CBMM Memo No. 076 **February 20, 2018**

An analysis of training and generalization errors in shallow and deep networks

H. N. Mhaskar and T. Poggio

Abstract

An open problem around deep networks is the apparent absence of over-fitting despite large over-parametrization which allows perfect fitting of the training data. In this paper, we explain this phenomenon when each unit evaluates a trigonometric polynomial. It is well understood in the theory of function approximation that approximation by trigonometric polynomials is a “role model” for many other processes of approximation that have inspired many theoretical constructions also in the context of approximation by neural and RBF networks. In this paper, we argue that the maximum loss functional is necessary to measure the generalization error. We give estimates on exactly how many parameters ensure both zero training error as well as a good generalization error, and how much error to expect at which test data. An interesting feature of our new method is that the variance in the training data is no longer an insurmountable lower bound on the generalization error.

Keywords: Deep learning, generalization error, interpolatory approximation

This work was supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216.

An analysis of training and generalization errors in shallow and deep networks

H. N. Mhaskar* and T. Poggio †

February 20, 2018

Abstract

An open problem around deep networks is the apparent absence of over-fitting despite large over-parametrization which allows perfect fitting of the training data. In this paper, we explain this phenomenon when each unit evaluates a trigonometric polynomial. It is well understood in the theory of function approximation that approximation by trigonometric polynomials is a "role model" for many other processes of approximation that have inspired many theoretical constructions also in the context of approximation by neural and RBF networks. In this paper, we argue that the maximum loss functional is necessary to measure the generalization error. We give estimates on exactly how many parameters ensure both zero training error as well as a good generalization error, and how much error to expect at which test data. An interesting feature of our new method is that the variance in the training data is no longer an insurmountable lower bound on the generalization error.

Keywords: Deep learning, generalization error, interpolatory approximation

1 Introduction

The main problem of machine learning is the following. Given data $(\mathbf{x}, y)$ sampled from an unknown probability distribution μ, the goal is find a function P that minimizes the generalization error $\mathbb{E}_\mu((y-P)^2)$ among all functions in some function class that is thought to represent the prior information about the distribution. Since we do not know μ, classical machine learning paradigm expresses the generalization error into three components: the variance, the approximation error, and the sampling error. The variance is a lower bound on the generalization error, and the estimation typically focuses on the other two errors. The sum of these two is given by $\mathbb{E}((f-P)^2)$, where the expectation is with respect to the marginal distribution of $\mathbf{x}$ and the *target function* f is the conditional expectation of y given $\mathbf{x}$. If the marginal distribution of $\mathbf{x}$ is known, then the split between approximation and sampling errors is no longer necessary, and one can obtain estimates and as well constructions directly from characteristics of the training data (e.g., [5, 7, 11]). In the classical paradigm where this distribution is not known, the approximation error decreases as the number of parameters in P increases to ∞. However, this makes the empirical risk minimization problem harder to solve; making it essential to choose the number of parameters in P to balance the two errors. In turn, this explains a commonly observed phenomenon that if one achieves a zero empirical risk on the training data by over-parametrized model P, the test error is generally not good.

There are several recent papers that demonstrate that this phenomenon is often not observed (e.g., [4, 13, 16, 19, 1, 15]). One of the vexing problems around deep learning is to explain the generalization error; why do the deep learning models, despite being highly over-parameterized, still predict well? It is suggested in [18] that a new approach to studying the notion of generalization error is required for understanding deep learning.

In the study of approximation error in deep learning, it is observed in [12] that the compositional structure of the target function can be utilized effectively via a property called good propagation of error to obtain substantially better error bounds allowing us to overcome the curse of dimensionality observed in shallow networks. However, it is essential to change the learning paradigm for this purpose drastically. For example, suppose we wish to approximate

*Department of Mathematics, California Institute of Technology, Pasadena, CA 91125; Institute of Mathematical Sciences, Claremont Graduate University, Claremont, CA 91711. The research of this author is supported in part by ARO Grant W911NF-15-1-0385. email: hrushikesh.mhaskar@cgu.edu

†Center for Brains, Minds, and Machines, McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA, 02139. The research of this author is supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216. tp@mit.edu

a function $f^* = f(f_1(\mathbf{x}_1), f_2(\mathbf{x}_2))$ by a network of the form $P^* = P(P_1(\mathbf{x}_1), P_2(\mathbf{x}_2))$, where $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$. Without the compositional structure, f^* has to be treated as a function of $2d$ variables. The compositional structure allows us to treat the approximation problem as a set of three approximation problems: approximating functions f_1, f_2 as functions of d variables each, and a function f of 2 variables.

How do we define generalization error? Defining it in terms of the original distribution of $((\mathbf{x}_1, \mathbf{x}_2), y)$ is not sensitive to the compositional structure. On the other hand, the input (f_1, f_2) to the function f is not the same (even may not have the same distribution) as the input (P_1, P_2) to the approximation P.

Our new paradigm therefore proposes to measure the errors in the uniform norm rather than searching for appropriate L^2 norms suitable for the compositionality structure of the target function. Thus, we define the generalization error as the maximal error between the target function and the model at all possible test points. One consequence is that the decomposition of the generalization error into three parts breaks down. Therefore, new ideas are required to achieve the approximation in terms of the training data alone. The approximation errors themselves are studied in [12, 8], but the techniques suggested there require the data points $\mathbf{x}_j$ to be sufficiently dense in the domain (Euclidean space, sphere, cube, etc.). When the data is not dense, it is clearly not expected to get a good approximation on the whole domain. However, if the data does become dense on some subset of the domain, one can expect a good approximation at points close by to the training data. With the new definition of generalization error, the variance is no longer an insurmountable lower bound on the generalization error.

In this paper, we initiate a systematic theoretical study of the relationship between the structure of the data and the extent of over-parametrization needed to allow the phenomena described above. It is well understood (e.g. [14, 6]) that as far as the number of parameters is concerned, approximation by neural networks with smooth activation functions is equivalent to that by algebraic polynomials. Moreover, it is also well known that approximation of a function f on the Euclidean cube $[-1, 1]^q$ by algebraic polynomials is equivalent to that of the function $f^\circ(\boldsymbol{\theta}) = f(\cos\theta_1, \cdots, \cos\theta_q)$ defined on the q-dimensional torus by trigonometric polynomials of the same degree. The theory of function approximation by trigonometric polynomials serves a "role model" for every other process of approximation. In [10, 9, 11], this theory has been used to suggest some explicit constructions and bounds in the context of approximation by neural and RBF networks. Therefore, we will focus in this paper on networks evaluating trigonometric polynomials.

After introducing the notation in Section 2, we will discuss our main results in the case of shallow networks in Section 3. Our view of deep networks will be discussed in Section 4, and the analogues of the theorems in Section 3 in this context will be proved. The proofs of the results in Section 3 are given in Section 5 and Section 6.

2 Notation

Let $q \ge 1$ be an integer, $\mathbb{T}^q$ be the q dimensional torus $(=\mathbb{R}^q/(2\pi\mathbb{Z})^q)$. For $\mathbf{x}, \mathbf{y} \in \mathbb{T}^q$,

$$|\mathbf{x} - \mathbf{y}| = \max_{1\le i\le q} |(x_i - y_i)(\mod 2\pi)|.$$

For multi-integer $\mathbf{k}$, $|\mathbf{k}|_p$ is ℓ^p norm of $\mathbf{k}$. If $p = 2$, we will write $|\mathbf{k}|$ in place of $|\mathbf{k}|_2$.

The space of all continuous functions $f : \mathbb{T}^q \to \mathbb{R}$, equipped with the supremum norm will be denoted by C^* (or $C^*(\mathbb{T}^q)$ if we wish to emphasize the input dimension to the functions). The norm on $C^*(\mathbb{T}^q)$ will be denoted by $\|\cdot\|$ or $\|\cdot\|_q$ if it is important to identify the dimension. For $n > 0$, the space $\mathbb{H}_n^q$ is defined by

$$\mathbb{H}_n^q = \mathsf{span}\{\exp(i\mathbf{k}\cdot\circ) : |\mathbf{k}| < n\}.$$

The dimension of $\mathbb{H}_n^q$ is $\sim n^q$. If $f \in C^*(\mathbb{T}^q)$, then its degree of approximation from $\mathbb{H}_n^q$ is defined by

$$E_n(f) = E_n(q; f) := \inf_{T\in\mathbb{H}_n^q} \|f - T\|.$$

When our models are trigonometric polynomials in $\mathbb{H}_n^q$, the quantity $E_n(f)$ denotes the ideal generalization error for the target function f. It is well known that there exists a trigonometric polynomial for which this error bound is attained. However, this polynomial of best approximation is neither easy nor worthwhile to construct. It is known to have many bad properties; for example, the operator of best approximation is non-linear, and very sensitive to lack of smoothness of f even at one point.

For any finite subset $\mathcal{C} \subset \mathbb{T}^q$, we define its minimal separation by

$$\eta(\mathcal{C}) = \min_{\mathbf{x},\mathbf{y}\in\mathcal{C},\mathbf{x}\neq\mathbf{y}} |\mathbf{x} - \mathbf{y}|. \tag{2.1}$$

We assume a training data of the form $\mathcal{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^M$, where $\mathcal{C} = \{\mathbf{x}_j\}_{j=1}^M \subset \mathbb{T}^q$, and $y_j = f(\mathbf{x}_j) + \epsilon_j$ for some $f \in C^*$. We denote

$$\epsilon = \max_{1\le j\le M} |\epsilon_j|. \tag{2.2}$$

The quantity ϵ plays the role of variance in our theory in this paper.

The constant convention

The symbols $c, c_1, \cdots$ will denote generic positive constants, depending on such fixed parameters of the problem as q, h, $\mathcal{G}$, and S (to be introduced later), etc. and other quantities explicitly indicated, but their value may different at different occurrences, even within a single formula. The notation $A \sim B$ means that $c_1 A \le B \le c_2 A$.

3 Shallow networks

Our first theorem in this section shows the connection between the structural properties of the training data and the existence of a trigonometric polynomial $T_N^\#(\mathcal{D})$ that can interpolate the noisy data (i.e., achieve a zero training error), as well as achieve a good generalization error in the sense defined in Section 1.

Theorem 3.1 *Let $f \in C^*$. There exists $B > 0$ with the following property: for $N \ge B\eta(\mathcal{C})^{-1}$, there exists $T_N^\#(\mathcal{D}) \in \mathbb{H}_N^q$ such that*

$$T_N^\#(\mathcal{D})(\mathbf{x}_j) = y_j, \qquad j = 1, \cdots, M, \qquad \textit{(Zero training error)} \tag{3.1}$$

and

$$\|f - T_N^\#(\mathcal{D})\| \le c\left\{\epsilon + E_{N/2}(f)\right\}. \qquad \textit{(Good generalization error).} \tag{3.2}$$

Remark 3.1 A volume comparison argument shows that the number of data points M satisfies $M \le c\eta(\mathcal{C})^{-q}$. Thus, Theorem 3.1 shows that for a right configuration of the training data, a good generalization error as well as zero training error can be achieved by choosing the number of parameters proportional to the number of data points. It is demonstrated in [1] that for RBF approximation, this phenomenon seems to hold in many applications with the number of parameters exactly equal to the number of data points.

Remark 3.2 In practice, the training data is high dimensional and sparse; i.e., $\eta(\mathcal{C})$ is large. The requirement that $N \ge B\eta(\mathcal{C})^{-1}$ is therefore satisfied with moderate degrees N.

Remark 3.3 A curious feature of Theorem 3.1 is that one obtains the bound (3.2) on the generalization on the entire torus $\mathbb{T}^q$ without requiring that the training data $\mathcal{C}$ be "dense" in $\mathbb{T}^q$. Of course, Theorem 3.1 is not constructive. The proof of this theorem shows that the polynomial $T_N^\#$ actually utilizes also the Fourier coefficients of the target function in addition to the information contained in $\mathcal{D}$.

Clearly, one cannot construct $T_N^\#(\mathcal{D})$ based only on the training data $\mathcal{C}$, unless $\mathcal{C}$ is sufficiently dense on $\mathbb{T}^q$. If we anticipate a scenario where the training data sets become denser and denser on some compact subset $K \subset \mathbb{T}^q$, then we cannot expect convergence of trigonometric polynomials that interpolate a noisy data, where the noise level does not decrease as well. The following discussion suggests a construction that keeps both training and generalization errors under control.

The space $W^* = W^*(\mathbb{T}^q)$ consists of all continuously differentiable functions $f \in C^*$. We define

$$\|f\|_{W^*} = \|f\|_{W^*(\mathbb{T}^q)} = \sum_{j=1}^q \|D_j f\|. \tag{3.3}$$

For $n > 0$ and $T \in \mathbb{H}_n^q$, let

$$R_n(T) = \max_{1\le j\le M} |y_j - T(\mathbf{x}_j)| + \frac{1}{n}\|T\|_{W^*}. \tag{3.4}$$

Theorem 3.2 *Let $f \in W^*$, B be as in Theorem 3.1, and $N \ge B\eta^{-1}$. Then*

$$\min_{T\in\mathbb{H}_N^q} R_N(T) \le c\left\{\epsilon + \frac{1}{N}\|f\|_{W^*}\right\}. \qquad \textit{(Good training error)} \tag{3.5}$$

Let $T^*(\mathcal{D}) = \underset{T\in\mathbb{H}_N^q}{arg\ min}\, R_N(T)$, $\mathbf{x}\in\mathbb{T}^q$, *and* $\delta = \min_{1\le j\le M}|\mathbf{x}-\mathbf{x}_j|$. *Then*

$$|f(\mathbf{x}) - T^*(\mathcal{D})(\mathbf{x})| \le c(1+N\delta)\left\{\epsilon + \frac{1}{N}\|f\|_{W^*}\right\}. \qquad \textit{(Good generalization error).} \tag{3.6}$$

Remark 3.4 The estimate (3.6) shows that if $\mathbf{x}$ is very close to the training data so that $N\delta < 1$, then the generalization error at $\mathbf{x}$ is of the same order of magitude as the loss in the training data measured by $R_N(T^*)$. As remarked earlier in Remark 3.2, we have greater liberty in choosing a large N when the data is sparse. To take advantage of this fact, let $\mathbf{x}$ be such that $N\delta \ge 1$. Then (3.6) can be reformulated in the form

$$|f(\mathbf{x}) - T^*(\mathcal{D})(\mathbf{x})| \le c\delta\left\{N\epsilon + \|f\|_{W^*}\right\}. \tag{3.7}$$

The term $\delta\|f\|_{W^*}$ is clearly a customary bound from numerical analysis in view of the mean value theorem. If $\epsilon \le \eta(\mathcal{C})\|f\|_{W^*}$, it is possible to choose N so that error bound is $c\delta\|f\|_{W^*}$. It is interesting to note that in the new paradigm, the variance in the dependent variable is no longer a lower bound on the generalization error; one can reduce the noise with sufficient over-parametrization.

Remark 3.5 It is well known (cf. [20, Chapter X, Theorem 7.28] for the univariate case) that for any $T\in\mathbb{H}_N^q$,

$$\|T\| \sim \max_{|\mathbf{k}|_\infty\le 3N-1}\left|T\left(\frac{2\pi\mathbf{k}}{3N}\right)\right|. \tag{3.8}$$

Therefore, the term $\|T\|_{W^*}$ in regularization functional $R_n(T)$ can be calculated in terms of the Fourier coefficients of T via matrix vector multiplications involving the discrete evaluations of D_jT as in (3.8). The results are not affected except for the actual values of the constants involved.

4 Deep networks

The following discussion about the terminology about the deep networks, including Figure 1, is based on the discussion in [12], and elaborates upon the same.

Let $\mathcal{G}$ be a directed acyclic graph (DAG), with the set of nodes V. A $\mathcal{G}$-function is defined as follows. The in-edges to each node of $\mathcal{G}$ represents an input real variable. For each node v, we denote its in-degree by $d(v)$. The node v itself represents the evaluation of a real valued function f_v of the $d(v)$ inputs. The out-edges fan out the result of this evaluation. Each of the source node obtains an input from some Euclidean space. Other nodes can also obtain such an input, but by introducing dummy nodes, it is convenient to assume that only the source nodes obtain an input from the Euclidean space. The set of all source nodes of $\mathcal{G}$ will be denoted by $\mathbf{S}$ (or $\mathbf{S}(\mathcal{G})$ if necessary).

We note that if q is the number of source nodes in $\mathcal{G}$, a $\mathcal{G}$–function is a function on $\mathbb{R}^q$. Viewed only as a function on $\mathbb{R}^q$, it is not clear whether two different DAG structures can give rise to the same function. Even if we assume a certain DAG, it is not clear that the choice of the constituent functions is uniquely determined for a given function on $\mathbb{R}^q$. For our mathematical analysis, we therefore find it convenient to think of a $\mathcal{G}$–function as a set of functions $f = \{f_v : \mathbb{R}^{d(v)}\to\mathbb{R}\}_{v\in V}$, rather than a single function on $\mathbb{R}^q$. The individual functions f_v will be called *constituent functions*.

For example, the DAG $\mathcal{G}$ in Figure 1 represents the compositional function

$$\begin{aligned} f^*(x_1,\cdots,x_9) &= h_{19}(h_{17}(h_{13}(h_{10}(x_1,x_2,x_3,h_{16}(h_{12}(x_6,x_7,x_8,x_9))),h_{11}(x_4,x_5)),\\ &\qquad h_{14}(h_{10},h_{11}),h_{16}),h_{18}(h_{15}(h_{11},h_{12}),h_{16})) \end{aligned} \tag{4.1}$$

The $\mathcal{G}$-function is $\{h_{10},\cdots,h_{19}=f^*\}$.

We assume that there is only one sink node, v^* (or $v^*(\mathcal{G})$) whose output is denoted by f^*. Technically, there are two functions involved here: one is the final output as a function of all the inputs to all source nodes, the other is the final output as a function of the inputs to the node v^*. We will use the symbol f^* to denote both with comments on which meaning is intended when we feel that it may not be clear from the context. A similar convention is followed with respect to each of the constituent functions as well. For example, in the DAG of Figure 1, the function h_{14} can be thought of both as a function of two variables, namely the outputs of h_{10} and h_{11} as well as a function of five variables $x_1,\cdots,x_5$.

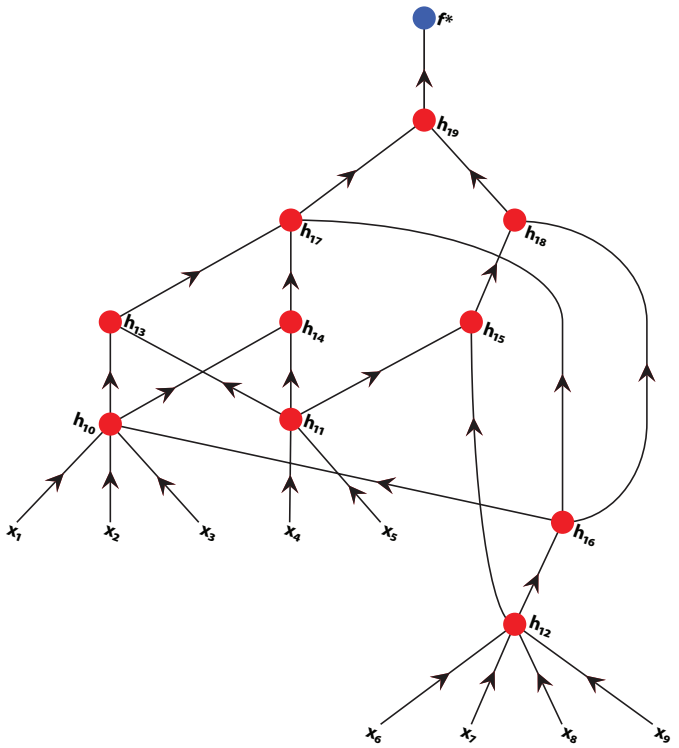

Figure 1: An example of a $\mathcal{G}$–function (f^* given in (4.1)). The vertices of the DAG $\mathcal{G}$ are denoted by red dots. The black dots represent the input to the various nodes as indicated by the in–edges of the red nodes, and the blue dot indicates the output value of the $\mathcal{G}$–function, f^* in this example.

In this paper, we are interested only in the case where each of the inputs to each of the source node is in $\mathbb{T}$ rather than $\mathbb{R}$. Although this is no longer true for the non-source nodes, it is possible to accomplish this in the case when each of the constituent functions is continuous, as follows. Let f_v be one of the constituent functions. With a re-normalization, we may assume that the range of f_v is a subset of $[-1, 1]$. Then the function $f_v^\circ(\mathbf{x}) = \arccos(f_v(\mathbf{x}))$, extended to $\mathbb{T}$ as an even function, is a function whose range is a subset of $\mathbb{T}$. Rather than complicating our notations, we will therefore assume in this paper that the domain of each constituent function is a torus, and the range is a subset of $\mathbb{T}$. In particular, we may assume that every constituent function $f_v \in C^*(\mathbb{T}^{d(v)})$.

We adopt the convention that for any function class $\mathbb{X}(\mathbb{T}^d)$, the class $\mathcal{G}$-$\mathbb{X}$ denotes the set of $\mathcal{G}$ functions $f = \{f_v\}_{v\in V}$, where each constituent function $f_v \in \mathbb{X}(\mathbb{T}^{d(v)})$. We define

$$\|f\|_{\mathcal{G},\mathbb{X}} = \sum_{v\in V} \|f_v\|_{\mathbb{X}(\mathbb{T}^{d(v)})}. \tag{4.2}$$

Thus, for example, $\mathcal{G}$-W^* is the set of all $\mathcal{G}$ functions $f = \{f_v\}_{v\in V}$, where each constituent function $f_v \in \mathbb{X}(\mathbb{T}^{d(v)})$, and we write

$$\|f\|_{\mathcal{G},W^*} = \sum_{v\in V} \|f_v\|_{W^*(\mathbb{T}^{d(v)})}.$$

The symbol $\mathcal{G}$-$\mathbb{H}_N$ denotes the set of $\mathcal{G}$ functions $\{T_v\}_{v\in V}$, where each $T_v \in \mathbb{H}_N^{d(v)}$, and for a $\mathcal{G}$ function $f = \{f_v\}_{v\in V}$,

$$E_{N,\mathcal{G}}(f) = \sum_{v\in V} E_N(d(v); f_v). \tag{4.3}$$

To make precise the various inputs to the constituent functions, we introduce some conventions. We assume an enumeration of V such that $\mathbf{S} = \{v_1, \cdots, v_s\}$, $q = \sum_{j=1}^s d(v_j)$, (i.e., the source nodes appear before other nodes). The input $\mathbf{x}$ can be viewed as a vector in $\mathbb{R}^q$, but it is sometimes convenient to think of it as a set of its components (ignoring the order). This set may then be partitioned as $\mathbf{x} = \{\mathbf{x}_{v_1}, \cdots, \mathbf{x}_{v_s}\}$ so that $\mathbf{x}_{v_j} \in \mathbb{R}^{d(v_j)}$ (again with the dual interpretation). Correspondingly, we define the sets

$$\mathcal{C}_{v_j} = \{(\mathbf{x})_{v_j} : \mathbf{x} \in \mathcal{C}\}.$$

Thus, $\mathcal{C}_{v_j}$ is the training data "seen" by the source node v_j. This notion is extended recursively to other nodes of $\mathcal{G}$. Let v not be a source node, $u_1, \cdots, u_{d(v)}$ be the children of v, $\mathcal{C}_{u_1}, \cdots, \mathcal{C}_{u_{d(v)}}$ be the training data seen by these nodes. Thus, for any $\mathbf{x} \in \mathcal{C}$, the components given by $(\mathbf{x})_{u_j}$ are seen by u_j.

$$\mathcal{C}_v = \left\{(f_{u_1}((\mathbf{x})_{u_1}), \cdots, f_{u_{d(v)}}((\mathbf{x})_{u_{d(v)}}) : \mathbf{x} \in \mathcal{C}\right\}.$$

For example, in the DAG of Figure 1, the children of h_{14} are h_{10} and h_{11}. For each $\mathbf{x} \in \mathbb{R}^9$, h_{10} sees the components (x_1, x_2, x_3), while h_{11} sees the components (x_4, x_5). We have

$$\begin{aligned} \mathcal{C}_{h_{10}} = \{((\mathbf{x})_1, (\mathbf{x})_2, (\mathbf{x})_3) : \mathbf{x} \in \mathcal{C}\}, \quad \mathcal{C}_{h_{11}} = \{((\mathbf{x})_4, (\mathbf{x})_5) : \mathbf{x} \in \mathcal{C}\}, \\ \mathcal{C}_{h_{14}} \quad = \quad \{((h_{10}((\mathbf{x})_1, (\mathbf{x})_2, (\mathbf{x})_3), h_{11}((\mathbf{x})_4, (\mathbf{x})_5)) : \mathbf{x} \in \mathcal{C}\} \end{aligned}$$

The analogue of Theorem 3.1 for deep networks naturally requires a condition on the constituent functions so as to be able to keep track of the minimal separations among the inputs to the different nodes. A function $f : \mathbb{T}^d \to \mathbb{R}$ is said to be *bi-Lipschitz* if there exists a positive constant $c(f) > 0$ such that

$$\frac{1}{c(f)}|\mathbf{x} - \mathbf{y}| \le |f(\mathbf{x}) - f(\mathbf{y})| \le c(f)|\mathbf{x} - \mathbf{y}|, \qquad \mathbf{x}, \mathbf{y} \in \mathbb{T}^q. \tag{4.4}$$

The analogue of Theorem 3.1 is the following.

Theorem 4.1 *Let $\mathcal{G}$ be a DAG with source nodes $\mathbf{S} = \{v_1, \cdots, v_s\}$ and sink node v^*. Let $f = \{f_v\}_{v \in V}$ be a $\mathcal{G}$-function such that each of the constituent functions is bi-Lipschitz. Let $\eta = \min_{1 \le j \le s} \eta_j(\mathcal{C}_{v_j})$, $d = \min_{v \in V} d(v_j)$. There exists $C = C(f) > 0$ with the following property: for $N \ge C(f)\eta^{-1}$, there exists $T_N^\#(\mathcal{D}) \in \mathcal{G}$-$\mathbb{H}_N$ such that*

$$(T_N^\#(\mathcal{D}))_{v^*}(\mathbf{x}_j) = y_j, \qquad j = 1, \cdots, M, \qquad \textit{(Zero training error)} \tag{4.5}$$

and

$$\|f - T_N^\#(\mathcal{D})\|_{\mathcal{G}} \le c\left\{\epsilon + E_{N/2,\mathcal{G}}(f)\right\}. \qquad \textit{(Good generalization error).} \tag{4.6}$$

Proof of Theorem 4.1. Since each of the functions f_v is bi-Lipschitz, $\eta(\mathcal{C}_v) \ge c\eta$ for each $v \in V$. Therefore, Theorem 3.1 implies that for each non-sink node v, there is $T_v^\# \in \mathbb{H}_N^{d(v)}$ satisfying

$$T_v^\#(\mathbf{x}) = f_v(\mathbf{x}), \qquad \mathbf{x} \in \mathcal{C}_v, \qquad \|f_v - T_v^\#\| \le cE_{N/2}(d(v); f_v). \tag{4.7}$$

Similarly, there is $T_{v^*}^\# \in \mathbb{H}_N(d(v^*))$ that satisfies (thought of as a function of all the inputs to the network)

$$T_{v^*}^\#(\mathbf{x}_j) = T_{v^*}^\#((\mathbf{x}_j)_{v^*}) = y_j, \qquad j = 1, \cdots, M, \tag{4.8}$$

and (thought of as a function on $\mathbb{T}^{d(v^*)}$)

$$\|f_{v^*} - T_{v^*}^\#\| \le c\left\{\epsilon + E_{N/2}(d(v^*); f_{v^*})\right\}. \tag{4.9}$$

Then $T_N^\# = \{T_v^\#\} \in \mathcal{G}$-$\mathbb{H}_N$. The equation (4.8) is the same as (4.5). Since each f_v is Lipschitz continuous, the estimate (4.6) follows from (4.7) and (4.9) by a repeated application of triangle inequality (*good propagation property*, cf. [12, Proof of Theorem 2.1]). ■

The analogue of Theorem 3.2 is similar; but one has to take into account the fact that one does not know the constituent functions. Nevertheless, knowing the DAG structure, one can construct a DAG trigonometric polynomial. For $n > 0$ and $T \in \mathcal{G}$-$\mathbb{H}_n$, let

$$R_{n,\mathcal{G}}(T) = \max_{1 \le j \le M} |y_j - T_{v^*}(\mathbf{x}_j)| + \frac{1}{n}\|T\|_{W^*,\mathcal{G}}. \tag{4.10}$$

In this definition, it is understood that T_{v^*} is thought of as a function of the q-dimensional vector $\mathbf{x}$, but is computed using the all the constituent functions in T using DAG structure prescribed by $\mathcal{G}$.

Theorem 4.2 *Let $\mathcal{G}$ be a DAG with source nodes $\mathbf{S} = \{v_1, \cdots, v_s\}$ and sink node v^*. Let $f = \{f_v\}_{v \in V} \in \mathcal{G}$-$W^*$, and each of the constituent functions be bi-Lipschitz. Let N be as in Theorem 4.1. Then with the notation as in that theorem,*

$$\min_{T \in \mathcal{G}\text{-}\mathbb{H}_N} R_{N,\mathcal{G}}(T) \le c\left\{\epsilon + \frac{1}{N}\|f\|_{W^*,\mathcal{G}}\right\}. \qquad \textit{(Good training error)} \tag{4.11}$$

Let $\mathbf{x} \in \mathbb{T}^q$, *and* $\delta = \min_{1\le j\le M} |\mathbf{x}-\mathbf{x}_j|$, *and* $T^*(\mathcal{D}) = \underset{\mathcal{G}\text{-}\mathbb{H}_N}{arg\ min} R_{N,\mathcal{G}}(T)$, *then*

$$|f(\mathbf{x}) - (T^*(\mathcal{D}))_{v^*}(\mathbf{x})| \le cN\delta \left\{\epsilon + \frac{1}{N}\|f\|_{W^*}\right\}. \qquad \textit{(Good generalization error).} \tag{4.12}$$

Proof of Theorem 4.2. This proof is essentially the same as that of Theorem 3.2. We point out some considerations required in the details which are different. Since $f \in \mathcal{G}\text{-}W^*$, the simultaneous approximation theorem Corollary 6.1 holds for each f_v. We may assume that the perturbation exists only at the sink node, and the rest of the data is exact. Finally, we use the good propagation property. ■

Remark 4.1 The theorems in this section indicate that the superiority of deep learning comes from two factors. One is that the compositional structure of the target function allows us to study the problem in a cascade of low dimensional problems. The other is that this structure might allow us to sparsify the training data as we move up the cascade.

5 Proof of Theorem 3.1.

The proof depends upon the construction and properties of a highly localized trigonometric polynomial kernel.

Let $h : \mathbb{R} \to [0,1]$ be an infinitely differentiable, even function such that $h(t) = 1$ if $|t| \le 1/2$, $h(t) = 0$ if $|t| \ge 1$. We define

$$\Phi_N(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^q} h\left(\frac{|\mathbf{k}|}{N}\right)\exp(i\mathbf{k}\cdot\mathbf{x}), \qquad \mathbf{x}\in\mathbb{T}^q,\ N>0. \tag{5.1}$$

If $f \in C^*$, the Fourier coefficients of f are defined by

$$\hat{f}(\mathbf{k}) := \frac{1}{(2\pi)^q}\int_{\mathbb{T}^q} f(\mathbf{x})\exp(-i\mathbf{k}\cdot\mathbf{x})d\mathbf{x}, \qquad \mathbf{k}\in\mathbb{Z}^q. \tag{5.2}$$

For $f \in C^*$, we define

$$\sigma_N(f)(\mathbf{x}) = \sum_{\mathbf{k}\in\mathbb{Z}^q} h\left(\frac{|\mathbf{k}|}{N}\right)\hat{f}(\mathbf{k})\exp(i\mathbf{k}\cdot\mathbf{x}), \qquad \mathbf{x}\in\mathbb{T}^q,\ N>0. \tag{5.3}$$

We note that the sums in both (5.1) and (5.3) are finite sums, although they are written as infinite sums for convenience of notation.

The following proposition summarizes some essential properties of these kernels and operators.

Proposition 5.1 *Let* $S > q$ *be an integer,* $1 \le p \le \infty$.
(a) For $N \ge 1$,

$$|\Phi_N(\mathbf{x})| \le \frac{cN^q}{\max(1,(N|\mathbf{x}|)^S)}, \quad , \qquad \mathbf{x}\in\mathbb{T}^q, \tag{5.4}$$

and

$$|\Phi_N(\mathbf{0})| \ge cN^q. \tag{5.5}$$

(b) If $\mathcal{C} = \{\mathbf{x}_1,\cdots,\mathbf{x}_M\}$. *There exists a constant* $B > 0$ *with the following property: if* $N \ge B\eta(\mathcal{C})^{-1}$ *then then for* $\mathbf{x}\in\mathbb{T}^q$,

$$\sum_{j=1}^{M} |\Phi_N(\mathbf{x}-\mathbf{x}_j)| \le cN^q. \tag{5.6}$$

and

$$\sum_{j:|\mathbf{x}_j-\mathbf{x}|\ge\eta} |\Phi_N(\mathbf{x}-\mathbf{x}_j)| \le (1/2)\Phi_N(\mathbf{0}) = (1/2)\Phi_N(\mathbf{x}-\mathbf{x}) \le cN^q. \tag{5.7}$$

(c) If $T \in \mathbb{H}^q_{N/2}$ *then* $\sigma_N(T) = T$. *Further,*

$$\|\sigma_N(f)\| \le c\|f\|, \qquad f\in C^*. \tag{5.8}$$

Consequently,

$$E_N(f) \le \|f-\sigma_N(f)\| \le cE_{N/2}(f), \qquad f\in C^*. \tag{5.9}$$

PROOF. Part (a) is proved in [2, Theorem 6.1]. It is not difficult to verify (cf. [7, Lemma 5.3]) that for any $\mathbf{x} \in \mathbb{T}^q$ and $r > 0$,

$$|\{\ell : |\mathbf{x}_\ell - \mathbf{x}| \le r\}| \le c\eta(\mathcal{C})^{-q}(r^q + \eta(\mathcal{C})^q).$$

The estimate follows from this fact and [7, Proposition 5.1, Definition 4.1], taking into account that $N \ge \eta^{-1}$. The estimate (5.7) follows from (5.6) and (5.5) taking into account the fact that there can be at most one $\mathbf{x}_j$ with $|\mathbf{x} - \mathbf{x}_j| < \eta(\mathcal{C})/2$. The estimate (5.8) is given in [2, Corollary 6.1] (The notation in [2] is different from the one used here). If $T \in \mathbb{H}^q_{N/2}$, then $\hat{T}(\mathbf{k}) = 0$ if $|\mathbf{k}| \ge N/2$, while $h(t) = 1$ if $|t| \le 1/2$. It follows from the definition (5.3) that $\sigma_N(T) = T$. Together with (5.8), this leads to (5.9). ■

A consequence of the above proposition is the following (cf. [7, Proposition 6.1]).

Lemma 5.1 *Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^M$, and*

$$\sum_{j=1}^{M} a_j \Phi_N(\mathbf{x}_\ell - \mathbf{x}_j) = b_\ell, \qquad \ell = 1, \cdots, M. \tag{5.10}$$

Then

$$\max_{1\le j\le M} |a_j| \le cN^{-q} \max_{1\le \ell\le M} |b_\ell|. \tag{5.11}$$

Proof of Theorem 3.1. In this proof, let N be an integer satisfying the conditions of Proposition 5.1(b), and for $\ell = 1, \cdots, M$, $z_\ell = f(\mathbf{x}_\ell) - \sigma_N(f)(\mathbf{x}_\ell)$. In view of (5.5) and (5.7), and a well known result in linear algebra (cf. [7, Proposition 6.1]), there exist $a_j \in \mathbb{R}$ such that

$$\sum_{j=1}^{M} a_j \Phi_N(\mathbf{x}_\ell - \mathbf{x}_j) = z_\ell = y_\ell - \sigma_N(f)(\mathbf{x}_\ell), \qquad \ell = 1, \cdots, M, \tag{5.12}$$

and (cf. (5.9))

$$\begin{aligned}
\max_{1\le j\le M} |a_j| &\le cN^{-q} \max_{1\le \ell\le M} |z_\ell| = cN^{-q} \max_{1\le \ell\le M} |y_\ell - \sigma_N(\mathbf{x}_\ell)| \\
&\le cN^{-q} \left\{ \max_{1\le \ell\le M} |y_\ell - f(\mathbf{x}_\ell)| + \max_{1\le \ell\le M} |f(\mathbf{x}_\ell) - \sigma_N(\mathbf{x}_\ell)| \right\} \\
&\le cN^{-q} \left\{ \epsilon + E_{N/2}(f) \right\}.
\end{aligned} \tag{5.13}$$

We now define

$$\mathcal{T}_N^{\#}(\mathcal{D})(\mathbf{x}) = \sigma_N(f)(\mathbf{x}) + \sum_{j=1}^{M} a_j \Phi_N(\mathbf{x} - \mathbf{x}_j), \qquad \mathbf{x} \in \mathbb{T}^q. \tag{5.14}$$

Clearly, $\mathcal{T}_N^{\#}(\mathcal{D}) \in \mathbb{H}^q_N$, and $\mathcal{T}_N^{\#}(\mathcal{D})(\mathbf{x}_\ell) = y_\ell$, $\ell = 1, \cdots, M$. This proves (3.1).

Moreover, for every $\mathbf{x} \in \mathbb{T}^q$, (5.14) and (5.9) lead to

$$\begin{aligned}
|f(\mathbf{x}) - \mathcal{T}_N^{\#}(\mathcal{D})(\mathbf{x})| &\le |f(\mathbf{x}) - \sigma_N(f)(\mathbf{x})| + \left\{ \max_{1\le j\le M} |a_j| \right\} \sum_{j=1}^{M} |\Phi_N(\mathbf{x} - \mathbf{x}_j)| \\
&\le c \left\{ E_{N/2,\infty}(f) + cN^{-q} \left\{ \epsilon + E_{N/2,\infty}(f) \right\} \sum_{j=1}^{M} |\Phi_N(\mathbf{x} - \mathbf{x}_j)| \right\}.
\end{aligned} \tag{5.15}$$

In view of (5.6), this leads to (3.2). ■

6 Proof of Theorem 3.2.

In order to prove Theorem 3.2, we need some preparation.

We recall a theorem from [3, Theorem 1°].

Theorem 6.1 *Let $q = 1$, $f \in W^*(\mathbb{T})$, $n \geq 1$ be an integer, $E > 0$, and $T \in \mathbb{H}_n^1$ satisfy*

$$\|f - T\| \leq E. \tag{6.1}$$

Then

$$\|f' - T'\| \leq c\left\{nE + E_n(f')\right\}. \tag{6.2}$$

In the multivariate case, we take the derivatives one variable at a time to deduce the following corollary of Theorem 6.1.

Corollary 6.1 *Let $f \in W^*$, $n \geq 1$ be an integer, $E > 0$, and $T \in \mathbb{H}_N^q$ satisfy*

$$\|f - T\| \leq E. \tag{6.3}$$

Then

$$\|f - T\|_{W^*} \leq c\left\{NE + \sum_{j=1}^{q} E_N(D_j f)\right\}. \tag{6.4}$$

In particular,

$$\|T\|_{W^*} \leq cN\left\{E + \frac{1}{N}\|f\|_{W^*}\right\}. \tag{6.5}$$

We note also the direct theorem of trigonometric approximation [17, Section 5.3].

Proposition 6.1 *If $f \in W^*$ then*

$$E_N(f) \leq \frac{c}{N}\|f\|_{W^*}. \tag{6.6}$$

We are now ready to prove Theorem 3.2.

Proof of Theorem 3.2. In this proof, let $\mathcal{T}^\# = T_N^\#(\mathcal{D})$ be as in Theorem 3.1, and

$$E = \epsilon + E_{N/2}(f). \tag{6.7}$$

In view of (3.2), we may use Corollary 6.1 to deduce that

$$\|f - \mathcal{T}^\#\|_{W^*} \leq c\left\{NE + \sum_{j=1}^{q} E_N(D_j f)\right\} \leq cN\left\{E + \frac{1}{N}\|f\|_{W^*}\right\}. \tag{6.8}$$

and in particular,

$$\|\mathcal{T}^\#\|_{W^*} \leq cN\left\{E + \frac{1}{N}\|f\|_{W^*}\right\}. \tag{6.9}$$

Using (3.1), (6.7) and (6.9), we obtain:

$$\begin{aligned}\min_{T\in\mathbb{H}_N} R_N(T) &\leq R_N(\mathcal{T}^\#) \leq \max_{1\leq j\leq M} |y_j - \mathcal{T}^\#(\mathbf{x}_j)| + \frac{1}{N}\|\mathcal{T}^\#\|_{W^*} \\ &= \frac{1}{N}\|\mathcal{T}^\#\|_{W^*} \leq c\left\{E + \frac{1}{N}\|f\|_{W^*}\right\}.\end{aligned}$$

In view of Proposition 6.1,

$$E \leq c\left\{\epsilon + \frac{1}{N}\|f\|_{W^*}\right\}.$$

Together with (6.9), this proves (3.5).

Next, let $\mathbf{x} \in \mathbb{T}^q$, and $\delta = \min_{1\leq j\leq M} |\mathbf{x} - \mathbf{x}_j| = |\mathbf{x} - \mathbf{x}_\ell|$. For brevity, we write

$$\tilde{E} = \epsilon + \frac{1}{N}\|f\|_{W^*}.$$

Using (3.5) and Corollary 6.1, we deduce that

$$\begin{aligned}|f(\mathbf{x}) - T^*(\mathbf{x})| &\leq |f(\mathbf{x}) - f(\mathbf{x}_\ell)| + |f(\mathbf{x}_\ell) - T^*(\mathbf{x}_\ell)| + |T^*(\mathbf{x}_\ell) - T^*(\mathbf{x})| \\ &\leq |\mathbf{x} - \mathbf{x}_\ell|\|f\|_{W^*} + c\tilde{E} + |\mathbf{x} - \mathbf{x}_\ell|\|T^*\|_{W^*} \leq N\delta\frac{1}{N}\|f\|_{W^*} + c\tilde{E} + N\delta\tilde{E} \\ &\leq c(1 + N\delta)\tilde{E}.\end{aligned}$$

This proves (3.6). ■

7 Conclusions

We have the puzzle that deep networks (and sometimes also shallow ones) do not exhibit over-fitting even though the number of parameters is very large and the training error is reduced to zero. We have initiated a rigorous study of this phenomenon from the point of view of function approximation, giving estimates on how many parameters are needed to exhibit a zero or good training error, which is also compatible with the generalization error. Our estimates are given in terms of the data characteristics and the smoothness of the target function. An interesting feature is that over-parametrization allows us to overcome the noise in the training data, in contrast to classical machine learning paradigm.

References

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

[2] S. Chandrasekaran, K. R. Jayaraman, and H. N. Mhaskar. Minimum Sobolev norm interpolation with trigonometric polynomials on the torus. *Journal of Computational Physics*, 249:96–112, 2013.

[3] J. Czipser and G. Freud. Sur l'approximation d'une fonction periodique et de ses desivees successires par un polynome trigonometrique et par ses derivees successives. *Acta Math.,(Sweden)*, 99:33–51, 1958.

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

[5] Q. T. Le Gia and H. N. Mhaskar. Localized linear polynomial operators and quadrature formulas on the sphere. *SIAM Journal on Numerical Analysis*, pages 440–466, 2008.

[6] H. N. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Computation*, 8(1):164–177, 1996.

[7] H. N. Mhaskar. Eignets for function approximation on manifolds. *Applied and Computational Harmonic Analysis*, 29(1):63–87, 2010.

[8] H. N. Mhaskar. Function approximation with relu-like zonal function networks. *arXiv preprint arXiv:1709.08174*, 2017.

[9] H. N. Mhaskar and C. A. Micchelli. Dimension-independent bounds on the degree of approximation by neural networks. *IBM Journal of Research and Development*, 38(3):277–284, 1994.

[10] H. N. Mhaskar and C. A. Micchelli. Degree of approximation by neural and translation networks with a single hidden layer. *Advances in Applied Mathematics*, 16(2):151–183, 1995.

[11] H. N. Mhaskar, P. Nevai, and E. Shvarts. Applications of classical approximation theory to periodic basis function networks and computational harmonic analysis. *Bulletin of Mathematical Sciences*, 3(3):485–549, 2013.

[12] H. N. Mhaskar and T. Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

[13] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5949–5958, 2017.

[14] A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[15] T. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. N. Mhaskar. Theory of deep learning III: the non-overfitting puzzle. *CBMM memo 073*, 2018.

[16] J. Sokolić, R. Giryes, G. Sapiro, and M. R. Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2016.

[17] A. F. Timan. *Theory of Approximation of Functions of a Real Variable: International Series of Monographs on Pure and Applied Mathematics*, volume 34. Elsevier, 2014.

[18] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[19] C. Zhang, Q. Liao, A. Rakhlin, B. Miranda, N. Golowich, and T. Poggio. Musings on deep learning: Properties of sgd. *CBMM Memo*, 67, 2017.

[20] A. Zygmund. *Trigonometric series*, volume 1. Cambridge University Press, 2002.