Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks

Sanjeev Arora [*] [1] [2] **Simon S. Du** [*] [3] **Wei Hu** [*] [1] **Zhiyuan Li** [*] [1] **Ruosong Wang** [*] [3]

Abstract

Recent works have cast some light on the mystery of why deep nets fit any data and generalize despite being very overparametrized. This paper analyzes training and generalization for a simple 2-layer ReLU net with random initialization, and provides the following improvements over recent works:

(i) Using a tighter characterization of training speed than recent papers, an explanation for why training a neural net with random labels leads to slower training, as originally observed in [Zhang et al. ICLR'17].

(ii) Generalization bound independent of network size, using a data-dependent complexity measure. Our measure distinguishes clearly between random labels and true labels on MNIST and CIFAR, as shown by experiments. Moreover, recent papers require sample complexity to increase (slowly) with the size, while our sample complexity is completely independent of the network size.

(iii) Learnability of a broad class of smooth functions by 2-layer ReLU nets trained via gradient descent.

The key idea is to track dynamics of training and generalization via properties of a related kernel.

1. Introduction

The well-known work of Zhang et al. (2017) highlighted intriguing experimental phenomena about deep net training – specifically, optimization and generalization – and asked whether theory could explain them. They showed that sufficiently powerful nets (with vastly more parameters than number of training samples) can attain zero training error, regardless of whether the data is properly labeled or randomly labeled. Obviously, training with randomly labeled data cannot generalize, whereas training with properly labeled data generalizes. See Figure 2 replicating some of these results.

Recent papers have begun to provide explanations, showing that gradient descent can allow an overparametrized multi-layer net to attain arbitrarily low training error on fairly generic datasets (Du et al., 2018a;c; Li & Liang, 2018; Allen-Zhu et al., 2018b; Zou et al., 2018), provided the amount of overparametrization is a high polynomial of the relevant parameters (i.e. vastly more than the overparametrization in (Zhang et al., 2017)). Under further assumptions it can also be shown that the trained net generalizes (Allen-Zhu et al., 2018a). But some issues were not addressed in these papers, and the goal of the current paper is to address them.

First, the experiments in (Zhang et al., 2017) show that though the nets attain zero training error on even random data, the convergence rate is much slower. See Figure 1.

Question 1. *Why do true labels give faster convergence rate than random labels for gradient descent?*

The above papers do not answer this question, since their proof of convergence does not distinguish between good and random labels.

The next issue is about generalization: clearly, some property of properly labeled data controls generalization, but what? Classical measures used in generalization theory such as VC-dimension and Rademacher complexity are much too pessimistic. A line of research proposed norm-based (e.g. (Bartlett et al., 2017a)) and compression-based bounds (Arora et al., 2018). But the sample complexity upper bounds obtained are still far too weak. Furthermore they rely on some property of the trained net that is *revealed/computed* at the end of training. There is no property of data alone that determine upfront whether the trained net will generalize. A recent paper (Allen-Zhu et al., 2018a) assumed that there exists an underlying (unknown) neural network that achieves low error on the data distribution, and the amount of data available is quite a bit more than the min-

[*]Alphabetical order [1]Princeton University, Princeton, NJ, USA [2]Institute for Advanced Study, Princeton, NJ, USA [3]Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Wei Hu <huwei@cs.princeton.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

imum number of samples needed to learn this underlying neural net. Under this condition, the overparametrized net (which has way more parameters) can learn in a way that generalizes. However, it is hard to verify from data whether this assumption is satisfied, even after the larger net has finished training.[1] Thus the assumption is in some sense unverifiable.

Question 2. *Is there an easily verifiable complexity measure that can differentiate true labels and random labels?*

Without explicit regularization, to attack this problem, one must resort to algorithm-dependent generalization analysis. One such line of work established that first-order methods can automatically find minimum-norm/maximum-margin solutions that fit the data in the settings of logistic regression, deep linear networks, and symmetric matrix factorization (Soudry et al., 2018; Gunasekar et al., 2018a;b; Ji & Telgarsky, 2018; Li et al., 2018b). However, how to extend these results to non-linear neural networks remains unclear (Wei et al., 2018). Another line of algorithm-dependent analysis of generalization (Hardt et al., 2015; Mou et al., 2017; Chen et al., 2018) used stability of specific optimization algorithms that satisfy certain generic properties like convexity, smoothness, etc. However, as the number of epochs becomes large, these generalization bounds are vacuous.

Our results. We give a new analysis that provides answers to Questions 1 and 2 for overparameterized two-layer neural networks with ReLU activation trained by gradient descent (GD), when the number of neurons in the hidden layer is sufficiently large. In this setting, Du et al. (2018c) have proved that GD with random initialization can achieve zero training error for any non-degenerate data. We give a more refined analysis of the trajectory of GD which enables us to provide answers to Questions 1 and 2. In particular:

- In Section 4, using the trajectory of the network predictions on the training data during optimization, we accurately estimate the magnitude of training loss in each iteration. Our key finding is that the number of iterations needed to achieve a target accuracy depends on the projections of data labels on the eigenvectors of a certain Gram matrix to be defined in Equation (3). On MNIST and CIFAR datasets, we find that such projections are significantly different for true labels and random labels, and as a result we are able to answer Question 1.

- In Section 5, we give a generalization bound for the solution found by GD, based on accurate estimates of how much the network parameters can move during optimization (in suitable norms). Our generalization bound depends on a *data-dependent complexity measure* (c.f. Equation (10)), and notably, is completely independent of the number of hidden units in the network. Again, we test this complexity measure on MNIST and CIFAR, and find that the complexity measures for true and random labels are significantly different, which thus answers Question 2.

  Notice that because zero training error is achieved by the solution found by GD, a generalization bound is an upper bound on the error on the data distribution (test error). We also remark that our generalization bound is valid for *any data labels* – it does not require the existence of a small ground-truth network as in (Allen-Zhu et al., 2018a). Moreover, our bound can be efficiently computed for any data labels.

- In Section 6, we further study what kind of functions can be provably learned by two-layer ReLU networks trained by GD. Combining the optimization and generalization results, we uncover a broad class of learnable functions, including linear functions, two-layer neural networks with polynomial activation $\phi(z) = z^{2l}$ or cosine activation, etc. Our requirement on the smoothness of learnable functions is weaker than that in (Allen-Zhu et al., 2018a).

[1]In Section 2, we discuss the related works in more details.

Finally, we note that the intriguing generalization phenomena in deep learning were observed in kernel methods as well (Belkin et al., 2018). The analysis in the current paper is also related to a kernel from the ReLU activation (c.f. Equation (3)).

2. Related Work

In this section we survey previous works on optimization and generalization aspects of neural networks.

Optimization. Many papers tried to characterize geometric landscapes of objective functions (Safran & Shamir, 2017; Zhou & Liang, 2017; Freeman & Bruna, 2016; Hardt & Ma, 2016; Nguyen & Hein, 2017; Kawaguchi, 2016; Venturi et al., 2018; Soudry & Carmon, 2016; Du & Lee, 2018; Soltanolkotabi et al., 2018; Haeffele & Vidal, 2015). The hope is to leverage recent advance in first-order algorithms (Ge et al., 2015; Lee et al., 2016; Jin et al., 2017) which showed that if the landscape satisfies (1) all local minima are global and (2) all saddle points are strict (i.e., there exists a negative curvature), then first-order methods can escape all saddle points and find a global minimum. Unfortunately, these desired properties do not hold even for simple non-linear shallow neural networks (Yun et al., 2018) or 3-layer linear neural networks (Kawaguchi, 2016).

Another approach is to directly analyze trajectory of the optimization method and to show convergence to global minimum. A series of papers made strong assumptions on input distribution as well as realizability of labels, and showed global convergence of (stochastic) gradient descent for some shallow neural networks (Tian, 2017; Soltanolkotabi, 2017; Brutzkus & Globerson, 2017; Du et al., 2017a;b; Li & Yuan, 2017). Some local convergence results have also been proved (Zhong et al., 2017; Zhang et al., 2018). However, these assumptions are not satisfied in practice.

For two-layer neural networks, a line of papers used mean field analysis to establish that for infinitely wide neural networks, the empirical distribution of the neural network parameters can be described as a Wasserstein gradient flow (Mei et al., 2018; Chizat & Bach, 2018a; Sirignano & Spiliopoulos, 2018; Rotskoff & Vanden-Eijnden, 2018; Wei et al., 2018). However, it is unclear whether this framework can explain the behavior of first-order methods on finite-size neural networks.

Recent breakthroughs were made in understanding optimization of overparameterized neural networks through the trajectory-based approach. They proved global polynomial time convergence of (stochastic) gradient descent on non-linear neural networks for minimizing empirical risk. Their proof techniques can be roughly classified into two categories. Li & Liang (2018); Allen-Zhu et al. (2018b); Zou et al. (2018) analyzed the trajectory of parameters and showed that on the trajectory, the objective function satisfies certain gradient dominance property. On the other hand, (Du et al., 2018a;c) analyzed the trajectory of network predictions on training samples and showed that it enjoys a strongly-convex-like property.

Generalization. It is well known that the VC-dimension of neural networks is at least linear in the number of parameters (Bartlett et al., 2017b), and therefore classical VC theory cannot explain the generalization ability of modern neural networks with more parameters than training samples. Researchers have proposed norm-based generalization bounds (Bartlett & Mendelson, 2002; Bartlett et al., 2017a; Neyshabur et al., 2015; 2017; 2019; Konstantinos et al., 2017; Golowich et al., 2017; Li et al., 2018a) and compression-based bounds (Arora et al., 2018). Dziugaite & Roy (2017); Zhou et al. (2019) used the PAC-Bayes approach to compute non-vacuous generalization bounds for MNIST and ImageNet, respectively. All these bounds are *posterior* in nature – they depend on certain properties of the *trained* neural networks. Therefore, one has to finish training a neural network to know whether it can generalize. Comparing with these results, our generalization bound only depends on training data and can be calculated without actually training the neural network.

Another line of work assumed the existence of a true model, and showed that the (regularized) empirical risk minimizer has good generalization with sample complexity that depends on the true model (Du et al., 2018b; Ma et al., 2018; Imaizumi & Fukumizu, 2018). These papers ignored the difficulty of optimization, while we are able to prove generalization of the solution found by gradient descent. Furthermore, our generic generalization bound does not assume the existence of any true model.

Our paper is closely related to (Allen-Zhu et al., 2018a) which showed that two-layer overparametrized neural networks trained by randomly initialized stochastic gradient descent can learn a class of infinite-order smooth functions. In contrast, our generalization bound depends on a data-dependent complexity measure that can be computed for any dataset, without assuming any ground-truth model. Furthermore, as a consequence of our generic bound, we also show that two-layer neural networks can learn a class of infinite-order smooth functions, with a less strict requirement for smoothness. Allen-Zhu et al. (2018a) also studied the generalization performance of three-layer neural nets.

Lastly, our work is related to kernel methods, especially recent discoveries of the connection between deep learning and kernels (Jacot et al., 2018; Chizat & Bach, 2018b; Daniely et al., 2016; Daniely, 2017). Our analysis utilized several properties of a related kernel from the ReLU activation (c.f. Equation (3)).

3. Preliminaries and Overview of Results

Notation. We use bold-faced letters for vectors and matrices. For a matrix $\mathbf{A}$, let $\mathbf{A}_{ij}$ be its (i,j)-th entry. We use $\|\cdot\|_2$ to denote the Euclidean norm of a vector or the spectral norm of a matrix, and use $\|\cdot\|_F$ to denote the Frobenius norm of a matrix. Denote by $\lambda_{\min}(\mathbf{A})$ the minimum eigenvalue of a symmetric matrix $\mathbf{A}$. Let $\text{vec}\left(\mathbf{A}\right)$ be the vectorization of a matrix $\mathbf{A}$ in column-first order. Let $\mathbf{I}$ be the identity matrix and $[n] = \{1, 2, \ldots, n\}$. Denote by $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Denote by $\sigma(\cdot)$ the ReLU function $\sigma(z) = \max\{z, 0\}$. Denote by $\mathbb{I}\{E\}$ the indicator function for an event E.

3.1. Setting: Two-Layer Neural Network Trained by Randomly Initialized Gradient Descent

We consider a two-layer ReLU activated neural network with m neurons in the hidden layer:

$$f_{\mathbf{W},\mathbf{a}}(\mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma\left(\mathbf{w}_r^\top \mathbf{x}\right),$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input, $\mathbf{w}_1, \ldots, \mathbf{w}_m \in \mathbb{R}^d$ are weight vectors in the first layer, $a_1, \ldots, a_m \in \mathbb{R}$ are weights in the second layer. For convenience we denote $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathbb{R}^{d\times m}$ and $\mathbf{a} = (a_1, \ldots, a_m)^\top \in \mathbb{R}^m$.

We are given n input-label samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ drawn i.i.d. from an underlying data distribution $\mathcal{D}$ over $\mathbb{R}^d \times \mathbb{R}$. We denote $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{d\times n}$ and $\mathbf{y} = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$. For simplicity, we assume that for $(\mathbf{x}, y)$ sampled from $\mathcal{D}$, we have $\|\mathbf{x}\|_2 = 1$ and $|y| \le 1$.

We train the neural network by *randomly initialized gradient descent (GD)* on the quadratic loss over data S. In particular, we first initialize the parameters randomly:

$$\mathbf{w}_r(0) \sim \mathcal{N}(\mathbf{0}, \kappa^2\mathbf{I}), a_r \sim \text{unif}\left(\{-1, 1\}\right), \quad \forall r \in [m], \tag{1}$$

where $0 < \kappa \le 1$ controls the magnitude of initialization, and all randomnesses are independent. We then fix the second layer $\mathbf{a}$ and optimize the first layer $\mathbf{W}$ through GD on the following objective function:

$$\Phi(\mathbf{W}) = \frac{1}{2}\sum_{i=1}^n \left(y_i - f_{\mathbf{W},\mathbf{a}}(\mathbf{x}_i)\right)^2. \tag{2}$$

The GD update rule can be written as:[2]

$$\mathbf{w}_r(k+1) - \mathbf{w}_r(k) = -\eta \frac{\partial \Phi(\mathbf{W}(k))}{\partial \mathbf{w}_r}$$
$$= -\eta \frac{a_r}{\sqrt{m}} \sum_{i=1}^n (f_{\mathbf{W}(k),\mathbf{a}}(\mathbf{x}_i) - y_i)\mathbb{I}\left\{\mathbf{w}_r(k)^\top \mathbf{x}_i \ge 0\right\}\mathbf{x}_i,$$

where $\eta > 0$ is the learning rate.

3.2. The Gram Matrix from ReLU Kernel

Given $\{\mathbf{x}_i\}_{i=1}^n$, we define the following *Gram matrix* $\mathbf{H}^\infty \in \mathbb{R}^{n\times n}$ as follows:

$$\mathbf{H}^\infty_{ij} = \mathbb{E}_{\mathbf{w}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\mathbf{x}_i^\top \mathbf{x}_j \mathbb{I}\left\{\mathbf{w}^\top \mathbf{x}_i \ge 0, \mathbf{w}^\top \mathbf{x}_j \ge 0\right\}\right]$$
$$= \frac{\mathbf{x}_i^\top \mathbf{x}_j\left(\pi - \arccos(\mathbf{x}_i^\top \mathbf{x}_j)\right)}{2\pi}, \quad \forall i, j \in [n]. \tag{3}$$

This matrix can be viewed as a Gram matrix from a kernel associated with the ReLU function, and has been studied in (Xie et al., 2017; Tsuchida et al., 2017; Du et al., 2018c).

In our setting of training a two-layer ReLU network, Du et al. (2018c) showed that if $\mathbf{H}^\infty$ is positive definite, GD converges to 0 training loss if m is sufficiently large:

Theorem 3.1 ((Du et al., 2018c)[3]). *Assume* $\lambda_0 = \lambda_{\min}(\mathbf{H}^\infty) > 0$. *For* $\delta \in (0, 1)$, *if* $m = \Omega\left(\frac{n^6}{\lambda_0^4\kappa^2\delta^3}\right)$ *and* $\eta = O\left(\frac{\lambda_0}{n^2}\right)$, *then with probability at least* $1 - \delta$ *over the random initialization* (1), *we have:*

- $\Phi(\mathbf{W}(0)) = O(n/\delta)$;
- $\Phi(\mathbf{W}(k+1)) \le \left(1 - \frac{\eta\lambda_0}{2}\right)\Phi(\mathbf{W}(k)), \ \forall k \ge 0.$

Our results on optimization and generalization also crucially depend on this matrix $\mathbf{H}^\infty$.

3.3. Overview of Our Results

Now we give an informal description of our main results. It assumes that the initialization magnitude κ is sufficiently small and the network width m is sufficiently large (to be quantified later).

The following theorem gives a precise characterization of how the objective decreases to 0. It says that this process is essentially determined by a power method for matrix $\mathbf{I} - \eta\mathbf{H}^\infty$ applied on the label vector $\mathbf{y}$.

Theorem 3.2 (Informal version of Theorem 4.1). *With high probability we have:*

$$\Phi(\mathbf{W}(k)) \approx \frac{1}{2}\left\|(\mathbf{I} - \eta\mathbf{H}^\infty)^k \mathbf{y}\right\|_2^2, \quad \forall k \ge 0.$$

As a consequence, we are able to distinguish the convergence rates for different labels $\mathbf{y}$, which can be determined by the projections of $\mathbf{y}$ on the eigenvectors of $\mathbf{H}^\infty$. This allows us to obtain an answer to Question 1. See Section 4 for details.

Our main result for generalization is the following:

Theorem 3.3 (Informal version of Theorem 5.1). *For any* 1*-Lipschitz loss function, the generalization error of the two-layer ReLU network found by GD is at most*

$$\sqrt{\frac{2\mathbf{y}^\top(\mathbf{H}^\infty)^{-1}\mathbf{y}}{n}}. \tag{4}$$

Notice that our generalization bound (4) can be computed from data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, and is completely independent of the network width m. We observe that this bound can clearly distinguish true labels and random labels, thus providing an answer to Question 2. See Section 5 for details.

Finally, using Theorem 3.3, we prove that we can use our two-layer ReLU network trained by GD to learn a broad class of functions, including linear functions, two-layer neural networks with polynomial activation $\phi(z) = z^{2l}$ or cosine activation, etc. See Section 6 for details.

3.4. Additional Notation

We introduce some additional notation that will be used.

Define $u_i = f_{\mathbf{W},\mathbf{a}}(\mathbf{x}_i)$, i.e., the network's prediction on the i-th input. We also use $\mathbf{u} = (u_1, \ldots, u_n)^\top \in \mathbb{R}^n$ to denote all n predictions. Then we have $\Phi(\mathbf{W}) = \frac{1}{2}\|\mathbf{y} - \mathbf{u}\|_2^2$ and

[2]Since ReLU is not differentiable at 0, we just define "gradient" using this formula, and this is indeed what is used in practice.

[3]Du et al. (2018c) only considered the case $\kappa = 1$, but it is straightforward to generalize their result to general κ at the price of an extra $1/\kappa^2$ factor in m.

the gradient of Φ can be written as:

$$\frac{\partial \Phi(\mathbf{W})}{\partial \mathbf{w}_r} = \frac{1}{\sqrt{m}} a_r \sum_{i=1}^{n} (u_i - y_i) \mathbb{I}_{r,i} \mathbf{x}_i, \quad \forall r \in [m], \quad (5)$$

where $\mathbb{I}_{r,i} = \mathbb{I}\left\{\mathbf{w}_r^\top \mathbf{x}_i \geq 0\right\}$.

We define two matrices $\mathbf{Z}$ and $\mathbf{H}$ which will play a key role in our analysis of the GD trajectory:

$$\mathbf{Z} = \frac{1}{\sqrt{m}} \begin{pmatrix} \mathbb{I}_{1,1} a_1 \mathbf{x}_1 & \cdots & \mathbb{I}_{1,n} a_1 \mathbf{x}_n \\ \vdots & \ddots & \vdots \\ \mathbb{I}_{m,1} a_m \mathbf{x}_1 & \cdots & \mathbb{I}_{m,n} a_m \mathbf{x}_n \end{pmatrix} \in \mathbb{R}^{md \times n},$$

and $\mathbf{H} = \mathbf{Z}^\top \mathbf{Z}$. Note that

$$\mathbf{H}_{ij} = \frac{\mathbf{x}_i^\top \mathbf{x}_j}{m} \sum_{r=1}^{m} \mathbb{I}_{r,i} \mathbb{I}_{r,j}, \quad \forall i, j \in [n].$$

With this notation we have a more compact form of the gradient (5):

$$\operatorname{vec}\left(\nabla \Phi(\mathbf{W})\right) = \mathbf{Z}(\mathbf{u} - \mathbf{y}).$$

Then the GD update rule is:

$$\operatorname{vec}\left(\mathbf{W}(k+1)\right) = \operatorname{vec}\left(\mathbf{W}(k)\right) - \eta \mathbf{Z}(k)(\mathbf{u}(k) - y), \quad (6)$$

for $k = 0, 1, \ldots$. Throughout the paper, we use k as the iteration number, and also use k to index all variables that depend on $\mathbf{W}(k)$. For example, we have $u_i(k) = f_{\mathbf{W}(k),\mathbf{a}}(\mathbf{x}_i)$, $\mathbb{I}_{r,i}(k) = \mathbb{I}\left\{\mathbf{w}_r(k)^\top \mathbf{x}_i \geq 0\right\}$, etc.

4. Analysis of Convergence Rate

Although Theorem 3.1 already predicts linear convergence of GD to 0 loss, it only provides an upper bound on the loss and does not distinguish different types of labels. In particular, it cannot answer Question 1. In this section we give a fine-grained analysis of the convergence rate.

Recall the loss function $\Phi(\mathbf{W}) = \frac{1}{2}\left\|\mathbf{y} - \mathbf{u}\right\|_2^2$. Thus, it is equivalent to study how fast the sequence $\left\{\mathbf{u}(k)\right\}_{k=0}^{\infty}$ converges to $\mathbf{y}$. Key to our analysis is the observation that when the size of initialization κ is small and the network width m is large, the sequence $\left\{\mathbf{u}(k)\right\}_{k=0}^{\infty}$ stays close to another sequence $\left\{\tilde{\mathbf{u}}(k)\right\}_{k=0}^{\infty}$ which has a *linear* update rule:

$$\begin{aligned} \tilde{\mathbf{u}}(0) &= \mathbf{0}, \\ \tilde{\mathbf{u}}(k+1) &= \tilde{\mathbf{u}}(k) - \eta \mathbf{H}^\infty \left(\tilde{\mathbf{u}}(k) - \mathbf{y}\right), \end{aligned} \quad (7)$$

where $\mathbf{H}^\infty$ is the Gram matrix defined in (3).

Write the eigen-decomposition $\mathbf{H}^\infty = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^\top$, where $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ are orthonormal eigenvectors of $\mathbf{H}^\infty$ and $\lambda_1, \ldots, \lambda_n$ are corresponding eigenvalues. Our main theorem in this section is the following:

Theorem 4.1. *Suppose* $\lambda_0 = \lambda_{\min}(\mathbf{H}^\infty) > 0$, $\kappa = O\left(\frac{\epsilon\delta}{\sqrt{n}}\right)$, $m = \Omega\left(\frac{n^7}{\lambda_0^4 \kappa^2 \delta^4 \epsilon^2}\right)$ *and* $\eta = O\left(\frac{\lambda_0}{n^2}\right)$. *Then with probability at least* $1 - \delta$ *over the random initialization, for all* $k = 0, 1, 2, \ldots$ *we have:*

$$\left\|\mathbf{y} - \mathbf{u}(k)\right\|_2 = \sqrt{\sum_{i=1}^{n} (1 - \eta\lambda_i)^{2k} \left(\mathbf{v}_i^\top \mathbf{y}\right)^2} \pm \epsilon. \quad (8)$$

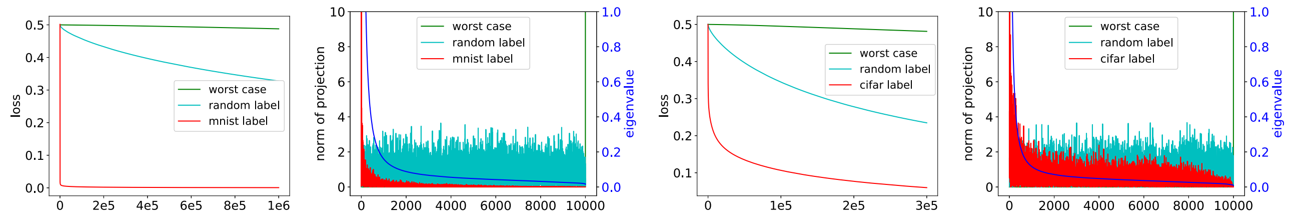
The proof of Theorem 4.1 is given in Appendix C.

In fact, the dominating term $\sqrt{\sum_{i=1}^{n} (1 - \eta\lambda_i)^{2k} \left(\mathbf{v}_i^\top \mathbf{y}\right)^2}$ is exactly equal to $\left\|\mathbf{y} - \tilde{\mathbf{u}}(k)\right\|_2$, which we prove in Section 4.1.

In light of (8), it suffices to understand how fast $\sum_{i=1}^{n} (1 - \eta\lambda_i)^{2k} \left(\mathbf{v}_i^\top \mathbf{y}\right)^2$ converges to 0 as k grows. Define $\xi_i(k) = (1 - \eta\lambda_i)^{2k} (\mathbf{v}_i^\top \mathbf{y})^2$, and notice that each sequence $\{\xi_i(k)\}_{k=0}^{\infty}$ is a geometric sequence which starts at $\xi_i(0) = (\mathbf{v}_i^\top \mathbf{y})^2$ and decreases at ratio $(1 - \eta\lambda_i)^2$. In other words, we can think of decomposing the label vector $\mathbf{y}$ into its projections onto all eigenvectors $\mathbf{v}_i$ of $\mathbf{H}^\infty$: $\left\|\mathbf{y}\right\|_2^2 = \sum_{i=1}^{n} (\mathbf{v}_i^\top \mathbf{y})^2 = \sum_{i=1}^{n} \xi_i(0)$, and the i-th portion shrinks exponentially at ratio $(1 - \eta\lambda_i)^2$. The larger λ_i is, the faster $\{\xi_i(k)\}_{k=0}^{\infty}$ decreases to 0, so in order to have faster convergence we would like the projections of $\mathbf{y}$ onto top eigenvectors to be larger. Therefore we obtain the following intuitive rule to compare the convergence rates on two sets of labels in a qualitative manner (for fixed $\left\|\mathbf{y}\right\|_2$):

- For a set of labels $\mathbf{y}$, if they align with the top eigenvectors, i.e., $(\mathbf{v}_i^\top \mathbf{y})^2$ is large for large λ_i, then gradient descent converges quickly.
- For a set of labels $\mathbf{y}$, if the projections on eigenvectors $\left\{\left(\mathbf{v}_i^\top \mathbf{y}\right)^2\right\}_{i=1}^{n}$ are uniform, or labels align with eigenvectors with respect to small eigenvalues, then gradient descent converges with a slow rate.

Answer to Question 1. We now use this reasoning to answer Question 1. In Figure 1(b), we compute the eigenvalues of $\mathbf{H}^\infty$ (blue curve) for the MNIST dataset. The plot shows the eigenvalues of $\mathbf{H}^\infty$ admit a fast decay. We further compute the projections $\left\{\left|\mathbf{v}_i^\top \mathbf{y}\right|\right\}_{i=1}^{n}$ of true labels (red) and random labels (cyan). We observe that there is a significant difference between the projections of true labels and random labels: true labels align well with top eigenvectors whereas projections of random labels are close to being uniform. Furthermore, according to our theory, if a set of labels align with the eigenvector associated with the least eigenvalue, the convergence rate of gradient descent will be extremely slow. We construct such labels and in Figure 1(a) we indeed observe slow convergence. We repeat the same experiments on CIFAR and have similar observations (Figures 1(c) and



(a) Convergence Rate, MNIST. (b) Eigenval & Projections, MNIST. (c) Convergence Rate, CIFAR. (d) Eigenval & Projections, CIFAR.

Figure 1: In Figures 1(a) and 1(c), we compare convergence rates of gradient descent between using true labels, random labels and the worst case labels (normalized eigenvector of $\mathbf{H}^\infty$ corresponding to $\lambda_{\min}(\mathbf{H}^\infty)$. In Figures 1(b) and 1(d), we plot the eigenvalues of $\mathbf{H}^\infty$ as well as projections of true, random, and worst case labels on different eigenvectors of $\mathbf{H}^\infty$. The experiments use gradient descent on data from two classes of MNIST or CIFAR. The plots clearly demonstrate that true labels have much better alignment with top eigenvectors, thus enjoying faster convergence.

1(d)). These empirical findings support our theory on the convergence rate of gradient descent. See Appendix A for implementation details.

4.1. Proof Sketch of Theorem 4.1

Now we prove $\|\mathbf{y} - \tilde{\mathbf{u}}(k)\|_2^2 = \sum_{i=1}^n (1-\eta\lambda_i)^{2k}\left(\mathbf{v}_i^\top \mathbf{y}\right)^2$. The entire proof of Theorem 4.1 is given in Appendix C, which relies on the fact that the dynamics of $\{\mathbf{u}(k)\}_{k=0}^\infty$ is essentially a perturbed version of (7).

From (7) we have $\tilde{\mathbf{u}}(k+1) - \mathbf{y} = (\mathbf{I} - \eta\mathbf{H}^\infty)(\tilde{\mathbf{u}}(k) - \mathbf{y})$, which implies $\tilde{\mathbf{u}}(k) - \mathbf{y} = (\mathbf{I} - \eta\mathbf{H}^\infty)^k (\tilde{\mathbf{u}}(0) - \mathbf{y}) = -(\mathbf{I} - \eta\mathbf{H}^\infty)^k \mathbf{y}$. Note that $(\mathbf{I} - \eta\mathbf{H}^\infty)^k$ has eigen-decomposition $(\mathbf{I} - \eta\mathbf{H}^\infty)^k = \sum_{i=1}^n (1-\eta\lambda_i)^k \mathbf{v}_i\mathbf{v}_i^\top$ and that $\mathbf{y}$ can be decomposed as $\mathbf{y} = \sum_{i=1}^n (\mathbf{v}_i^\top \mathbf{y})\mathbf{v}_i$. Then we have $\tilde{\mathbf{u}}(k) - \mathbf{y} = -\sum_{i=1}^n (1-\eta\lambda_i)^k (\mathbf{v}_i^\top\mathbf{y})\mathbf{v}_i$, which implies $\|\tilde{\mathbf{u}}(k) - \mathbf{y}\|_2^2 = \sum_{i=1}^n (1-\eta\lambda_i)^{2k}(\mathbf{v}_i^\top\mathbf{y})^2$.

5. Analysis of Generalization

In this section, we study the generalization ability of the two-layer neural network $f_{\mathbf{W}(k),\mathbf{a}}$ trained by GD.

First, in order for optimization to succeed, i.e., zero training loss is achieved, we need a *non-degeneracy* assumption on the data distribution, defined below:

Definition 5.1. *A distribution $\mathcal{D}$ over $\mathbb{R}^d \times \mathbb{R}$ is (λ_0, δ, n)-non-degenerate, if for n i.i.d. samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ from $\mathcal{D}$, with probability at least $1-\delta$ we have $\lambda_{\min}(\mathbf{H}^\infty) \geq \lambda_0 > 0$.*

Remark 5.1. *Note that as long as no two $\mathbf{x}_i$ and $\mathbf{x}_j$ are parallel to each other, we have $\lambda_{\min}(\mathbf{H}^\infty) > 0$. (See (Du et al., 2018c)). For most real-world distributions, any two training inputs are not parallel.*

Our main theorem is the following:

Theorem 5.1. *Fix a failure probability $\delta \in (0,1)$. Suppose our data $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ are i.i.d. samples from a $(\lambda_0, \delta/3, n)$-non-degenerate distribution $\mathcal{D}$, and $\kappa = O\left(\frac{\lambda_0\delta}{n}\right), m \geq \kappa^{-2}\mathrm{poly}\left(n, \lambda_0^{-1}, \delta^{-1}\right)$. Consider any loss function $\ell : \mathbb{R}\times\mathbb{R} \to [0,1]$ that is 1-Lipschitz in the first argument such that $\ell(y,y) = 0$. Then with probability at least $1-\delta$ over the random initialization and the training samples, the two-layer neural network $f_{\mathbf{W}(k),\mathbf{a}}$ trained by GD for $k \geq \Omega\left(\frac{1}{\eta\lambda_0}\log\frac{n}{\delta}\right)$ iterations has population loss $L_{\mathcal{D}}(f_{\mathbf{W}(k),\mathbf{a}}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\ell(f_{\mathbf{W}(k),\mathbf{a}}(\mathbf{x}), y)\right]$ bounded as:*

$$L_{\mathcal{D}}(f_{\mathbf{W}(k),\mathbf{a}}) \leq \sqrt{\frac{2\mathbf{y}^\top\left(\mathbf{H}^\infty\right)^{-1}\mathbf{y}}{n}} + O\left(\sqrt{\frac{\log\frac{n}{\lambda_0\delta}}{n}}\right). \tag{9}$$

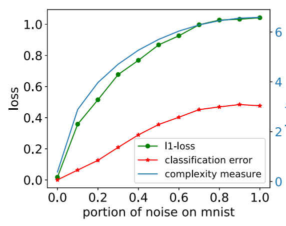
The proof of Theorem 5.1 is given in Appendix D and we sketch the proof in Section 5.1.

Note that in Theorem 5.1 there are three sources of possible failures: (i) failure of satisfying $\lambda_{\min}(\mathbf{H}^\infty) \geq \lambda_0$, (ii) failure of random initialization, and (iii) failure in the data sampling procedure (c.f. Theorem B.1). We ensure that all these failure probabilities are at most $\delta/3$ so that the final failure probability is at most δ.

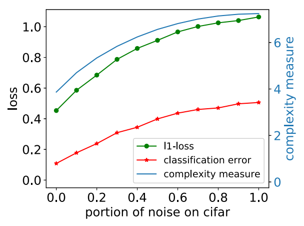
As a corollary of Theorem 5.1, for binary classification problems (i.e., labels are ± 1), we can show that (9) also bounds the *population classification error* of the learned classifier. See Appendix D for the proof.

Corollary 5.2. *Under the same assumptions as in Theorem 5.1 and additionally assuming that $y \in \{\pm 1\}$ for $(\mathbf{x}, y) \sim \mathcal{D}$, with probability at least $1-\delta$, the population classification error $L_{\mathcal{D}}^{01}(f_{\mathbf{W}(k),\mathbf{a}}) = \Pr_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathrm{sign}\left(f_{\mathbf{W}(k),\mathbf{a}}(\mathbf{x})\right) \neq y\right]$ is bounded as:*

$$L_{\mathcal{D}}^{01}(f_{\mathbf{W}(k),\mathbf{a}}) \leq \sqrt{\frac{2\mathbf{y}^\top\left(\mathbf{H}^\infty\right)^{-1}\mathbf{y}}{n}} + O\left(\sqrt{\frac{\log\frac{n}{\lambda_0\delta}}{n}}\right).$$



(a) MNIST Data.



(b) CIFAR Data.

Figure 2: Generalization error (ℓ_1 loss and classification error) v.s. our complexity measure when different portions of random labels are used. We apply GD on data from two classes of MNIST or CIFAR until convergence. Our complexity measure almost matches the trend of generalization error as the portion of random labels increases. Note that ℓ_1 loss is always an upper bound on the classification error.

Now we discuss our generalization bound. The dominating term in (9) is:

$$\sqrt{\frac{2\mathbf{y}^\top \left(\mathbf{H}^\infty\right)^{-1}\mathbf{y}}{n}}. \tag{10}$$

This can be viewed as a *complexity measure of data* that one can use to predict the test accuracy of the learned neural network. Our result has the following advantages: (i) our complexity measure (10) can be directly computed given data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, without the need of training a neural network or assuming a ground-truth model; (ii) our bound is completely independent of the network width m.

Evaluating our completixy measure (10). To illustrate that the complexity measure in (10) effectively determines test error, in Figure 2 we compare this complexity measure versus the test error with true labels and random labels (and mixture of true and random labels). Random and true labels have significantly different complexity measures, and as the portion of random labels increases, our complexity measure also increases. See Appendix A for implementation details.

5.1. Proof Sketch of Theorem 5.1

The main ingredients in the proof of Theorem 5.1 are Lemmas 5.3 and 5.4. We defer the proofs of these lemmas as well as the full proof of Theorem 5.1 to Appendix D.

Our proof is based on a careful characterization of the trajectory of $\{\mathbf{W}(k)\}_{k=0}^\infty$ during GD. In particular, we bound its *distance to initialization* as follows:

Lemma 5.3. *Suppose* $m \geq \kappa^{-2}\text{poly}\left(n, \lambda_0^{-1}, \delta^{-1}\right)$ *and* $\eta = O\left(\frac{\lambda_0}{n^2}\right)$. *Then with probability at least* $1-\delta$ *over the random initialization, we have for all* $k \geq 0$*:*

- $\left\|\mathbf{w}_r(k) - \mathbf{w}_r(0)\right\|_2 = O\left(\frac{n}{\sqrt{m}\lambda_0\sqrt{\delta}}\right)$ $(\forall r \in [m])$, *and*
- $\left\|\mathbf{W}(k) - \mathbf{W}(0)\right\|_F \leq \sqrt{\mathbf{y}^\top \left(\mathbf{H}^\infty\right)^{-1}\mathbf{y}} + O\left(\frac{n\kappa}{\lambda_0\delta}\right) + \frac{\text{poly}\left(n, \lambda_0^{-1}, \delta^{-1}\right)}{m^{1/4}\kappa^{1/2}}$.

The bound on the movement of each $\mathbf{w}_r$ was proved in (Du et al., 2018c). Our main contribution is the bound on $\left\|\mathbf{W}(k) - \mathbf{W}(0)\right\|_F$ which corresponds to the *total movement of all neurons*. The main idea is to couple the trajectory of $\{\mathbf{W}(k)\}_{k=0}^\infty$ with another simpler trajectory $\left\{\widetilde{\mathbf{W}}(k)\right\}_{k=0}^\infty$ defined as:

$$\begin{aligned}\widetilde{\mathbf{W}}(0) &= \mathbf{0}, \\ \text{vec}\left(\widetilde{\mathbf{W}}(k+1)\right) &= \text{vec}\left(\widetilde{\mathbf{W}}(k)\right) \\ &\quad - \eta\mathbf{Z}(0)\left(\mathbf{Z}(0)^\top \text{vec}\left(\widetilde{\mathbf{W}}(k)\right) - \mathbf{y}\right).\end{aligned} \tag{11}$$

We prove $\left\|\widetilde{\mathbf{W}}(\infty) - \widetilde{\mathbf{W}}(0)\right\|_F = \sqrt{\mathbf{y}^\top \mathbf{H}(0)^{-1}\mathbf{y}}$ in Section 5.2.[4] The actually proof of Lemma 5.3 is essentially a perturbed version of this.

Lemma 5.3 implies that the learned function $f_{\mathbf{W}(k),\mathbf{a}}$ from GD is in a *restricted* class of neural nets whose weights are close to initialization $\mathbf{W}(0)$. The following lemma bounds the Rademacher complexity of this function class:

Lemma 5.4. *Given* $R > 0$, *with probability at least* $1-\delta$ *over the random initialization* $(\mathbf{W}(0), \mathbf{a})$, *simultaneously for every* $B > 0$, *the following function class*

$$\begin{aligned}\mathcal{F}_{R,B}^{\mathbf{W}(0),\mathbf{a}} = \{f_{\mathbf{W},\mathbf{a}} : \left\|\mathbf{w}_r - \mathbf{w}_r(0)\right\|_2 \leq R\, (\forall r \in [m]), \\ \left\|\mathbf{W} - \mathbf{W}(0)\right\|_F \leq B\}\end{aligned}$$

has empirical Rademacher complexity bounded as:

$$\begin{aligned}&\mathcal{R}_S\left(\mathcal{F}_{R,B}^{\mathbf{W}(0),\mathbf{a}}\right) = \frac{1}{n}\mathbb{E}_{\boldsymbol{\varepsilon}\in\{\pm1\}^n}\left[\sup_{f\in\mathcal{F}_{R,B}^{\mathbf{W}(0),\mathbf{a}}} \sum_{i=1}^n \varepsilon_i f(\mathbf{x}_i)\right] \\ &\leq \frac{B}{\sqrt{2n}}\left(1 + \left(\frac{2\log\frac{2}{\delta}}{m}\right)^{1/4}\right) + \frac{2R^2\sqrt{m}}{\kappa} + R\sqrt{2\log\frac{2}{\delta}}.\end{aligned}$$

Finally, combining Lemmas 5.3 and 5.4, we are able to conclude that the neural network found by GD belongs to a function class with Rademacher complexity at most $\sqrt{\mathbf{y}^\top(\mathbf{H}^\infty)^{-1}\mathbf{y}/(2n)}$ (plus negligible errors). This gives us the generalization bound in Theorem 5.1 using the theory of Rademacher complexity (Appendix B).

5.2. Analysis of the Auxiliary Sequence $\left\{\widetilde{\mathbf{W}}(k)\right\}_{k=0}^\infty$

Now we give a proof of $\left\|\widetilde{\mathbf{W}}(\infty) - \widetilde{\mathbf{W}}(0)\right\|_F = \sqrt{\mathbf{y}^\top \mathbf{H}(0)^{-1}\mathbf{y}}$ as an illustration for the proof of Lemma 5.3.

[4] Note that we have $\mathbf{H}(0) \approx \mathbf{H}^\infty$ from standard concentration. See Lemma C.3.

Define $\mathbf{v}(k) = \mathbf{Z}(0)^\top \text{vec}\left(\widetilde{\mathbf{W}}(k)\right) \in \mathbb{R}^n$. Then from (11) we have $\mathbf{v}(0) = \mathbf{0}$ and $\mathbf{v}(k+1) = \mathbf{v}(k) - \eta\mathbf{H}(0)(\mathbf{v}(k) - \mathbf{y})$, yielding $\mathbf{v}(k) - \mathbf{y} = -(\mathbf{I} - \eta\mathbf{H}(0))^k\mathbf{y}$. Plugging this back to (11) we get $\text{vec}\left(\widetilde{\mathbf{W}}(k+1)\right) - \text{vec}\left(\widetilde{\mathbf{W}}(k)\right) = \eta\mathbf{Z}(0)(\mathbf{I} - \eta\mathbf{H}(0))^k\mathbf{y}$. Then taking a sum over $k = 0, 1, \ldots$ we have

$$\begin{aligned}\text{vec}\left(\widetilde{\mathbf{W}}(\infty)\right) - \text{vec}\left(\widetilde{\mathbf{W}}(0)\right) &= \sum_{k=0}^{\infty} \eta\mathbf{Z}(0)(\mathbf{I} - \eta\mathbf{H}(0))^k\mathbf{y} \\ &= \mathbf{Z}(0)\mathbf{H}(0)^{-1}\mathbf{y}.\end{aligned}$$

The desired result thus follows:

$$\begin{aligned}\left\|\widetilde{\mathbf{W}}(\infty) - \widetilde{\mathbf{W}}(0)\right\|_F^2 &= \mathbf{y}^\top\mathbf{H}(0)^{-1}\mathbf{Z}(0)^\top\mathbf{Z}(0)\mathbf{H}(0)^{-1}\mathbf{y} \\ &= \mathbf{y}^\top\mathbf{H}(0)^{-1}\mathbf{y}.\end{aligned}$$

6. Provable Learning using Two-Layer ReLU Neural Networks

Theorem 5.1 determines that $\sqrt{\frac{2\mathbf{y}^\top(\mathbf{H}^\infty)^{-1}\mathbf{y}}{n}}$ controls the generalization error. In this section, we study what functions can be provably learned in this setting. We assume the data satisfy $y_i = g(\mathbf{x}_i)$ for some underlying function $g : \mathbb{R}^d \to \mathbb{R}$. A simple observation is that if we can prove

$$\mathbf{y}^\top(\mathbf{H}^\infty)^{-1}\mathbf{y} \le M_g$$

for some quantity M_g that is *independent* of the number of samples n, then Theorem 5.1 implies we can provably learn the function g on the underlying data distribution using $O\left(\frac{M_g + \log(1/\delta)}{\epsilon^2}\right)$ samples. The following theorem shows that this is indeed the case for a broad class of functions.

Theorem 6.1. *Suppose we have*

$$y_i = g(\mathbf{x}_i) = \alpha\left(\boldsymbol{\beta}^\top\mathbf{x}_i\right)^p, \quad \forall i \in [n],$$

where $p = 1$ or $p = 2l$ ($l \in \mathbb{N}_+$), $\boldsymbol{\beta} \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}$. Then we have

$$\sqrt{\mathbf{y}^\top(\mathbf{H}^\infty)^{-1}\mathbf{y}} \le 3p\left|\alpha\right| \cdot \left\|\boldsymbol{\beta}\right\|_2^p.$$

The proof of Theorem 6.1 is given in Appendix E.

Notice that for two label vectors $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$, we have

$$\begin{aligned}&\sqrt{(\mathbf{y}^{(1)} + \mathbf{y}^{(2)})^\top(\mathbf{H}^\infty)^{-1}\left(\mathbf{y}^{(1)} + \mathbf{y}^{(2)}\right)} \\ \le &\sqrt{(\mathbf{y}^{(1)})^\top(\mathbf{H}^\infty)^{-1}\mathbf{y}^{(1)}} + \sqrt{(\mathbf{y}^{(2)})^\top(\mathbf{H}^\infty)^{-1}\mathbf{y}^{(2)}}.\end{aligned}$$

This implies that *the sum of learnable functions is also learnable*. Therefore, the following is a direct corollary of Theorem 6.1:

Corollary 6.2. *Suppose we have*

$$y_i = g(\mathbf{x}_i) = \sum_j \alpha_j\left(\boldsymbol{\beta}_j^\top\mathbf{x}_i\right)^{p_j}, \quad \forall i \in [n], \tag{12}$$

where for each j, $p_j \in \{1, 2, 4, 6, 8, \ldots\}$, $\boldsymbol{\beta}_j \in \mathbb{R}^d$ and $\alpha_j \in \mathbb{R}$. Then we have

$$\sqrt{\mathbf{y}^\top(\mathbf{H}^\infty)^{-1}\mathbf{y}} \le 3\sum_j p_j\left|\alpha_j\right| \cdot \left\|\boldsymbol{\beta}_j\right\|_2^{p_j}. \tag{13}$$

Corollary 6.2 shows that overparameterized two-layer ReLU network can learn any function of the form (12) for which (13) is bounded. One can view (12) as two-layer neural networks with polynomial activation $\phi(z) = z^p$, where $\{\boldsymbol{\beta}_j\}$ are weights in the first layer and $\{\alpha_j\}$ are the second layer. Below we give some specific examples.

Example 6.1 (Linear functions)**.** *For $g(\mathbf{x}) = \boldsymbol{\beta}^\top\mathbf{x}$, we have $M_g = O(\|\boldsymbol{\beta}\|_2^2)$.*

Example 6.2 (Quadratic functions)**.** *For $g(\mathbf{x}) = \mathbf{x}^\top\mathbf{A}\mathbf{x}$ where $\mathbf{A} \in \mathbb{R}^{d\times d}$ is symmetric, we can write down the eigen-decomposition $\mathbf{A} = \sum_{j=1}^d \alpha_j\boldsymbol{\beta}_j\boldsymbol{\beta}_j^\top$. Then we have $g(\mathbf{x}) = \sum_{j=1}^d \alpha_j(\boldsymbol{\beta}_j^\top\mathbf{x})^2$, so $M_g = O\left(\sum_{i=1}^d |\alpha_j|\right) = O(\|\mathbf{A}\|_*)$.*[5] *This is also the class of two-layer neural networks with quadratic activation.*

Example 6.3 (Cosine activation)**.** *Suppose $g(\mathbf{x}) = \cos(\boldsymbol{\beta}^\top\mathbf{x}) - 1$ for some $\boldsymbol{\beta} \in \mathbb{R}^d$. Using Taylor series we know $g(\mathbf{x}) = \sum_{j=1}^\infty \frac{(-1)^j(\boldsymbol{\beta}^\top\mathbf{x})^{2j}}{(2j)!}$. Thus we have $M_g = O\left(\sum_{j=1}^\infty \frac{j}{(2j)!}\|\boldsymbol{\beta}\|_2^{2j}\right) = O\left(\|\boldsymbol{\beta}\|_2 \cdot \sinh(\|\boldsymbol{\beta}\|_2)\right)$.*

Finally, we note that our "smoothness" requirement (13) is weaker than that in (Allen-Zhu et al., 2018a), as illustrated in the following example.

Example 6.4 (A not-so-smooth function)**.** *Suppose $g(\mathbf{x}) = \phi(\boldsymbol{\beta}^\top\mathbf{x})$, where $\phi(z) = z \cdot \arctan(\frac{z}{2})$ and $\|\boldsymbol{\beta}\|_2 \le 1$. We have $g(\mathbf{x}) = \sum_{j=1}^\infty \frac{(-1)^{j-1}2^{1-2j}}{2j-1}\left(\boldsymbol{\beta}^\top\mathbf{x}\right)^{2j}$ since $\left|\boldsymbol{\beta}^\top\mathbf{x}\right| \le 1$. Thus $M_g = O\left(\sum_{j=1}^\infty \frac{j\cdot 2^{1-2j}}{2j-1}\|\boldsymbol{\beta}\|_2^{2j}\right) \le O\left(\sum_{j=1}^\infty 2^{1-2j}\|\boldsymbol{\beta}\|^{2j}\right) = O\left(\|\boldsymbol{\beta}\|_2^2\right)$, so our result implies that this function is learnable by 2-layer ReLU nets.*

However, Allen-Zhu et al. (2018a)'s generalization theorem would require $\sum_{j=1}^\infty \frac{\left(C\sqrt{\log(1/\epsilon)}\right)^{2j}2^{1-2j}}{2j-1}$ to be bounded, where C is a large constant and ϵ is the target generalization error. This is clearly not satisfied.

7. Conclusion

This paper shows how to give a fine-grained analysis of the optimization trajectory and the generalization ability of overparameterized two-layer neural networks trained by gradient descent. We believe that our approach can also be useful in analyzing overparameterized deep neural networks and other machine learning models.

[5] $\|\mathbf{A}\|_*$ is the trace-norm of $\mathbf{A}$.

Acknowledgments

SA, WH and ZL acknowledge support from NSF, ONR, Simons Foundation, Schmidt Foundation, Mozilla Research, Amazon Research, DARPA and SRC. SSD acknowledges support from AFRL grant FA8750-17-2-0212 and DARPA D17AP00001. RW acknowledges support from ONR grant N00014-18-1-2562. Part of the work was done while SSD and RW were visiting the Simons Institute.

References

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. *arXiv preprint arXiv:1811.04918*, 2018a.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018b.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6241–6250, 2017a.

Bartlett, P. L., Harvey, N., Liaw, C., and Mehrabian, A. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *arXiv preprint arXiv:1703.02930*, 2017b.

Belkin, M., Ma, S., and Mandal, S. To understand deep learning we need to understand kernel learning. *arXiv preprint arXiv:1802.01396*, 2018.

Brutzkus, A. and Globerson, A. Globally optimal gradient descent for a ConvNet with gaussian inputs. *arXiv preprint arXiv:1702.07966*, 2017.

Chen, Y., Jin, C., and Yu, B. Stability and convergence trade-off of iterative optimization algorithms. *arXiv preprint arXiv:1804.01619*, 2018.

Chizat, L. and Bach, F. On the global convergence of gradient descent for over-parameterized models using optimal transport. *arXiv preprint arXiv:1805.09545*, 2018a.

Chizat, L. and Bach, F. A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*, 2018b.

Daniely, A. SGD learns the conjugate kernel class of the network. *arXiv preprint arXiv:1702.08503*, 2017.

Daniely, A., Frostig, R., and Singer, Y. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pp. 2253–2261, 2016.

Du, S. S. and Lee, J. D. On the power of over-parametrization in neural networks with quadratic activation. *arXiv preprint arXiv:1803.01206*, 2018.

Du, S. S., Lee, J. D., and Tian, Y. When is a convolutional filter easy to learn? *arXiv preprint arXiv:1709.06129*, 2017a.

Du, S. S., Lee, J. D., Tian, Y., Poczos, B., and Singh, A. Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima. *arXiv preprint arXiv:1712.00779*, 2017b.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018a.

Du, S. S., Wang, Y., Zhai, X., Balakrishnan, S., Salakhutdinov, R. R., and Singh, A. How many samples are needed to estimate a convolutional neural network? In *Advances in Neural Information Processing Systems*, pp. 371–381, 2018b.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018c.

Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Freeman, C. D. and Bruna, J. Topology and geometry of half-rectified network optimization. *arXiv preprint arXiv:1611.01540*, 2016.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle points − online stochastic gradient for tensor decomposition. In *Proceedings of The 28th Conference on Learning Theory*, pp. 797–842, 2015.

Golowich, N., Rakhlin, A., and Shamir, O. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry. *arXiv preprint arXiv:1802.08246*, 2018a.

Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. *arXiv preprint arXiv:1806.00468*, 2018b.

Haeffele, B. D. and Vidal, R. Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540*, 2015.

Hardt, M. and Ma, T. Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*, 2016.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

Imaizumi, M. and Fukumizu, K. Deep neural networks learn non-smooth functions effectively. *arXiv preprint arXiv:1802.04474*, 2018.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

Ji, Z. and Telgarsky, M. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1724–1732, 2017.

Kawaguchi, K. Deep learning without poor local minima. In *Advances In Neural Information Processing Systems*, pp. 586–594, 2016.

Konstantinos, P., Davies, M., and Vandergheynst, P. PAC-Bayesian margin bounds for convolutional neural networks-technical report. *arXiv preprint arXiv:1801.00171*, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. Gradient descent only converges to minimizers. In *Conference on Learning Theory*, pp. 1246–1257, 2016.

Li, X., Lu, J., Wang, Z., Haupt, J., and Zhao, T. On tighter generalization bound for deep neural networks: CNNs, ResNets, and beyond. *arXiv preprint arXiv:1806.05159*, 2018a.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. *arXiv preprint arXiv:1808.01204*, 2018.

Li, Y. and Yuan, Y. Convergence analysis of two-layer neural networks with ReLU activation. *arXiv preprint arXiv:1705.09886*, 2017.

Li, Y., Ma, T., and Zhang, H. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pp. 2–47, 2018b.

Ma, C., Wu, L., et al. A priori estimates of the generalization error for two-layer neural networks. *arXiv preprint arXiv:1810.06397*, 2018.

Mei, S., Montanari, A., and Nguyen, P.-M. A mean field view of the landscape of two-layers neural networks. *arXiv preprint arXiv:1804.06561*, 2018.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. Foundations of machine learning. *MIT Press*, 2012.

Mou, W., Wang, L., Zhai, X., and Zheng, K. Generalization bounds of SGLD for non-convex learning: Two theoretical viewpoints. *arXiv preprint arXiv:1707.05947*, 2017.

Neyshabur, B., Tomioka, R., and Srebro, N. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BygfghAcYX`.

Nguyen, Q. and Hein, M. The loss surface of deep and wide neural networks. *arXiv preprint arXiv:1704.08045*, 2017.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Rotskoff, G. M. and Vanden-Eijnden, E. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

Safran, I. and Shamir, O. Spurious local minima are common in two-layer relu neural networks. *arXiv preprint arXiv:1712.08968*, 2017.

Sirignano, J. and Spiliopoulos, K. Mean field analysis of neural networks. *arXiv preprint arXiv:1805.01053*, 2018.

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

Soltanolkotabi, M., Javanmard, A., and Lee, J. D. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 2018.

Soudry, D. and Carmon, Y. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19 (70), 2018.

Tian, Y. An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis. *arXiv preprint arXiv:1703.00560*, 2017.

Tsuchida, R., Roosta-Khorasani, F., and Gallagher, M. Invariance of weight distributions in rectified mlps. *arXiv preprint arXiv:1711.09090*, 2017.

Venturi, L., Bandeira, A., and Bruna, J. Neural networks with finite intrinsic dimension have no spurious valleys. *arXiv preprint arXiv:1802.06384*, 2018.

Wei, C., Lee, J. D., Liu, Q., and Ma, T. On the margin theory of feedforward neural networks. *arXiv preprint arXiv:1810.05369*, 2018.

Xie, B., Liang, Y., and Song, L. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics*, pp. 1216–1224, 2017.

Yun, C., Sra, S., and Jadbabaie, A. A critical view of global optimality in deep learning. *arXiv preprint arXiv:1802.03487*, 2018.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations (ICLR), 2017*, 2017.

Zhang, X., Yu, Y., Wang, L., and Gu, Q. Learning one-hidden-layer relu networks via gradient descent. *arXiv preprint arXiv:1806.07808*, 2018.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. Recovery guarantees for one-hidden-layer neural networks. *arXiv preprint arXiv:1706.03175*, 2017.

Zhou, W., Veitch, V., Austern, M., Adams, R. P., and Orbanz, P. Non-vacuous generalization bounds at the imagenet scale: a PAC-bayesian compression approach. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJgqqsAct7`.

Zhou, Y. and Liang, Y. Critical points of neural networks: Analytical forms and landscape properties. *arXiv preprint arXiv:1710.11205*, 2017.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Stochastic gradient descent optimizes over-parameterized deep ReLU networks. *arXiv preprint arXiv:1811.08888*, 2018.