Towards Understanding the Spectral Bias of Deep Learning

Yuan Cao[1*], **Zhiying Fang**[2*], **Yue Wu**[1*] and **Ding-Xuan Zhou**[2] and **Quanquan Gu**[1]

[1]Department of Computer Science, University of California, Los Angeles
[2]School of Data Science and Department of Mathematics, City University of Hong Kong
yuancao@cs.ucla.edu, zyfang4-c@my.cityu.edu.hk, ywu@cs.ucla.edu, mazhou@cityu.edu.hk, qgu@cs.ucla.edu

Abstract

An intriguing phenomenon observed during training neural networks is the spectral bias, which states that neural networks are biased towards learning less complex functions. The priority of learning functions with low complexity might be at the core of explaining the generalization ability of neural networks, and certain efforts have been made to provide a theoretical explanation for spectral bias. However, there is still no satisfying theoretical result justifying the underlying mechanism of spectral bias. In this paper, we give a comprehensive and rigorous explanation for spectral bias and relate it with the neural tangent kernel function proposed in recent work. We prove that the training process of neural networks can be decomposed along different directions defined by the eigenfunctions of the neural tangent kernel, where each direction has its own convergence rate and the rate is determined by the corresponding eigenvalue. We then provide a case study when the input data is uniformly distributed over the unit sphere, and show that lower degree spherical harmonics are easier to be learned by over-parameterized neural networks. Finally, we provide numerical experiments to demonstrate the correctness of our theory. Our experimental results also show that our theory can tolerate certain model misspecification in terms of the input data distribution.

1 Introduction

Over-parameterized neural networks have achieved great success in many applications. However, the success of deep learning has not been well understood. In order to understand neural network training, a recent work [Rahaman *et al.*, 2019] pointed out an intriguing phenomenon called *spectral bias*, which says that during training, neural networks tend to learn the components of lower complexity faster. Similar observation has also been pointed out in [Xu *et al.*, 2019; Xu *et al.*, 2020]. The concept of spectral bias is appealing because this may intuitively explain why over-parameterized neural networks can achieve a good generalization performance without overfitting. During training, the networks fit the low complexity components first and thus lie in the concept class of low complexity. Arguments like this may lead to rigorous guarantee for generalization.

Great efforts have been made in search of explanations about the spectral bias. [Rahaman *et al.*, 2019] evaluated the Fourier spectrum of ReLU networks and empirically showed that the lower frequencies are learned first; also lower frequencies are more robust to random perturbation. [Andoni *et al.*, 2014] showed that for a sufficiently wide two-layer network, gradient descent with respect to the second layer can learn any low degree bounded polynomial. [Xu, 2018] provided Fourier analysis to two-layer networks and showed similar empirical results on one-dimensional functions and real data. [Nakkiran *et al.*, 2019] used information theoretical approach to show that networks obtained by stochastic gradient descent can be explained by a linear classifier during early training. These studies provide certain explanations about why neural networks exhibit spectral bias in real tasks. But explanations in the theoretical aspect, if any, are to some extent limited. For example, Fourier analysis is usually done in the one-dimensional setting, and thus lacks generality.

Meanwhile, a recent line of work has taken a new approach to analyze neural networks based on the *neural tangent kernel* (NTK) [Jacot *et al.*, 2018]. In particular, they show that under certain over-parameterization condition, the neural network trained by gradient descent behaves similarly to the kernel regression predictor using the neural tangent kernel. For training a neural network with hidden layer width m and sample size n, recent optimization results on the training loss in the so-called "neural tangent kernel regime" can be roughly categorized into the following two families: (i) Without any assumption on the target function (the function used to generate the true labels based on the data input), if the network width $m \geqslant \mathrm{poly}(n, \lambda_{\min}^{-1})$, where $\lambda_{\min}$ is the smallest eigenvalue of the NTK Gram matrix, then square loss/cross-entropy loss can be optimized to zero [Du *et al.*, 2019b; Allen-Zhu *et al.*, 2019; Du *et al.*, 2019a; Zou *et al.*, 2019; Zou and Gu, 2019; Lee *et al.*, 2019]; and (ii) If the target function has bounded norm in the NTK-induced reproducing kernel Hilbert space (RKHS), then global convergence can be achieved with milder requirements on m [Arora *et al.*, 2019; Cao and Gu, 2019].

*Equal contribution

Inspired by these works mentioned above in the neural tangent kernel regime, in this paper we study the spectral bias of over-parameterized two-layer neural networks and its connection to the neural tangent kernel. We show that, given a training data set that is generated based on a target function, a fairly narrow network, although cannot fit the training data well due to its limited width, can still learn certain low-complexity components of the target function in the eigenspace corresponding to large eigenvalues of neural tangent kernel. As the width of the network increases, more high-frequency components of the target function can be learned with a slower convergence rate. As a special case, our result implies that when the input data follows uniform distribution on the unit sphere, polynomials of lower degrees can be learned by a narrower neural network at a faster rate. We also conduct experiments to corroborate the theory we establish.

Our contributions are as follows:

1. We prove a generic theorem for arbitrary data distributions, which states that under certain sample complexity and over-parameterization conditions, the convergence of the training error along different eigendirections of NTK relies on the corresponding eigenvalues. This theorem gives a more precise control on the regression residual than [Su and Yang, 2019], where the authors focused on the case when the labeling function is close to the subspace spanned by the first few eigenfunctions.
2. We establish a rigorous explanation for spectral bias based on the aforementioned theoretical results without any specific assumptions on the target function. We show that the error terms from different frequencies are controlled by the eigenvalues of the NTK, and the lower-frequency components can be learned with less training examples and narrower networks at a faster convergence rate.
3. For uniformly distributed input data, we further give convergence results of two-layer ReLU networks by characterizing the spectra of the neural tangent kernel. We show that in this setting the eigenvalues of neural tangent kernel are $\mu_k = \Omega(\max\{k^{-d-1}, d^{-k+1}\})$, $k \geqslant 0$, with corresponding eigenfunctions being the k-th order spherical harmonics. Our result is better than the bound $\Omega(k^{-d-1})$ derived in [Bietti and Mairal, 2019] when $d \gg k$, which is in a more practical setting.

1.1 Additional Related Work

A few theoretical results have been established towards understanding the spectra of neural tangent kernels. To name a few, [Bach, 2017] studied two-layer ReLU networks by relating it to kernel methods, and proposed a harmonic decomposition for the functions in the reproducing kernel Hilbert space which we utilize in our proof. Based on the technique in [Bach, 2017], [Bietti and Mairal, 2019] studied the eigenvalue decay of integrating operator defined by the neural tangent kernel on unit sphere by using spherical harmonics. [Vempala and Wilmes, 2019] calculated the eigenvalues of neural tangent kernel corresponding to two-layer neural networks with sigmoid activation function. [Basri *et al.*, 2019] established similar results as [Bietti and Mairal, 2019], but considered the case of training the first layer parameters of a two-layer networks with bias terms. [Yang and Salman, 2019] studied the the eigenvalues of integral operator with respect to the NTK on Boolean cube by Fourier analysis. Very recently, [Bordelon *et al.*, 2020] gave a spectral analysis on the generalization error of NTK-based kernel ridge regression. [Basri *et al.*, 2020] studied the convergence of full training residual with a focus on one-dimensional, non-uniformly distributed data. [Advani *et al.*, 2020] gave an average case analysis of the learning dynamics of large neural networks trained by gradient descent and studied the generalization of over-parameterized neural networks.

2 Preliminaries

In this section we introduce the basic problem setup including the neural network structure and the training algorithm, as well as some background on the neural tangent kernel proposed recently in [Jacot *et al.*, 2018].

Notation. We use lower case, lower case bold face, and upper case bold face letters to denote scalars, vectors and matrices respectively. For a vector $\mathbf{v} = (v_1, \ldots, v_d)^T \in \mathbb{R}^d$ and a number $1 \leqslant p < \infty$, we denote its $p-$norm by $\|\mathbf{v}\|_p = (\sum_{i=1}^d |v_i|^p)^{1/p}$. We also define infinity norm by $\|\mathbf{v}\|_\infty = \max_i |v_i|$. For a matrix $\mathbf{A} = (A_{i,j})_{m\times n}$, we use $\|\mathbf{A}\|_0$ to denote the number of non-zero entries of $\mathbf{A}$, and use $\|\mathbf{A}\|_F = (\sum_{i,j=1}^d A_{i,j}^2)^{1/2}$ to denote its Frobenius norm. Let $\|\mathbf{A}\|_p = \max_{\|\mathbf{v}\|_p \leqslant 1} \|\mathbf{A}\mathbf{v}\|_p$ for $p \geqslant 1$, and $\|\mathbf{A}\|_{\max} = \max_{i,j} |A_{i,j}|$. For two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m\times n}$, we define $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{Tr}(\mathbf{A}^\top \mathbf{B})$. We use $\mathbf{A} \succeq \mathbf{B}$ if $\mathbf{A} - \mathbf{B}$ is positive semi-definite. In addition, we define the asymptotic notations $\mathcal{O}(\cdot)$, $\widetilde{\mathcal{O}}(\cdot)$, $\Omega(\cdot)$ and $\widetilde{\Omega}(\cdot)$ as follows. Suppose that a_n and b_n be two sequences. We write $a_n = \mathcal{O}(b_n)$ if $\limsup_{n\to\infty} |a_n/b_n| < \infty$, and $a_n = \Omega(b_n)$ if $\liminf_{n\to\infty} |a_n/b_n| > 0$. We use $\widetilde{\mathcal{O}}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ to hide the logarithmic factors in $\mathcal{O}(\cdot)$ and $\Omega(\cdot)$.

Problem Setup. Here we introduce the basic problem setup. We consider two-layer fully connected neural networks of the form

$$f_{\mathbf{W}}(\mathbf{x}) = \sqrt{m} \cdot \mathbf{W}_2 \sigma(\mathbf{W}_1 \mathbf{x}),$$

where $\mathbf{W}_1 \in \mathbb{R}^{m\times(d+1)}$, $\mathbf{W}_2 \in \mathbb{R}^{1\times m}$ are[1] the first and second layer weight matrices respectively, and $\sigma(\cdot) = \max\{0, \cdot\}$ is the entry-wise ReLU activation function. The network is trained according to the square loss on n training examples $S = \{(\mathbf{x}_i, y_i) : i \in [n]\}$:

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

where θ is a small coefficient to control the effect of initialization, and the data inputs $\{\mathbf{x}_i\}_{i=1}^n$ are assumed to follow some unknown distribution τ on the unit sphere $\mathbb{S}^d \subseteq \mathbb{R}^{d+1}$. Without loss of generality, we also assume that $|y_i| \leqslant 1$.

We first randomly initialize the parameters of the network, and run gradient descent for both layers. We present our detailed neural network training algorithm in Algorithm 1.

[1] Here the input dimension is $d+1$ since in this paper we assume that all training data lie in the d-dimensional unit sphere $\mathbb{S}^d \subseteq \mathbb{R}^{d+1}$.

Algorithm 1 GD for DNNs starting at Gaussian initialization

Input: Number of iterations T, step size η.
Generate each entry of $\mathbf{W}_1^{(0)}$ and $\mathbf{W}_2^{(0)}$ from $N(0, 2/m)$ and $N(0, 1/m)$ respectively.
for $t = 0, 1, \ldots, T-1$ **do**
  Update $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \cdot \nabla_{\mathbf{W}} L_S(\mathbf{W}^{(t)})$.
end for
Output: $\mathbf{W}^{(T)}$.

The initialization scheme for $\mathbf{W}^{(0)}$ given in Algorithm 1 is known as He initialization [He *et al.*, 2015]. It is consistent with the initialization scheme used in [Cao and Gu, 2019].

Neural Tangent Kernel. Many attempts have been made to study the convergence of gradient descent assuming the width of the network is extremely large [Du *et al.*, 2019b; Li and Liang, 2018]. When the width of the network goes to infinity, with certain initialization on the model weights, the limit of inner product of network gradients defines a kernel function, namely the neural tangent kernel [Jacot *et al.*, 2018]. The neural tangent kernel is derived by linearizing the network around its random initialization, and is defined as

$$\kappa(\mathbf{x}, \mathbf{x}') = \lim_{m\to\infty} m^{-1} \langle \nabla_{\mathbf{W}} f_{\mathbf{W}^{(0)}}(\mathbf{x}), \nabla_{\mathbf{W}} f_{\mathbf{W}^{(0)}}(\mathbf{x}') \rangle.$$

For two-layer networks, by the definition of Gaussian initialization, we can obtain the following calculations of the neural tangent kernel by the law of large numbers:

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle \cdot \kappa_1(\mathbf{x}, \mathbf{x}') + 2 \cdot \kappa_2(\mathbf{x}, \mathbf{x}'), \tag{2.1}$$

where

$$\begin{aligned} \kappa_1(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{\mathbf{w}\sim N(\mathbf{0}, \mathbf{I})}[\sigma'(\langle \mathbf{w}, \mathbf{x} \rangle)\sigma'(\langle \mathbf{w}, \mathbf{x}' \rangle)], \\ \kappa_2(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{\mathbf{w}\sim N(\mathbf{0}, \mathbf{I})}[\sigma(\langle \mathbf{w}, \mathbf{x} \rangle)\sigma(\langle \mathbf{w}, \mathbf{x}' \rangle)]. \end{aligned} \tag{2.2}$$

Since we apply gradient descent to both layers, the neural tangent kernel is the sum of the two different kernel functions and clearly it can be reduced to one layer training setting. These two kernels are arc-cosine kernels of degree 0 and 1 [Cho and Saul, 2009], which are given as $\kappa_1(\mathbf{x}, \mathbf{x}') = \widehat{\kappa}_1(\langle \mathbf{x}, \mathbf{x}' \rangle (\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2))$, $\kappa_2(\mathbf{x}, \mathbf{x}') = \widehat{\kappa}_2(\langle \mathbf{x}, \mathbf{x}' \rangle / (\|\mathbf{x}\|_2 \|\mathbf{x}'\|_2))$, where

$$\begin{aligned} \widehat{\kappa}_1(t) &= \frac{1}{2\pi} \left(\pi - \arccos(t) \right), \\ \widehat{\kappa}_2(t) &= \frac{1}{2\pi} \left(t \cdot (\pi - \arccos(t)) + \sqrt{1 - t^2} \right). \end{aligned} \tag{2.3}$$

Integral Operator. The theory of integral operator with respect to kernel function has been well studied in literature [Smale and Zhou, 2007; Rosasco *et al.*, 2010] thus we only give a brief introduction here. Let $L_\tau^2(X)$ be the Hilbert space of square-integrable functions with respect to a Borel measure τ from $X \to \mathbb{R}$. For any continuous kernel function $\kappa : X \times X \to \mathbb{R}$ and τ we can define an integral operator L_κ on $L_\tau^2(X)$ by

$$L_\kappa(f)(\mathbf{x}) = \int_X \kappa(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\tau(\mathbf{y}), \quad \mathbf{x} \in X. \tag{2.4}$$

It has been pointed out in [Cho and Saul, 2009] that arc-cosine kernels are positive semi-definite. Thus the kernel function κ defined by (2.1) is positive semi-definite being a product and a sum of positive semi-definite kernels. Clearly this kernel is also continuous and symmetric, which implies that the neural tangent kernel κ is a Mercer kernel.

3 Main Results

In this section we present our main results. We first give a general result on the convergence rate of gradient descent along different eigendirections of neural tangent kernel, and then apply this result to give an explicit convergence rate for uniformly distributed data on the unit sphere.

3.1 Convergence Analysis of Gradient Descent

In this section we study the convergence of Algorithm 1. Instead of studying the standard convergence of loss function value, we provide a refined analysis on the speed of convergence along different directions defined by the eigenfunctions of L_κ. For simplicity, we focus on the setting where the data inputs have normalized lengths, i.e., $\mathbf{x} \in \mathbb{S}^d$.

Let $\{\lambda_i\}_{i\geqslant 1}$ with $\lambda_1 \geqslant \lambda_2 \geqslant \cdots$ be the strictly positive eigenvalues of L_κ, and $\phi_1(\cdot), \phi_2(\cdot), \ldots$ be the corresponding orthonormal eigenfunctions (i.e., $L_\kappa(\phi_i)(\mathbf{x}) = \lambda_i \phi_i(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{S}^d$). Set $\mathbf{v}_i = n^{-1/2}(\phi_i(\mathbf{x}_1), \ldots, \phi_i(\mathbf{x}_n))^\top$, $i = 1, 2, \ldots$. Note that L_κ may have eigenvalues with multiplicities larger than 1 and λ_i, $i \geqslant 1$ are not distinct. Therefore for any integer k, we define r_k as the sum of the multiplicities of the first k distinct eigenvalues of L_κ. Define $\mathbf{V}_{r_k} = (\mathbf{v}_1, \ldots, \mathbf{v}_{r_k})$. By definition, $\mathbf{v}_i$, $i \in [r_k]$ are rescaled restrictions of orthonormal functions in $L_\tau^2(\mathbb{S}^d)$ on the training examples. Therefore we can expect them to form a set of almost orthonormal bases in the vector space $\mathbb{R}^n$. The following lemma follows by standard concentration inequality.

Lemma 3.1. Suppose that $|\phi_i(\mathbf{x})| \leqslant M$ for all $\mathbf{x} \in \mathbb{S}^d$ and $i \in [r_k]$. For any $\delta > 0$, with probability at least $1 - \delta$,

$$\|\mathbf{V}_{r_k}^\top \mathbf{V}_{r_k} - \mathbf{I}\|_{\max} \leqslant CM^2 \sqrt{\log(r_k/\delta)/n},$$

where C is an absolute constant.

Denote $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\widehat{\mathbf{y}}^{(t)} = \theta \cdot (f_{\mathbf{W}^{(t)}}(\mathbf{x}_1), \ldots, f_{\mathbf{W}^{(t)}}(\mathbf{x}_n))^\top$ for $t = 0, \ldots, T$. Then Lemma 3.1 shows that the convergence rate of $\|\mathbf{V}_{r_k}^\top(\mathbf{y} - \widehat{\mathbf{y}}^{(t)})\|_2$ roughly represents the speed gradient descent learns the components of the target function corresponding to the first r_k eigenvalues. The following theorem gives the convergence guarantee of $\|\mathbf{V}_{r_k}^\top(\mathbf{y} - \widehat{\mathbf{y}}^{(t)})\|_2$.

Theorem 3.2. *Suppose $|\phi_j(\mathbf{x})| \leqslant M$ for $j \in [r_k]$ and $\mathbf{x} \in \mathbb{S}^d$. For any $\epsilon, \delta > 0$ and integer k, if $n \geqslant \widetilde{\Omega}(\epsilon^{-2} \cdot \max\{(\lambda_{r_k} - \lambda_{r_k+1})^{-2}, M^4 r_k^2\})$, $m \geqslant \widetilde{\Omega}(\mathrm{poly}(T, \lambda_{r_k}^{-1}, \epsilon^{-1}))$, then with probability at least $1 - \delta$, Algorithm 1 with $\eta = \widetilde{\mathcal{O}}(m^{-1}\theta^{-2})$, $\theta = \widetilde{\mathcal{O}}(\epsilon)$ satisfies*

$$\begin{aligned} &n^{-1/2} \cdot \|\mathbf{V}_{r_k}^\top(\mathbf{y} - \widehat{\mathbf{y}}^{(T)})\|_2 \\ &\quad \leqslant 2(1 - \lambda_{r_k})^T \cdot n^{-1/2} \cdot \|\mathbf{V}_{r_k}^\top \mathbf{y}\|_2 + \epsilon. \end{aligned}$$

Theorem 3.2 shows that the convergence rate of $\|\mathbf{V}_{r_k}^\top(\mathbf{y} - \widehat{\mathbf{y}}^{(t)})\|_2$ (or its normalized version, $\|\mathbf{V}_{r_k}^\top(\mathbf{y} - \widehat{\mathbf{y}}^{(t)})\|_2 / \|\mathbf{V}_{r_k}^\top \mathbf{y}\|_2$) is determined by the r_k-th eigenvalue λ_{r_k}.

This reveals the spectral bias of neural network training under the NTK regime. Specifically, when the network is wide enough and the sample size is large enough, gradient descent first learns the target function along the eigendirections of neural tangent kernel with larger eigenvalues, and then learns the rest components corresponding to smaller eigenvalues[2]. Moreover, by showing that learning the components corresponding to larger eigenvalues can be done with smaller sample size and narrower networks, our theory pushes the study of neural networks towards more practical settings. Therefore, Theorem 3.2 provides an explanation of the observations given in [Rahaman *et al.*, 2019] in the NTK regime.

It is also worth noting that Theorem 3.2 is *not* based on the common NTK-type assumption that the target function belongs to the NTK-induced RKHS [Arora *et al.*, 2019; Cao and Gu, 2019]. Instead, Theorem 3.2 works for arbitrary labeling, and is therefore rather general.

Comparison with existing results. The most relevant results to ours are [Arora *et al.*, 2019; Su and Yang, 2019; Bordelon *et al.*, 2020; Basri *et al.*, 2020]. Compared with [Arora *et al.*, 2019] which focuses on the setting where the network is wide enough to guarantee global convergence, our result works for narrower networks for which global convergence may not even be possible. Compared with [Su and Yang, 2019], the key difference is that while [Su and Yang, 2019] studied the full residual $\|\mathbf{y}-\widehat{\mathbf{y}}^{(T)}\|_2$ and required that the target function lies approximately in the eigenspace of large eigenvalues of the neural tangent kernel, our result in Theorem 3.2 works for arbitrary target function, and shows that even if the target function has very high frequency components, its components in the eigenspace of large eigenvalues can still be learned very efficiently by neural networks. More recently, [Bordelon *et al.*, 2020] studied the solution of NTK-based kernel ridge regression. Compared with their result, our analysis is directly on the practical neural network training procedure, and specifies the width requirement for a network to successfully learn a certain component of the target function. Another recent work by [Basri *et al.*, 2020] provides theoretical analysis for one-dimensional non-uniformly distributed data, and only studies the convergence of the full residual vector. In comparison, our results cover high-dimensional data, and provide a more detailed analysis on the convergence of different projections of the residual.

3.2 Spectral Analysis of NTK for Uniform Data

We now study the case when the data inputs are uniformly distributed over the unit sphere as an example where the eigendecompositon of NTK can be calculated. We present our results on the spectral analysis of neural tangent kernel in the form of a Mercer decomposition, which explicitly gives the eigenvalues and eigenfunctions of NTK.

Theorem 3.3. *For any $\mathbf{x},\mathbf{x}'\in\mathbb{S}^d$, the Mercer decomposition of the neural tangent kernel $\kappa:\mathbb{S}^d\times\mathbb{S}^d\to\mathbb{R}$ is*

$$\kappa\left(\mathbf{x},\mathbf{x}'\right)=\sum_{k=0}^{\infty}\mu_k\sum_{j=1}^{N(d,k)}Y_{k,j}\left(\mathbf{x}\right)Y_{k,j}\left(\mathbf{x}'\right),\qquad(3.1)$$

where $Y_{k,j}$ for $j=1,\cdots,N(d,k)$ are linearly independent spherical harmonics of degree k in $d+1$ variables with $N(d,k)=\frac{2k+d-1}{k}\binom{k+d-2}{d-1}$ and orders of μ_k are given by

$$\mu_0=\mu_1=\Omega(1),\ \mu_k=0, k=2j+1,$$
$$\mu_k=\Omega(\max\{d^{d+1}k^{k-1}(k+d)^{-k-d},d^{d+1}k^k(k+d)^{-k-d-1},$$
$$d^{d+2}k^{k-2}(k+d)^{-k-d-1}\}), k=2j,$$

where $j\in\mathbb{N}^+$. Specifically, we have $\mu_k=\Omega\left(k^{-d-1}\right)$ when $k\gg d$ and $\mu_k=\Omega\left(d^{-k+1}\right)$ when $d\gg k$, $k=2,4,6,\ldots$.

Remark 3.4. The μ_k's in Theorem 3.3 are the distinct eigenvalues of the integral operator L_κ on $L^2_{\tau_d}(\mathbb{S}^d)$ defined by

$$L_\kappa(f)(\mathbf{y})=\int_{\mathbb{S}^d}\kappa(\mathbf{x},\mathbf{y})f(\mathbf{x})d\tau_d(\mathbf{x}),\quad f\in L^2_{\tau_d}(\mathbb{S}^d),$$

where τ_d is the uniform probability measure on unit sphere $\mathbb{S}^d$. Therefore the eigenvalue λ_{r_k} in Theorem 3.2 is just μ_{k-1} given in Theorem 3.3 when τ_d is uniform distribution.

Remark 3.5. [Vempala and Wilmes, 2019] studied two-layer neural networks with sigmoid activation function, and proved that in order to achieve $\epsilon_0+\epsilon$ error, it requires $T=(d+1)^{\mathcal{O}(k)\log(\|f^*\|_2/\epsilon)}$ iterations and $m=(d+1)^{\mathcal{O}(k)\mathrm{poly}(\|f^*\|_2/\epsilon)}$ wide neural networks, where f^* is the target function, and ϵ_0 is certain function approximation error. Another highly related work is [Bietti and Mairal, 2019], which gives $\mu_k=\Omega(k^{-d-1})$. The order of eigenvalues we present appears as $\mu_k=\Omega(\max(k^{-d-1},d^{-k+1}))$. This is better when $d\gg k$, which is closer to the practical setting.

3.3 Convergence for Uniformly Distributed Data

In this subsection, we apply the result in the previous subsection and give explicit convergence rate for uniformly distributed data on the unit sphere. The following two corollaries are based on the calculation of the spectral decomposition of the NTK for uniform data in Theorem 3.1.

Corollary 3.6. Suppose that $k\gg d$, and the sample $\{\mathbf{x}_i\}_{i=1}^n$ follows the uniform distribution τ_d on the unit sphere $\mathbb{S}^d$. For any $\epsilon,\delta>0$ and integer k, if $n\geqslant\widetilde{\Omega}(\epsilon^{-2}\cdot\max\{k^{2d+2},k^{2d-2}r_k^2\})$, $m\geqslant\widetilde{\Omega}(\mathrm{poly}(T,k^{d+1},\epsilon^{-1}))$, then with probability at least $1-\delta$, Algorithm 1 with $\eta=\widetilde{\mathcal{O}}((m\theta^2)^{-1})$, $\theta=\widetilde{\mathcal{O}}(\epsilon)$ satisfies

$$n^{-1/2}\cdot\|\mathbf{V}_{r_k}^\top(\mathbf{y}-\widehat{\mathbf{y}}^{(T)})\|_2$$
$$\leqslant 2\left(1-\Omega\left(k^{-d-1}\right)\right)^T\cdot n^{-1/2}\cdot\|\mathbf{V}_{r_k}^\top\mathbf{y}\|_2+\epsilon,$$

where $r_k=\sum_{k'=0}^{k-1}N(d,k')$ and $\mathbf{V}_{r_k}=(n^{-1/2}\phi_j(\mathbf{x}_i))_{n\times r_k}$ with $\phi_1,\ldots,\phi_{r_k}$ being a set of orthonormal spherical harmonics of degrees up to $k-1$.

Corollary 3.7. Suppose that $d\gg k$, and the sample $\{\mathbf{x}_i\}_{i=1}^n$ follows the uniform distribution τ_d on the unit sphere $\mathbb{S}^d$. For any $\epsilon,\delta>0$ and integer k, if $n\geqslant\widetilde{\Omega}(\epsilon^{-2}d^{2k-2}r_k^2)$, $m\geqslant\widetilde{\Omega}(\mathrm{poly}(T,d^{k-2},\epsilon^{-1}))$, then with probability at least $1-\delta$, Algorithm 1 with $\eta=\widetilde{\mathcal{O}}((m\theta^2)^{-1})$, $\theta=\widetilde{\mathcal{O}}(\epsilon)$ satisfies

$$n^{-1/2}\cdot\|\mathbf{V}_{r_k}^\top(\mathbf{y}-\widehat{\mathbf{y}}^{(T)})\|_2$$
$$\leqslant 2\left(1-\Omega\left(d^{-k+2}\right)\right)^T\cdot n^{-1/2}\cdot\|\mathbf{V}_{r_k}^\top\mathbf{y}\|_2+\epsilon,$$

[2]Note that by the definition we have $\lambda_{r_k}<1$ for all k. More details are given in Section 3.2 and th proofs in the appendix

where $r_k = \sum_{k'=0}^{k-1} N(d,k')$ and $\mathbf{V}_{r_k} = (n^{-1/2}\phi_j(\mathbf{x}_i))_{n\times r_k}$ with $\phi_1, \ldots, \phi_{r_k}$ being a set of orthonormal spherical harmonics of degrees up to $k-1$.

Corollaries 3.6 and 3.7 further illustrate the spectral bias of neural networks by providing exact calculations of λ_{r_k}, $\mathbf{V}_{r_k}$ and M in Theorem 3.2. They show that if the input distribution is uniform over unit sphere, then spherical harmonics with lower degrees are learned first by wide neural networks.

4 Experiments

In this section we present experimental results to verify our theory. Across all tasks, we train a two-layer neural networks with 4096 hidden neurons and initialize it exactly as defined in the problem setup. The optimization method is vanilla gradient descent, and the training sample size is 1000.

Learning Polynomials. We first experimentally verify our theoretical results by learning linear combinations of spherical harmonics with data inputs uniformly distributed over unit sphere. Define

$$f^*(\mathbf{x}) = a_1 P_1(\langle \boldsymbol{\zeta}_1, \mathbf{x}\rangle) + a_2 P_2(\langle \boldsymbol{\zeta}_2, \mathbf{x}\rangle) + a_4 P_4(\langle \boldsymbol{\zeta}_4, \mathbf{x}\rangle),$$

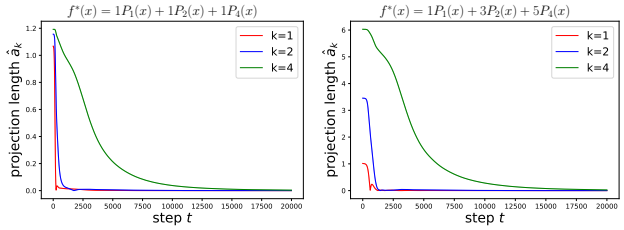
where the $P_k(t)$ is the Gegenbauer polynomial, and $\boldsymbol{\zeta}_k$, $k = 1, 2, 4$ are fixed vectors that are independently generated from uniform distribution on unit sphere in $\mathbb{R}^{10}$. Note that according to the addition formula $\sum_{j=1}^{N(d,k)} Y_{k,j}(\mathbf{x}) Y_{k,j}(\mathbf{y}) = N(d,k) P_k(\langle \mathbf{x}, \mathbf{y}\rangle)$, every normalized Gegenbauer polynomial is a spherical harmonic, so $f^*(\mathbf{x})$ is a linear combination of spherical harmonics of order 1,2 and 4. The higher odd-order Gegenbauer polynomials are omitted because the spectral analysis showed that $\mu_k = 0$ for $k = 3, 5, 7 \ldots$.

Following our theoretical analysis, we denote $\mathbf{v}_k = n^{-1/2}(P_k(\mathbf{x}_1), \ldots, P_k(\mathbf{x}_n))$. By Lemma 3.2 $\mathbf{v}_k$'s are almost orthonormal. So we define the (approximate) projection length of residual $\mathbf{r}^{(t)}$ onto $\mathbf{v}_k$ at step t as $\widehat{a}_k = |\mathbf{v}_k^\top \mathbf{r}^{(t)}|$, where $\mathbf{r}^{(t)} = (f^*(\mathbf{x}_1) - \theta f_{\mathbf{W}^{(t)}}(\mathbf{x}_1), \ldots, f^*(\mathbf{x}_n) - \theta f_{\mathbf{W}^{(t)}}(\mathbf{x}_n))$ and $f_{\mathbf{W}^{(t)}}(\mathbf{x})$ is the neural network function.

The results are shown in Figure 1. It can be seen that the residual at the lowest frequency ($k = 1$) converges to zero first and then the second lowest ($k = 2$). The highest frequency component is the last one to converge. Following the setting of [Rahaman *et al.*, 2019] we assign high frequencies a larger scale in Figure 1 (b), expecting that larger scale will introduce a better descending speed. Still, the low frequencies are learned first.

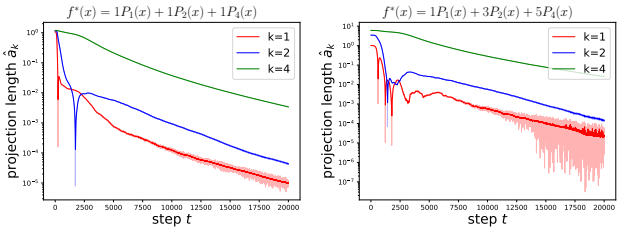
Note that $\widehat{a}_k$ is the projection length onto an approximate vector. In the function space, we can also project the residual function $r(\mathbf{x}) = f^*(\mathbf{x}) - \theta f_{\mathbf{W}^{(t)}}(\mathbf{x})$ onto the orthonormal Gegenbauer functions $P_k(\mathbf{x})$. Replacing the training data with randomly sampled data points $\mathbf{x}_i$ can lead to a random estimate of the projection length in function space. Experiments in this setting can be found in the appendix.

We verify the linear convergence rate proved in Theorem 3.2. Figure 2 presents the convergence curve in logarithmic scale. We can see from Figure 2 that the convergence of different projection length is close to linear convergence, which is well-aligned with our theoretical analysis. Note that we performed a moving average of range 20 on these curves to avoid the heavy oscillation especially at late stage.



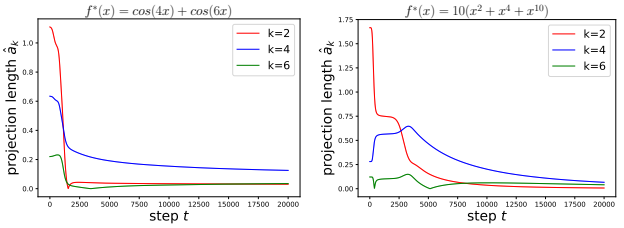
(a) same scale (b) different scale

Figure 1: Convergence curve of projection lengths. (a) shows the curve when the target function have the same scale for different components. (b) shows the curve when the higher-order components have larger scale.



(a) same scale (b) different scale

Figure 2: Log-scale convergence curve for projection lengths. (a) shows the curve when the target function have the same scale for different components. (b) shows the curve when the higher-order components have larger scale.



(a) cosine function (b) even polynomial

Figure 3: Convergence curve for different components. (a) shows the curve of a trigonometric function. (b) shows the curve of a polynomial with even degrees.

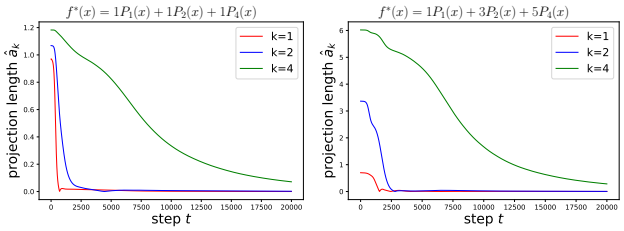
Learning Functions of Simple Forms. Apart from the synthesized low frequency function, we also show the dynamics of more general functions' projection to $P_k(x)$. These functions, though in a simple form, have non-zero high-frequency components. In this subsection we show our results still apply when all frequencies exist in the target functions, which are given by $f^*(\mathbf{x}) = \sum_i \cos(a_i \langle \boldsymbol{\zeta}, \mathbf{x}\rangle)$ and $f^*(\mathbf{x}) = \sum_i \langle \boldsymbol{\zeta}, \mathbf{x}\rangle^{p_i}$, where $\boldsymbol{\zeta}$ is a fixed unit vector.

Figure 3 verifies the result of Theorem 3.2 with more general target functions, and backs up our claim that Theorem 3.2 does not make any assumptions on the target function. Notice

that in the early training stage, not all the curves monotonically descend. This is due to the unseen components' influence on the gradient. Again, as the training proceeds, the residual projections converge at the predicted rates.

Non-uniform Input Data Distributions. In this subsection, we provide experimental results for non-uniformly distributed input data. Note that the eigendecomposition of NTK for general multi-dimensional input distributions do not necessarily have good analytic forms. Therefore, here we treat the non-uniform distribution of the input data as model misspecification, and test whether residual projections onto spherical harmonics of different degrees can still exhibit various convergence speed. We consider three examples of non-uniform distributions: (i) piece-wise uniform distribution, (ii) normalized non-isotropic Gaussian, and (iii) normalized Gaussian mixture.

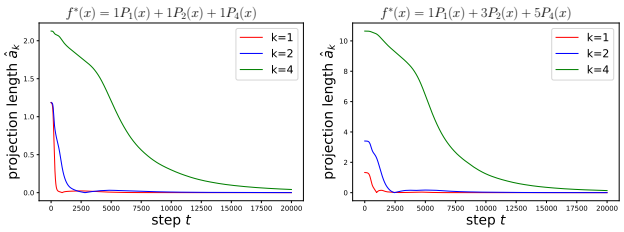
Piece-wise uniform distribution. We divide the unit sphere into two semi-spheres along a randomly drawn direction $\boldsymbol{\zeta}_0$. A data input is then generated as follows: with probability $1/4$, draw the input uniformly over the first unit sphere; with probability $3/4$, draw the input uniformly over the second unit sphere. The results are shown in Figure 4.



(a) same scale (b) different scale

Figure 4: Convergence curve of projection lengths for the piece-wise uniform distribution example.

Normalized non-isotropic Gaussian. We generate the data by first generating non-zero mean, non-isotropic Gaussian vectors, and then normalize them to unit length. The Gaussian mean vector is generated elementwise from $\mathrm{Unif}([-1, 1])$; the covariance matrix is generated as $\boldsymbol{\Sigma} = \mathbf{A}^\top \mathbf{A}$, where $\mathbf{A} \in \mathbb{R}_{d\times D}(d = 10, D = 20)$ has i.i.d. standard Gaussian entries. The results are shown in Figure 5.

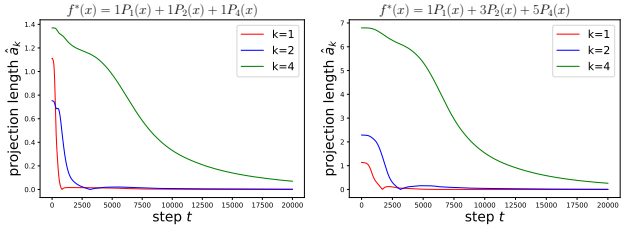


(a) same scale (b) different scale

Figure 5: Convergence curve for projection lengths for the normalized non-isotropic Gaussian example.

Normalized Gaussian mixture. Here the inputs are drawn from a mixture of 3 non-isotropic Gaussian distributions described in the second example. The results are in Figure 6.

Figures 4, 5, 6 show that the residual components corresponding to lower order polynomials are still learned relatively faster, even for the non-uniform distributions described above, where spherical harmonics are no longer exactly the eigenfunctions of NTK. This suggests that our theoretical results can tolerate certain level of model misspecification, and therefore the spectral bias characterized in Theorem 3.2 and Corollaries 3.6, 3.7 holds in a variety of problem settings.



(a) same scale (b) different scale

Figure 6: Convergence curve of projection lengths for the normalized Gaussian mixture example.

5 Conclusion and Future Work

This paper gives theoretical justification for spectral bias through a detailed analysis of the convergence behavior of two-layer ReLU networks. We show that the convergence of gradient descent in different directions depends on the corresponding eigenvalues of the neural tangent kernel, and give the exact order of convergence rate on different directions. We also conduct experiments on synthetic data to support our theoretical result.

Our current analysis of spectral bias is mostly in the NTK regime. Extending our analysis and results to the general setting beyond NTK regime is an important future work direction. Studying the spectral bias phenomenon for the training of deep neural networks with regularization [Hu *et al.*, 2021] is another interesting future direction.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments. YC and QG are partially supported by the National Science Foundation IIS-2008981 and Salesforce Deep Learning Research Award. DZ is supported partially by the Research Grants Council of Hong Kong [Project # CityU 11307319], Hong Kong Institute for Data Science, National Science Foundation of China [Project No. 12061160462], and an InnoHK project of the Hong Kong Innovation and Technology Commission.

References

[Advani *et al.*, 2020] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 2020.

[Allen-Zhu *et al.*, 2019] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019.

[Andoni *et al.*, 2014] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning polynomials with neural networks. In *ICML*, 2014.

[Arora *et al.*, 2019] Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *ICML*, 2019.

[Bach, 2017] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017.

[Basri *et al.*, 2019] Ronen Basri, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *NeurIPS*, 2019.

[Basri *et al.*, 2020] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. *arXiv preprint arXiv:2003.04560*, 2020.

[Bietti and Mairal, 2019] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. In *NeurIPS*, 2019.

[Bordelon *et al.*, 2020] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. *arXiv preprint arXiv:2002.02561*, 2020.

[Cao and Gu, 2019] Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *NeurIPS*, 2019.

[Cho and Saul, 2009] Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *NeurIPS*, 2009.

[Du *et al.*, 2019a] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *ICML*, 2019.

[Du *et al.*, 2019b] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *ICML*, 2019.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[Hu *et al.*, 2021] Tianyang Hu, Wenjia Wang, Cong Lin, and Guang Cheng. Regularization matters: A nonparametric perspective on overparametrized neural network. In *International Conference on Artificial Intelligence and Statistics*, pages 829–837. PMLR, 2021.

[Jacot *et al.*, 2018] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

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

[Li and Liang, 2018] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *NeurIPS*, 2018.

[Nakkiran *et al.*, 2019] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. In *NeurIPS*, 2019.

[Rahaman *et al.*, 2019] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *ICML*, 2019.

[Rosasco *et al.*, 2010] Lorenzo Rosasco, Mikhail Belkin, and Ernesto De Vito. On learning with integral operators. *Journal of Machine Learning Research*, 2010.

[Smale and Zhou, 2007] Steve Smale and Ding-Xuan Zhou. Learning theory estimates via integral operators and their approximations. *Constructive approximation*, 2007.

[Su and Yang, 2019] Lili Su and Pengkun Yang. On learning over-parameterized neural networks: A functional approximation prospective. In *NeurIPS*, 2019.

[Vempala and Wilmes, 2019] Santosh Vempala and John Wilmes. Gradient descent for one-hidden-layer neural networks: Polynomial convergence and sq lower bounds. In *Conference on Learning Theory*, 2019.

[Xu *et al.*, 2019] Zhi-Qin John Xu, Yaoyu Zhang, and Yanyang Xiao. Training behavior of deep neural network in frequency domain. In *ICONIP*, 2019.

[Xu *et al.*, 2020] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *Communications in Computational Physics*, 2020.

[Xu, 2018] Zhi-qin John Xu. Understanding training and generalization in deep learning by fourier analysis. *arXiv preprint arXiv:1808.04295*, 2018.

[Yang and Salman, 2019] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. *arXiv preprint arXiv:1907.10599*, 2019.

[Zou and Gu, 2019] Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. In *NeurIPS*, 2019.

[Zou *et al.*, 2019] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*, 2019.