_j) - y_{\text{LS}}(\boldsymbol{x}_j))^2$, for $j \in [n]$, where $y_{\text{LS}}(\boldsymbol{x}_j)$ is the prediction of the least-squares estimator with access only to the norm of the input. In order to obtain a smoother curve, we average the results over a window of size 10^4. Note that, for $r = 2500$, the estimator $y_{\text{LS}}(\boldsymbol{x}_j)$ generalizes poorly, in the sense that its loss is close to $\text{Var}\{y(\boldsymbol{x})\}$. On the bottom, we have the weight estimation error $\frac{1}{2r}\sum_{i=1}^{r} \min_{j \in [r]} \|\hat{\boldsymbol{w}}_i - \boldsymbol{w}_j\|^2 + \frac{1}{2r}\sum_{i=1}^{r} \min_{j \in [r]} \|\hat{\boldsymbol{w}}_j - \boldsymbol{w}_i\|^2$, which represents the average of the minimum distances between the ground-truth weights $\{\boldsymbol{w}_i\}_{1 \le i \le r}$ and the estimated weights $\{\hat{\boldsymbol{w}}_i\}_{1 \le i \le r}$. Different pairs of plots correspond to different choices for the number of hidden nodes r, and in each plot we have several curves for different values of the step s. Similar results are obtained by taking random weights of the form (11).

The numerical results corroborate the picture that we have proved in the paper for Gaussian features. As the number of hidden units r becomes much larger than d (from $r = d$ to $r \approx d^{3/2}$ and $r \approx d^2$), the problem of learning the weights of the neural network becomes harder and harder, similarly to what happens for tensor decomposition. Furthermore, the generalization error has the same qualitative behavior of the weight estimation error: the neural network generalizes well if and only if the weights are learned accurately.

Acknowledgements

M. M. was supported by an Early Postdoc.Mobility fellowship from the Swiss National Science Foundation and by the Simons Institute for the Theory of Computing. A. M. was partially supported by grants NSF DMS-1613091, CCF-1714305, IIS-1741162 and ONR N00014-18-1-2729. This work was carried out in part while the authors were visiting the Simons Institute for the Theory of Computing.

References

[Anandkumar et al., 2015] Anandkumar, A., Ge, R., and Janzamin, M. (2015). Learning overcomplete latent variable models through tensor methods. In *Proc. of Conference on Learning Theory (COLT)*, volume 40, pages 36–112.

[Anandkumar et al., 2017] Anandkumar, A., Ge, R., and Janzamin, M. (2017). Analyzing tensor power method dynamics in overcomplete regime. *Journal of Machine Learning Research (JMLR)*, 18:1–40.

[Arora et al., 2014] Arora, S., Bhaskara, A., Ge, R., and Ma, T. (2014). Provable bounds for learning some deep representations. In *Proc. of International Conference on Machine Learning (ICML)*, pages 584–592.

[Barak et al., 2016] Barak, B., Hopkins, S. B., Kelner, J., Kothari, P., Moitra, A., and Potechin, A. (2016). A nearly tight sum-of-squares lower bound for the planted clique problem. In *Proc. of Annual Symposium on Foundations of Computer Science (FOCS)*, pages 428–437.

[Barak et al., 2015] Barak, B., Kelner, J. A., and Steurer, D. (2015). Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proc. of Annual ACM Symposium on Theory of Computing (STOC)*, pages 143–151.

[Barak and Steurer, 2014] Barak, B. and Steurer, D. (2014). Sum-of-squares proofs and the quest toward optimal algorithms. arXiv:1404.5236.

[Bartlett and Ben-David, 1999] Bartlett, P. and Ben-David, S. (1999). Hardness results for neural network approximation problems. In *Proc. of Computational Learning Theory (COLT)*, pages 50–62.

[Berthet and Rigollet, 2013] Berthet, Q. and Rigollet, P. (2013). Complexity theoretic lower bounds for sparse principal component detection. In *Proc. of Conference on Learning Theory (COLT)*, pages 1046–1066.

[Bhaskara et al., 2014] Bhaskara, A., Charikar, M., Moitra, A., and Vijayaraghavan, A. (2014). Smoothed analysis of tensor decompositions. In *Proc. of Annual ACM Symposium on Theory of Computing (STOC)*, pages 594–603.

[Blum and Rivest, 1989] Blum, A. and Rivest, R. L. (1989). Training a 3-node neural network is NP-complete. In *Advances in Neural Information Processing Systems (NIPS)*, pages 494–501.

[Bousquet and Elisseeff, 2002] Bousquet, O. and Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research (JMLR)*, 2:499–526.

[Brennan et al., 2018] Brennan, M., Bresler, G., and Huleihel, W. (2018). Reducibility and computational lower bounds for problems with planted sparse structure. In *Proc. of Conference On Learning Theory (COLT)*, volume 75, pages 48–166.

[Brutzkus and Globerson, 2017] Brutzkus, A. and Globerson, A. (2017). Globally optimal gradient descent for a convnet with gaussian inputs. arXiv:1702.07966.

[Daniely, 2016] Daniely, A. (2016). Complexity theoretic limitations on learning halfspaces. In *Proceedings of Annual ACM symposium on Theory of Computing (STOC)*, pages 105–117.

[De Lathauwer et al., 1996] De Lathauwer, L., De Moor, B., and Vandewalle, J. (1996). Blind source separation by simultaneous third-order tensor diagonalization. In *Proc. of European Signal Processing Conference (EUSIPCO)*, pages 1–4.

[Freeman and Bruna, 2017] Freeman, C. D. and Bruna, J. (2017). Topology and geometry of half-rectified network optimization. In *Proc. of International Conference on Learning Representations (ICLR)*.

[Ge et al., 2018] Ge, R., Lee, J. D., and Ma, T. (2018). Learning one-hidden-layer neural networks with landscape design. In *Proc. of International Conference on Learning Representations (ICLR)*.

[Ge and Ma, 2015] Ge, R. and Ma, T. (2015). Decomposing overcomplete 3rd order tensors using sum-of-squares algorithms. In *Proc. of APPROX-RANDOM*, pages 829–849.

[Harshman, 1970] Harshman, R. A. (1970). Foundations of the parafac procedure: models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, (16):1–84.

[Håstad, 1990] Håstad, J. (1990). Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654.

[Hillar and Lim, 2013] Hillar, C. J. and Lim, L.-H. (2013). Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):1–45.

[Hopkins et al., 2017] Hopkins, S. B., Kothari, P. K., Potechin, A., Raghavendra, P., Schramm, T., and Steurer, D. (2017). The power of sum-of-squares for detecting hidden structures. In *Proc. of IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 720–731.

[Hopkins et al., 2016] Hopkins, S. B., Schramm, T., Shi, J., and Steurer, D. (2016). Fast spectral algorithms from sum-of-squares proofs: tensor decomposition and planted sparse vectors. In *Proc. of Annual ACM Symposium on Theory of Computing (STOC)*, pages 178–191.

[Hopkins et al., 2015] Hopkins, S. B., Shi, J., and Steurer, D. (2015). Tensor principal component analysis via sum-of-square proofs. In *Proc. of Conference on Learning Theory (COLT)*, pages 956–1006.

[Janzamin et al., 2015] Janzamin, M., Sedghi, H., and Anandkumar, A. (2015). Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. `arXiv:1506.08473`.

[Kuhlmann, 2000] Kuhlmann, C. (2000). Hardness results for general two-layer neural networks. In *Proc. of Computational Learning Theory (COLT)*, pages 275–285.

[Ma et al., 2016] Ma, T., Shi, J., and Steurer, D. (2016). Polynomial-time tensor decompositions with sum-of-squares. In *Proc. of IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 438–446.

[Panigrahy et al., 2018] Panigrahy, R., Rahimi, A., Sachdeva, S., and Zhang, Q. (2018). Convergence results for neural networks via electrodynamics. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 94. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Safran and Shamir, 2016] Safran, I. and Shamir, O. (2016). On the quality of the initial basin in over-specified neural networks. In *Proc. of International Conference on Machine Learning (ICML)*, pages 774–782.

[Schramm and Steurer, 2017] Schramm, T. and Steurer, D. (2017). Fast and robust tensor decomposition with applications to dictionary learning. *Proc. of Machine Learning Research (PMLR)*, 65:1–34.

[Schramm and Weitz, 2015] Schramm, T. and Weitz, B. (2015). Low-rank matrix completion with adversarial missing entries. `arXiv:1506.03137`.

[Sedghi and Anandkumar, 2015] Sedghi, H. and Anandkumar, A. (2015). Provable methods for training neural networks with sparse connectivity. In *Proc. of International Conference on Learning Representation (ICLR)*.

[Shalev-Shwartz and Ben-David, 2014] Shalev-Shwartz, S. and Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[Shamir, 2018] Shamir, O. (2018). Distribution-specific hardness of learning neural networks. *Journal of Machine Learning Research (JMLR)*, 19(32).

[Šíma, 2002] Šíma, J. (2002). Training a single sigmoidal neuron is hard. *Neural Computation*, 14(11):2709–2728.

[Soltanolkotabi et al., 2019] Soltanolkotabi, M., Javanmard, A., and Lee, J. D. (2019). Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769.

[Soudry and Carmon, 2016] Soudry, D. and Carmon, Y. (2016). No bad local minima: Data independent training error guarantees for multilayer neural networks. `arXiv:1605.08361`.

[Tian, 2017] Tian, Y. (2017). Symmetry-breaking convergence analysis of certain two-layered neural networks with ReLU nonlinearity. In *Proc. of International Conference on Learning Representation (ICLR)*.

[Zhong et al., 2017] Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. (2017). Recovery guarantees for one-hidden-layer neural networks. In *Proc. of International Conference on Machine Learning (ICML)*, pages 4140–4149.