Towards a Mathematical Understanding of Neural Network-Based Machine Learning: what we know and what we don't

Weinan E * [†1,2], Chao Ma [‡3], Stephan Wojtowytsch [§2], and Lei Wu [¶2]

[1]Department of Mathematics, Princeton University
[2]Program in Applied and Computational Mathematics, Princeton University
[3]Department of Mathematics, Stanford University

December 9, 2020

Contents

*Also at Beijing Institute of Big Data Research.
†weinan@math.princeton.edu
‡chaoma@stanford.edu
§stephanw@princeton.edu
¶leiwu@princeton.edu

1 Introduction

Neural network-based machine learning is both very powerful and very fragile. On the one hand, it can be used to approximate functions in very high dimensions with the efficiency and accuracy never possible before. This has opened up brand new possibilities in a wide spectrum of different disciplines. On the other hand, it has got the reputation of being somewhat of a "black magic": Its success depends on lots of tricks, and parameter tuning can be quite an art. The main objective for a mathematical study of machine learning is to

1. explain the reasons behind the success and the subtleties, and

2. propose new models that are equally successful but much less fragile.

We are still quite far from completely achieving these goals but it is fair to say that a reasonable big picture is emerging.

The purpose of this article is to review the main achievements towards the first goal and discuss the main remaining puzzles. In the tradition of good old applied mathematics, we

will not only give attention to rigorous mathematical results, but also discuss the insight we have gained from careful numerical experiments as well as the analysis of simplified models.

At the moment much less attention has been given to the second goal. One proposal that we should mention is the continuous formulation advocated in [33]. The idea there is to first formulate "well-posed" continuous models of machine learning problems and then discretize to get concrete algorithms. What makes this proposal attractive is the following:

- many existing machine learning models and algorithms can be recovered in this way in a *scaled* form;
- there is evidence suggesting that indeed machine learning models obtained this way is more robust with respect to the choice of hyper-parameters than conventional ones (see for example Figure 5 below);
- new models and algorithms are borne out naturally in this way. One particularly interesting example is the maximum principle-based training algorithm for ResNet-like models [58].

However, at this stage one cannot yet make the claim that the continuous formulation is the way to go. For this reason we will postpone a full discussion of this issue to future work.

1.1 The setup of supervised learning

The model problem of supervised learning which we focus on in this article can be formulated as follows: given a dataset $S = \{(\boldsymbol{x}_i, y_i = f^*(\boldsymbol{x}_i)), i \in [n]\}$, approximate f^* as accurately as we can. If f^* takes continuous values, this is called a regression problem. If f^* takes discrete values, this is called a classification problem.

We will focus on the regression problem. For simplicity, we will neglect the so-called "measurement noise" since it does not change much the big picture that we will describe, even though it does matter for a number of important specific issues. We will assume $\boldsymbol{x}_i \in X := [0,1]^d$, and we denote by P the distribution of $\{\boldsymbol{x}_i\}$. We also assume for simplicity that $\sup_{\boldsymbol{x}\in X} |f^*(\boldsymbol{x})| \leq 1$.

Obviously this is a problem of function approximation. As such, it can either be regarded as a problem in numerical analysis or a problem in statistics. We will take the former viewpoint since it is more in line with the algorithmic and analysis issues that we will study.

The standard procedure for supervised learning is as follows:

1. Choose a hypothesis space, the set of trial functions, which will be denoted by $\mathcal{H}_m$. In classical numerical analysis, one often uses polynomials or piecewise polynomials. In modern machine learning, it is much more popular to use neural network models. The subscript m characterizes the size of the model. It can be the number of parameters or neurons, and will be specified later for each model.

2. Choose a loss function. Our primary goal is to fit the data. Therefore the most popular choice is the "empirical risk":

$$\hat{\mathcal{R}}_n(f) = \frac{1}{n}\sum_i (f(\boldsymbol{x}_i) - y_i)^2 = \frac{1}{n}\sum_i (f(\boldsymbol{x}_i) - f^*(\boldsymbol{x}_i))^2$$

   Sometimes one adds some regularization terms.

3. Choose an optimization algorithm and the hyper-parameters. The most popular choices are gradient descent (GD), stochastic gradient descent (SGD) and advanced optimizers such as Adam [52], RMSprop [85].

The overall objective is to minimize the "population risk", also known as the "generalization error":

$$\mathcal{R}(f) = \mathbb{E}_{\boldsymbol{x}\sim P}(f(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2$$

In practice, this is estimated on a finite data set (which is unrelated to any data used to train the model) and called test error, whereas the empirical risk (which is used for training purposes) is called the training error.

1.2 The main issues of interest

From a mathematical perspective, there are three important issues that we need to study:

1. Properties of the hypothesis space. In particular, what kind of functions can be approximated efficiently by a particular machine learning model? What can we say about the generalization gap, i.e. the difference between training and test errors.

2. Properties of the loss function. The loss function defines the variational problem used to find the solution to the machine learning problem. Questions such as the landscape of the variational problem are obviously important. The landscape of neural network models is typically non-convex, and there may exist many saddle points and bad local minima.

3. Properties of the training algorithm. Two obvious questions are: Can we optimize the loss function using the selected training algorithm? Does the solutions obtained from training generalize well to test data?

The second and third issues are closely related. In the under-parametrized regime (when the size of the training dataset is larger than the number of free parameters in the hypothesis space), this loss function largely determines the solution of the machine learning model. In the opposite situation, the over-parametrized regime, this is no longer true. Indeed it is often the case that there are infinite number of global minimizers of the loss function. Which one is picked depends on the details of the training algorithm.

The most important parameters that we should keep in mind are:

- m: number of free parameters
- n: size of the training dataset
- t: number of training steps
- d: the input dimension.

Typically we are interested in the situation when $m, n, t \to \infty$ and $d \gg 1$.

1.3 Approximation and estimation errors

Denote by $\hat{f}$ the output of the machine learning (abbreviated ML) model. Let

$$f_m = \text{argmin}_{f \in \mathcal{H}_m} \mathcal{R}(f)$$

We can decompose the error $f^* - \hat{f}$ into:

$$f^* - \hat{f} = f^* - f_m + f_m - \hat{f}$$

$f^* - f_m$ is the *approximation error*, due entirely to the choice of the hypothesis space. $f_m - \hat{f}$ is the *estimation error*, the additional error due to the fact that we only have a finite dataset.

To get some basic idea about the approximation error, note that classically when approximating functions using polynomials, piecewise polynomials, or truncated Fourier series, the error typically satisfies

$$\|f - f_m\|_{L^2(X)} \leq C_0 m^{-\alpha/d} \|f\|_{H^\alpha(X)}$$

where H^α denotes the Sobolev space of order α. The appearance of $1/d$ in the exponent of m is a signature of an important phenomenon, the **curse of dimensionality (CoD)**: The number of parameters required to achieve certain accuracy depends exponentially on the dimension. For example, if we want $m^{-\alpha/d} = 0.1$, then we need $m = 10^{d/\alpha} = 10^d$, if $\alpha = 1$.

At this point, it is useful to recall the one problem that has been extensively studied in high dimension: the problem of evaluating an integral, or more precisely, computing an expectation. Let g be a function defined on X. We are interested in computing approximately

$$I(g) = \int_X g(\boldsymbol{x}) d\boldsymbol{x} = \mathbb{E} g$$

where the expectation is taken with respect to the uniform distribution. Typical grid-based quadrature rules, such as the Trapezoidal rule and the Simpson's rule, all suffer from CoD. The one algorithm that does not suffer from CoD is the Monte Carlo algorithm which works as follows. Let $\{\boldsymbol{x}_i\}_{i=1}^n$ be a set of independent, uniformly distributed random variables on X. Let

$$I_n(g) = \frac{1}{n} \sum_{i=1}^n g(\boldsymbol{x}_i)$$

Then a simple calculation gives us

$$\mathbb{E}(I(g) - I_n(g))^2 = \frac{\mathrm{Var}(g)}{n}, \quad \mathrm{Var}(g) = \int_X g^2(\boldsymbol{x})d\boldsymbol{x} - \left(\int_X g(\boldsymbol{x})d\boldsymbol{x}\right)^2 \tag{1}$$

The $O(1/\sqrt{n})$ rate is independent of d. It turns out that this rate is almost the best one can hope for.

In practice, $\mathrm{Var}(g)$ can be very large in high dimension. Therefore, variance reduction techniques are crucial in order to make Monte Carlo methods truly practical.

Turning now to the estimation error. Our concern is how the approximation produced by the machine learning algorithm behaves away from the training dataset, or in practical terms, whether the training and test errors are close. Shown in Figure 1 is the classical Runge phenomenon for interpolating functions on uniform grids using high order polynomials. One can see that while on the training set, here the grid points, the error of the interpolant is 0, away from the training set, the error can be very large.

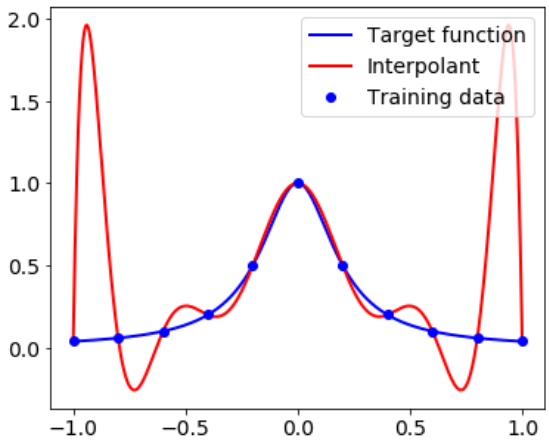


Figure 1: The Runge phenomenon with target function $f^*(x) = (1 + 25\,x^2)^{-1}$.

It is often easier to study a related quantity, the generalization gap. Consider the solution that minimizes the empirical risk, $\hat{f} = \mathrm{argmin}_{f\in\mathcal{H}_m} \hat{\mathcal{R}}_n(f)$. The "generalization gap" of $\hat{f}$ is the quantity $|\mathcal{R}(\hat{f}) - \hat{\mathcal{R}}_n(\hat{f})|$. Since it is equal to $|I(g) - I_n(g)|$ with $g(\boldsymbol{x}) = (\hat{f}(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2$, one might be tempted to conclude that

$$\text{generalization gap} = O(1/\sqrt{n})$$

based on (1). This is NOT necessarily true since $\hat{f}$ is highly correlated with $\{\boldsymbol{x}_i\}$. In fact, controlling this gap is among the most difficult problems in ML.

Studying the correlations of $\hat{f}$ is a rather impossible problem. Therefore to estimate the generalization gap, we resort to the uniform bound:

$$|\mathcal{R}(\hat{f}) - \hat{\mathcal{R}}_n(\hat{f})| \le \sup_{f\in\mathcal{H}_m} |\mathcal{R}(f) - \hat{\mathcal{R}}_n(f)| = \sup_{f\in\mathcal{H}_m} |I(g) - I_n(g)| \tag{2}$$

The RHS of this equation depends heavily on the nature of $\mathcal{H}_m$. If we take $\mathcal{H}_m$ to be the unit ball in the Lipschitz space, we have [39]

$$\sup_{\|h\|_{Lip}\leq 1} |I(g) - I_n(g)| \sim \frac{1}{n^{1/d}}$$

with $g = (h - f^*)^2$. This gives rise to **CoD for the size of the dataset** (commonly referred to as "sample complexity"). However if we take $\mathcal{H}_m$ to be the unit ball in the Barron space, to be defined later, we have

$$\sup_{\|h\|_{\mathcal{B}}\leq 1} |I(h) - I_n(h)| \sim \frac{1}{\sqrt{n}}$$

This is the kind of estimates that we should look for.

Assuming that all the functions under consideration are bounded, the problem of estimating the RHS of (2) reduces to the estimation of $\sup_{h\in\mathcal{H}_m} |I(h) - I_n(h)|$. One way to do this is to use the notion of Rademacher complexity [15].

Definition 1. Let $\mathcal{F}$ be a set of functions, and $S = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)$ be a set of data points. Then, the Rademacher complexity of $\mathcal{F}$ with respect to S is defined as

$$\mathrm{Rad}_S(\mathcal{F}) = \frac{1}{n}\mathbb{E}_\xi \left[\sup_{h\in\mathcal{F}} \sum_{i=1}^n \xi_i h(\boldsymbol{x}_i)\right],$$

where $\{\xi_i\}_{i=1}^n$ are i.i.d. random variables taking values ± 1 with equal probability.

Roughly speaking, Rademacher complexity quantifies the degree to which functions in the hypothesis space can fit random noise on the given dataset. It bounds the quantity of interest, $\sup_{h\in\mathcal{H}} |I(h) - I_n(h)|$, from above and below.

Theorem 2. *For any $\delta \in (0, 1)$, with probability at least $1 - \delta$ over the random samples $S = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)$, we have*

$$\sup_{h\in\mathcal{F}} \left| \mathbb{E}_{\boldsymbol{x}} [h(\boldsymbol{x})] - \frac{1}{n}\sum_{i=1}^n h(\boldsymbol{x}_i) \right| \leq 2\,\mathrm{Rad}_S(\mathcal{F}) + \sup_{h\in\mathcal{F}} \|h\|_\infty \sqrt{\frac{\log(2/\delta)}{2n}}.$$

$$\sup_{h\in\mathcal{F}} \left| \mathbb{E}_{\boldsymbol{x}} [h(\boldsymbol{x})] - \frac{1}{n}\sum_{i=1}^n h(\boldsymbol{x}_i) \right| \geq \frac{1}{2}\,\mathrm{Rad}_S(\mathcal{F}) - \sup_{h\in\mathcal{F}} \|h\|_\infty \sqrt{\frac{\log(2/\delta)}{2n}}.$$

For a proof, see for example [76, Theorem 26.5]. For this reason, a very important part of theoretical machine learning is to study the Rademacher complexity of a given hypothesis space.

It should be noted that there are other ways of analyzing the generalization gap, such as the stability method [17].

Notations. For any function $f : \mathbb{R}^m \mapsto \mathbb{R}^n$, let $\nabla f = (\frac{\partial f_i}{\partial x_j})_{i,j} \in \mathbb{R}^{n\times m}$ and $\nabla^T f = (\nabla f)^T$. We use $X \lesssim Y$ to mean that $X \leq CY$ for some absolute constant C. For any $\boldsymbol{x} \in \mathbb{R}^d$, let $\tilde{\boldsymbol{x}} = (\boldsymbol{x}^T, 1)^T \in \mathbb{R}^{d+1}$. Let Ω be a subset of $\mathbb{R}^d$, and denote by $\mathcal{P}(\Omega)$ the space of probability measures. Define $\mathcal{P}_2(\Omega) = \{\mu \in \mathcal{P}(\Omega) : \int \|\boldsymbol{x}\|_2^2 d\mu(\boldsymbol{x}) < \infty\}$. We will also follow the convention in probability theory to use ρ_t to denote the value of ρ at time t.

2 Preliminary remarks

In this section we set the stage for our discussion by going over some of the classical results as well as recent qualitative studies that are of general interest.

2.1 Universal approximation theorem and the CoD

Let us consider the two-layer neural network hypothesis space:

$$\mathcal{H}_m = \{f_m(\boldsymbol{x}) = \sum_{j=1}^{m} a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})\}$$

where σ is a nonlinear function, the activation function. The Universal Approximation Theorem (UAT) states that under very mild conditions, any continuous function can be uniformly approximated on compact domains by neural network functions.

Theorem 3. *[24, Theorem 5] If σ is sigmoidal, in the sense that $\lim_{z\to-\infty}\sigma(z) = 0, \lim_{z\to\infty}\sigma(z) = 1$, then any function in $C([0,1]^d)$ can be approximated uniformly by two layer neural network functions.*

This result can be extended to any activation functions that are not exactly a polynomial [56].

UAT plays the role of Weierstrass Theorem on the approximation of continuous functions by polynomials. It is obviously an important result, but it is of limited use due to the lack of quantitative information about the error in the approximation. For one thing, the same conclusion can be drawn for polynomial approximation, which we know is of limited use in high dimension.

Many quantitative error estimates have been established since then. But most of these estimates suffer from CoD. The one result that stands out is the estimate proved by Barron [11]:

$$\inf_{f_m\in\mathcal{H}_m} \|f^* - f_m\|_{L^2(P)}^2 \lesssim \frac{\Delta(f^*)^2}{m} \tag{3}$$

where $\Delta(f)$ is a norm defined by

$$\Delta(f) := \inf_{\hat{f}} \int_{\mathbb{R}^d} \|\omega\|_1 \, |\hat{f}(\omega)| d\omega < \infty,$$

where $\hat{f}$ is the Fourier transform of an extension of f to $L^2(\mathbb{R}^d)$. The convergence rate in (3) is independent of dimension. However, the constant $\Delta(f^*)$ could be dimension-dependent since it makes use of the Fourier transform. As the dimensionality goes up, the number of derivatives required to make Δ finite also goes up. This is distinct from the CoD, which is the dimension-dependence of the rate.

2.2 The loss landscape of large neural network models

The landscape of the loss function for large neural networks was studied in [22] using an analogy with high dimensional spherical spin glasses and numerical experiments. The landscape of high dimensional spherical spin glass models has been analyzed in [6]. It was shown that the lowest critical values of the Hamiltonians of these models form a layered structure, ordered by the indices of the critical points. They are located in a well-defined band lower-bounded by the global minimum. The probability of finding critical points outside the band diminishes exponentially with the dimension of the spin-glass model. Choromanska et al [22] used the observation that neural networks with independent weights can be mapped onto spherical spin glass models, and suggested that the same picture should hold qualitatively for large neural network models. They provided numerical evidence to support this suggestion.

This work is among the earliest that suggests even though it is highly non-convex, the landscape of larger neural networks might be simpler than the ones for smaller networks.

2.3 Over-parametrization, interpolation and implicit regularization

Modern deep learning often works in the over-parametrized regime where the number of free parameters is larger than the size of the training dataset. This is a new territory as far as machine learning theory is concerned. Conventional wisdom would suggest that one should expect overfitting in this regime, i.e. an increase in the generalization gap. Whether overfitting really happens is a problem of great interest in theoretical machine learning. As we will see later, one can show that, overfitting does happen in the highly over-parametrized regime.

An enlightening numerical study of the optimization and generalization properties in this regime was carried out in [91]. Among other things, it was discovered that in this regime, the neural networks are so expressive that they can fit any data, no matter how much noise one adds to the data. Later it was shown by Cooper that the set of global minima with no training error forms a manifold of dimension $m - n$ [23].

Some of these global minima generalize very poorly. Therefore an important question is how to select the ones that do generalize well. It was suggested in [91] that by tuning the hyper-parameters of the optimization algorithm, one can obtain models with good generalization properties without adding explicit regularization. This means that the training dynamics itself has some implicit regularization mechanism which ensures that bad global minima are not selected during training. Understanding the possible mechanism for this

implicit regularization is one of the key issues in understanding modern machine learning.

2.4 The selection of topics

There are several rather distinct ways for a mathematical analysis of machine learning and particularly neural network models:

1. The numerical analysis perspective. Basically machine learning problems are viewed as (continuous) function approximation and optimization problems, typically in high dimension.

2. The harmonic analysis perspective. Deep learning is studied from the viewpoint of building hierarchical wavelets-like transforms. Examples of such an approach can be found in [18, 83].

3. The statistical physics perspective. A particularly important tool is the replica trick. For special kinds of problems, this approach allows us to perform asymptotically exact (hence sharp) calculations. It is also useful to study the performance of models and algorithms from the viewpoint of phase transitions. See for example [90, 9, 40].

4. The information theory perspective. See [1] for an example of the kinds of results obtained along this line.

5. The PAC learning theory perspective. This is closely related to the information theory perspective. It studies machine learning from the viewpoint of complexity theory. See for example [61].

It is not yet clear how the different approaches are connected, even though many results may fall in several categories. In this review, we will only cover the results obtained along the lines of numerical analysis. We encourage the reader to consult the papers referenced above to get a taste of these alternative perspectives.

Within the numerical analysis perspective, supervised machine learning and neural network models are still vast topics. By necessity, this article focusses on a few aspects which we believe to be key problems in machine learning. As a model problem, we focus on L^2-regression here, where the data is assumed to be of the form $(\boldsymbol{x}_i, f^*(\boldsymbol{x}_i))$ without uncertainty in the y-direction.

In Section 3, we focus on the function spaces developed for neural network models. Section 4 discusses the (very short) list of results available for the energy landscape of loss functionals in machine learning. Training dynamics for network weights are discussed in Section 5 with a focus on gradient descent. The specific topics are selected for two criteria:

- We believe that they have substantial importance for the mathematical understanding of machine learning.

- We are reasonably confident that the mathematical models developed so far will over time find their way into the standard language of the theoretical machine learning community.

There are many glaring omissions in this article, among them:

1. Classification problems. While most benchmark problems for neural networks fall into the framework of classification, we focus on the more well-studied area of regression.

2. Some other common neural network architectures, such as convolutional neural networks, long short-term memory (LSTM) networks, encoder-decoder networks.

3. The impact of stochasticity. Many training algorithms and initialization schemes for neural networks use random variables. While toy models with standard Gaussian noise are well understood, the realistic case remains out of reach [45].

4. Simplified neural networks such as linear and quadratic networks. While these models are not relevant for applications, the simpler model allows for simpler analysis. Some insight is available for these models which has not been achieved in the non-linear case [75, 49, 4].

5. Important "tricks", such as dropout, batch normalization, layer normalization, gradient clipping, etc. To our knowledge, these remain empirically useful, but mysterious from a mathematical perspective.

6. Highly data-dependent results. The geometry of data is a large field which we avoid in this context. While many data-distributions seem to be concentrated close to relatively low-dimensional manifolds in a much higher-dimensional ambient space, these 'data-manifolds' are in many cases high-dimensional enough that CoD, a central theme in this review, is still an important issue.

3 The approximation property and the Rademacher complexity of the hypothesis space

The most important issue in classical approximation theory is to identify the function space naturally associated with a particular approximation scheme, e.g. approximation by piecewise polynomials on regular grids. These spaces are typically some Sobolev or Besov spaces, or their variants. They are the natural spaces for the particular approximation scheme, since one can prove matching direct and inverse approximation theorems, namely any function in the space can be approximated using the given approximation scheme with the specified rate of convergence, and conversely any function that can be approximated to the specified order of accuracy belongs to that function space.

Machine learning is just another way to approximate functions, therefore we can ask similar questions, except that our main interest in this case is in high dimension. Any

machine learning model hits the 'curse of dimensionality' when approximating the class of Lipschitz functions. Nevertheless, many important problems seem to admit accurate approximations by neural networks. Therefore it is important to understand the class of functions that can be well approximated by a particular machine learning model.

There is one important difference from the classical setting. In high dimension, the rate of convergence is limited to the Monte Carlo rate and its variants. There is limited room regarding order of convergence and consequently there is no such thing as the order of the space as is the case for Sobolev spaces.

Ideally, we would like to accomplish the following:

1. Given a type of hypothesis space $\mathcal{H}_m$, say two-layer neural networks, identify the natural function space associated with them (in particular, identify a norm $\|f^*\|_*$) that satisfies:

   - Direct approximation theorem:

   $$\inf_{f\in\mathcal{H}_m} \mathcal{R}(f) = \inf_{f\in\mathcal{H}_m} \|f - f^*\|^2_{L^2(P)} \lesssim \frac{\|f^*\|^2_*}{m}$$

   - Inverse approximation theorem: If a function f^* can be approximated efficiently by the functions in $\mathcal{H}_m$, as $m \to \infty$ with some uniform bounds, then $\|f^*\|_*$ is finite.

2. Study the generalization gap for this function space. One way of doing this is to study the Rademacher complexity of the set $\mathcal{F}_Q = \{f : \|f\|_* \leq Q\}$. Ideally, we would like to have:

   $$\mathrm{Rad}_S(\mathcal{F}_Q) \lesssim \frac{Q}{\sqrt{n}}$$

If both holds, then a combination gives us, up to logarithmic terms:

$$\mathcal{R}(\hat{f}) \lesssim \frac{\|f^*\|^2_*}{m} + \frac{\|f^*\|_*}{\sqrt{n}}$$

Remark 4. It should be noted that what we are really interested in is the quantitative measures of the target function that control the approximation and estimation errors. We call these quantities "norms" but we are not going to insist that they are really norms. In addition, we would like to use one norm to control both the approximation and estimation errors. This way we have one function space that meets both requirements. However, it could very well be the case that we need different quantities to control different errors. See the discussion about residual networks below. This means that we will be content with a generalized version of (3):

$$\mathcal{R}(\hat{f}) \lesssim \frac{\Gamma(f^*)}{m} + \frac{\gamma(f^*)}{\sqrt{n}}$$

We will see that this can indeed be done for the most popular neural network models.

3.1 Random feature model

Let $\phi(\cdot;\boldsymbol{w})$ be the feature function parametrized by $\boldsymbol{w}$, e.g. $\phi(\boldsymbol{x};\boldsymbol{w}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$. A random feature model is given by

$$f_m(\boldsymbol{x};\boldsymbol{a}) = \frac{1}{m}\sum_{j=1}^{m} a_j\phi(\boldsymbol{x};\boldsymbol{w}_j^0). \tag{4}$$

where $\{\boldsymbol{w}_j^0\}_{j=1}^m$ are i.i.d random variables drawn from a prefixed distribution π_0. The collection $\{\phi(\cdot;\boldsymbol{w}_j^0)\}$ are the random features, $\boldsymbol{a} = (a_1,\ldots,a_m)^T \in \mathbb{R}^m$ are the coefficients. For this model, the natural function space is the reproducing kernel Hilbert space (RKHS) [3] induced by the kernel

$$k(\boldsymbol{x},\boldsymbol{x}') = \mathbb{E}_{\boldsymbol{w}\sim\pi_0}[\phi(\boldsymbol{x};\boldsymbol{w})\phi(\boldsymbol{x}';\boldsymbol{w})] \tag{5}$$

Denote by $\mathcal{H}_k$ this RKHS. Then for any $f \in \mathcal{H}_k$, there exists $a(\cdot) \in L^2(\pi_0)$ such that

$$f(\boldsymbol{x}) = \int a(\boldsymbol{w})\phi(\boldsymbol{x};\boldsymbol{w})d\pi_0(\boldsymbol{w}), \tag{6}$$

and

$$\|f\|_{\mathcal{H}_k}^2 = \inf_{a\in S_f}\int a^2(\boldsymbol{w})d\pi_0(\boldsymbol{w}), \tag{7}$$

where $S_f := \{a(\cdot) : f(\boldsymbol{x}) = \int a(\boldsymbol{w})\phi(\boldsymbol{x};\boldsymbol{w})d\pi_0(\boldsymbol{w})\}$. We also define $\|f\|_\infty = \inf_{a\in S_f}\|a(\cdot)\|_{L^\infty(\pi_0)}$. For simplicity, we assume that $\Omega := \mathrm{supp}(\pi_0)$ is compact. Denote $\boldsymbol{W}^0 = (\boldsymbol{w}_1^0,\ldots,\boldsymbol{w}_m^0)^T \in \mathbb{R}^{m\times d}$ and $a(\boldsymbol{W}^0) = (a(\boldsymbol{w}_1^0),\ldots,a(\boldsymbol{w}_m^0))^T \in \mathbb{R}^m$.

Theorem 5 (Direct Approximation Theorem)**.** *Assume $f^* \in \mathcal{H}_k$, then there exists $a(\cdot)$, such that*

$$\mathbb{E}_{\boldsymbol{W}^0}[\|f_m(\cdot;a(\boldsymbol{W}^0)) - f^*)\|_{L^2}^2] \le \frac{\|f^*\|_{\mathcal{H}_k}^2}{m}.$$

Theorem 6 (Inverse Approximation Theorem)**.** *Let $(\boldsymbol{w}_j^0)_{j=0}^\infty$ be a sequence of i.i.d. random variables drawn from π_0. Let f^* be a continuous function on X. Assume that there exist constants C and a sequence $(a_j)_{j=0}^\infty$ satisfying $\sup_j|a_j| \le C$, such that*

$$\lim_{m\to\infty}\frac{1}{m}\sum_{j=1}^{m} a_j\phi(\boldsymbol{x};\boldsymbol{w}_j^0) = f^*(\boldsymbol{x}), \tag{8}$$

for all $\boldsymbol{x} \in X$. Then with probability 1, there exists a function $a^*(\cdot) : \Omega \mapsto \mathbb{R}$ such that

$$f^*(\boldsymbol{x}) = \int_\Omega a^*(\boldsymbol{w})\phi(\boldsymbol{x};\boldsymbol{w})d\pi_0(\boldsymbol{w}),$$

Moreover, $\|f\|_\infty \le C$.

To see how these approximation theory results can play out in a realistic ML setting, consider the regularized model:

$$\mathcal{L}_{n,\lambda}(\boldsymbol{a}) = \hat{\mathcal{R}}_n(\boldsymbol{a}) + \frac{1}{\sqrt{n}}\frac{\|\boldsymbol{a}\|}{\sqrt{m}},$$

and define the regularized estimator:

$$\hat{\boldsymbol{a}}_{n,\lambda} = \operatorname{argmin} \mathcal{L}_{n,\lambda}(\boldsymbol{a}).$$

Theorem 7. *For any $\delta \in (0,1)$, with probability $1-\delta$, the population risk of the regularized estimator satisfies*

$$\mathcal{R}(\hat{\boldsymbol{a}}_n) \le \frac{1}{m}\left(\log(n/\delta)\|f^*\|_{\mathcal{H}_k}^2 + \frac{\log^2(n/\delta)}{m}\|f^*\|_\infty^2\right) \tag{9}$$

$$+ \frac{1}{\sqrt{n}}\left(\|f^*\|_{\mathcal{H}_k} + \left(\frac{\log(1/\delta)}{m}\right)^{1/4}\|f^*\|_\infty + \sqrt{\log(2/\delta)}\right). \tag{10}$$

These results should be standard. However, they do not seem to be available in the literature. In the appendix, we provide a proof for these results.

It is worth noting that the dependence on $\|f\|_\infty$ and $\log(n/\delta)$ can be removed by a more sophisticated analysis [8]. However, to achieve the rate of $O(1/m + 1/\sqrt{n})$, one must make an explicit assumption on the decay rate of eigenvalues of the corresponding kernel operator [8].

3.2 Two-layer neural network model

The hypothesis space for two-layer neural networks is defined by:

$$\mathcal{F}_m = \{f_m(\boldsymbol{x}) = \frac{1}{m}\sum_{j=1}^m a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})\}$$

We will focus on the case when the activation function σ is ReLU: $\sigma(z) = \max(z, 0)$. Many of the results discussed below can be extended to more general activation functions [60].

The function space for this model is called the Barron space ([34, 31], see also [11, 53, 37] and particularly [7]). Consider the function $f : X = [0,1]^d \mapsto \mathbb{R}$ of the following form

$$f(\boldsymbol{x}) = \int_\Omega a\sigma(\boldsymbol{w}^T\boldsymbol{x})\rho(da, d\boldsymbol{w}) = \mathbb{E}_\rho[a\sigma(\boldsymbol{w}^T\boldsymbol{x})], \quad \boldsymbol{x} \in X$$

where $\Omega = \mathbb{R}^1 \times \mathbb{R}^{d+1}$, ρ is a probability distribution on Ω. The Barron norm is defined by

$$\|f\|_{\mathcal{B}_p} = \inf_{\rho \in P_f} \left(\mathbb{E}_\rho[a^p \|\boldsymbol{w}\|_1^p]\right)^{1/p}$$

where $P_f := \{\rho : f(\boldsymbol{x}) = \mathbb{E}_\rho[a\sigma(\boldsymbol{w}^T\boldsymbol{x})]\}$. Define

$$\mathcal{B}_p = \{f \in C^0 : \|f\|_{\mathcal{B}_p} < \infty\}$$

Functions in $\mathcal{B}_p$ are called Barron functions. As shown in [31], for the ReLU activation function, we actually have $\|\cdot\|_{\mathcal{B}_p} = \|\cdot\|_{\mathcal{B}_q}$ for any $1 \le p \le q \le \infty$. Hence, we will use $\|\cdot\|_{\mathcal{B}}$ and $\mathcal{B}$ denote the Barron norm and Barron space, respectively.

Remark 8. Barron space and Barron functions are named in reference to the article [11] which was the first to recognize and rigorously establish the advantages of non-linear approximation over linear approximation by considering neural networks with a single hidden layer.

It should be stressed that the Barron norm introduced above is not the same as the one used in [11], which was based on the Fourier transform (see (11)). To highlight this distinction, we will call the kind of norm in (11) *spectral norm*.

An important property of the ReLU activation function is the homogeneity property $\sigma(\lambda z) = \lambda\,\sigma(z)$ for all $\lambda > 0$. A discussion of the representation for Barron functions with partial attention to homogeneity can be found in [37]. Barron spaces for other activation functions are discussed in [35, 60].

One natural question is what kind of functions are Barron functions. The following result gives a partial answer.

Theorem 9 (Barron and Klusowski (2016))**.** *If*

$$\Delta(f) := \int_{\mathbb{R}^d} \|\omega\|_1^2 |\hat{f}(\omega)| d\omega < \infty, \tag{11}$$

where $\hat{f}$ is the Fourier transform of f, then f can be represented as

$$f(\boldsymbol{x}) = \int_\Omega a\sigma(\boldsymbol{b}^T\boldsymbol{x} + c)\rho(da, d\boldsymbol{b}, dc), \quad \forall \boldsymbol{x} \in X$$

where $\sigma(x) = \max(0, x)$. Moreover $\|f\|_{\mathcal{B}} \le 2\Delta(f) + 2\|\nabla f(0)\|_1 + 2f(0)$.

A consequence of this is that every function in $H^s(\mathbb{R}^d)$ is Barron for $s > \frac{d}{2}+1$. In addition, it is obvious that every finite sum of neuron activations is also a Barron function. Another interesting example of a Barron function is the function $f(\boldsymbol{x}) = \|x\|_{\ell^2} = \mathbb{E}_{\boldsymbol{b}\sim\mathcal{N}(0,I_d)}[\sigma(\boldsymbol{b}^T\boldsymbol{x})]$.

On the other hand, every Barron function is Lipschitz-continuous. An important criterion to establish that certain functions are *not* in Barron space is the following structure theorem.

Theorem 10. *[37, Theorem 5.4] Let f be a Barron function. Then $f = \sum_{i=1}^\infty f_i$ where $f_i \in C^1(\mathbb{R}^d \setminus V_i)$ where V_i is a k-dimensional affine subspace of $\mathbb{R}^d$ for some k with $0 \le k \le d-1$.*

As a consequence, distance functions to curved surfaces are not Barron functions.

The claim that the Barron space is the natural space associated with two-layer networks is justified by the following series of results.

Theorem 11 (Direct Approximation Theorem, L^2-version)**.** *For any $f \in \mathcal{B}$ and $m \in \mathbb{N}$, there exists a two-layer neural network f_m with m neurons $\{(a_j, \boldsymbol{w}_j)\}_{j=1}^m$ such that*

$$\|f - f_m\|_{L^2(P)} \lesssim \frac{\|f\|_{\mathcal{B}}}{\sqrt{m}}.$$

Theorem 12 (Direct Approximation Theorem, L^∞-version)**.** *For any $f \in \mathcal{B}$ and $m \in \mathbb{N}$, there exists a two-layer neural network f_m with m neurons $\{(a_j, \boldsymbol{w}_j)\}_{j=1}^m$ such that*

$$\|f - f_m\|_{L^\infty([0,1]^d)} \lesssim 4\,\|f\|_{\mathcal{B}} \sqrt{\frac{d+1}{m}}.$$

We present a brief self-contained proof of the L^∞-direct approximation theorem in the appendix. We believe the idea to be standard, but have been unable to locate a good reference for it.

Remark 13. In fact, there exists a constant $C > 0$ such that

$$\|f - f_m\|_{L^\infty([0,1]^d)} \leq C\,\|f\|_{\mathcal{B}}\,\frac{\sqrt{\log(m)}}{m^{1/2+1/2d}}$$

and for every $\varepsilon > 0$ there exists $f \in \mathcal{B}$ such that

$$\|f - f_m\|_{L^\infty([0,1]^d)} \geq c\,m^{-1/2-1/d-\varepsilon},$$

see [10, 67]. Further approximation results, including in classical functions spaces, can be found in [72].

Theorem 14 (Inverse Approximation Theorem)**.** *Let*

$$\mathcal{N}_C \stackrel{def}{=} \{\, \frac{1}{m}\sum_{j=1}^m a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x}) \;:\; \frac{1}{m}\sum_{j=1}^m |a_j| \|\boldsymbol{w}_j\|_1 \leq C, m \in \mathbb{N}^+ \,\}.$$

Let f^* be a continuous function. Assume there exists a constant C and a sequence of functions $f_m \in \mathcal{N}_C$ such that

$$f_m(\boldsymbol{x}) \to f^*(\boldsymbol{x})$$

for all $\boldsymbol{x} \in X$, then there exists a probability distribution ρ^* on Ω, such that

$$f^*(\boldsymbol{x}) = \int a\sigma(\boldsymbol{w}^T \boldsymbol{x})\rho^*(da, d\boldsymbol{w}),$$

for all $\boldsymbol{x} \in X$ and $\|f^*\|_{\mathcal{B}} \leq C$.

Both theorems are proved in [34]. In addition, it just so happens that the Rademacher complexity is also controlled by a Monte Carlo like rate:

Theorem 15 ([7]). *Let $\mathcal{F}_Q = \{f \in \mathcal{B}, \|f\|_{\mathcal{B}} \leq Q\}$. Then we have*

$$\mathrm{Rad}_S(\mathcal{F}_Q) \leq 2Q\sqrt{\frac{2\ln(2d)}{n}}$$

In the same way as before, one can now consider the regularized model:

$$\mathcal{L}_n(\theta) = \hat{\mathcal{R}}_n(\theta) + \lambda\sqrt{\frac{\log(2d)}{n}}\|\theta\|_{\mathcal{P}}, \qquad \hat{\theta}_n = \operatorname{argmin} \mathcal{L}_n(\theta)$$

where the path norm is defined by:

$$\|\theta\|_{\mathcal{P}} = \frac{1}{m}\sum_{j=1}^{m} |a_j| \|\boldsymbol{w}_j\|_1$$

Theorem 16. *[34]: Assume $f^* : X \mapsto [0,1] \in \mathcal{B}$. There exist constants absolute C_0, such that for any $\delta > 0$, if $\lambda \geq C_0$, then with probability at least $1-\delta$ over the choice of the training set, we have*

$$\mathcal{R}(\hat{\theta}_n) \lesssim \frac{\|f^*\|_{\mathcal{B}}^2}{m} + \lambda\|f^*\|_{\mathcal{B}}\sqrt{\frac{\log(2d)}{n}} + \sqrt{\frac{\log(n/\delta)}{n}}.$$

3.3 Residual networks

Consider a residual network model

$$\begin{aligned}
\boldsymbol{z}_{0,L}(\boldsymbol{x}) &= \boldsymbol{V}\boldsymbol{x}, \\
\boldsymbol{z}_{l+1,L}(\boldsymbol{x}) &= z_{l,L}(\boldsymbol{x}) + \frac{1}{L}\boldsymbol{U}_l\sigma \circ (\boldsymbol{W}_l z_{l,L}(\boldsymbol{x})), \quad l = 0, 1, \cdots, L-1 \\
f(\boldsymbol{x}, \theta) &= \boldsymbol{\alpha}^T \boldsymbol{z}_{L,L}(\boldsymbol{x}),
\end{aligned}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the input, $\boldsymbol{V} \in \mathbb{R}^{D\times d}, \boldsymbol{W}_l \in \mathbb{R}^{m\times D}$, $\boldsymbol{U}_l \in \mathbb{R}^{D\times m}, \boldsymbol{\alpha} \in \mathbb{R}^D$. Without loss of generality, we will fix $\boldsymbol{V}$ to be

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{I}_{d\times d} \\ \boldsymbol{0}_{(D-d)\times d} \end{bmatrix}. \tag{12}$$

We use $\Theta := \{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_L, \boldsymbol{W}_l, \ldots, \boldsymbol{W}_L, \boldsymbol{\alpha}\}$ to denote all the parameters to be learned from data.

Consider the following ODE system [31]:

$$\begin{aligned}
\boldsymbol{z}(\boldsymbol{x}, 0) &= \boldsymbol{V}\boldsymbol{x}, \\
\dot{\boldsymbol{z}}(\boldsymbol{x}, t) &= \mathbb{E}_{(\boldsymbol{U},\boldsymbol{W})\sim\rho_t}\boldsymbol{U}\sigma(\boldsymbol{W}\boldsymbol{z}(\boldsymbol{x}, t)), \\
f_{\boldsymbol{\alpha},\{\rho_t\}}(\boldsymbol{x}) &= \boldsymbol{\alpha}^T \boldsymbol{z}(\boldsymbol{x}, 1).
\end{aligned}$$

This ODE system can be viewed as the limit of the residual network (12) ([31]). Consider the following linear ODEs ($p \geq 1$)

$$\begin{aligned} N_p(0) &= \mathbf{1}, \\ \dot{N}_p(t) &= (\mathbb{E}_{\rho_t}(|\boldsymbol{U}||\boldsymbol{W}|)^p)^{1/p} N_p(t), \end{aligned}$$

where $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^d$, $|\mathbf{A}|$ and $\mathbf{A}^q$ are element-wise operations for the matrix $\mathbf{A}$ and positive number q.

Definition 17 ([31])**.** Let f be a function that admits the form $f = f_{\boldsymbol{\alpha},\{\rho_t\}}$ for a pair of $(\boldsymbol{\alpha}, \{\rho_t\})$, then we define

$$\|f\|_{\mathcal{D}_p(\boldsymbol{\alpha},\{\rho_t\})} = |\boldsymbol{\alpha}|^T N_p(1), \tag{13}$$

to be the $\mathcal{D}_p$ norm of f with respect to the pair $(\boldsymbol{\alpha}, \{\rho_t\})$. Here $|\boldsymbol{\alpha}|$ is obtained from $\boldsymbol{\alpha}$ by taking element-wise absolute values. We define

$$\|f\|_{\mathcal{D}_p} = \inf_{f=f_{\boldsymbol{\alpha},\{\rho_t\}}} |\boldsymbol{\alpha}|^T N_p(1). \tag{14}$$

to be the $\mathcal{D}_p$ norm of f, and let $\mathcal{D}_p = \{f : \|f\|_{\mathcal{D}_p} < \infty\}$ be the flow-induced function space.

Remark 18. In our definition, ρ_t is a distribution on $\mathbb{R}^{D\times m} \times \mathbb{R}^{m\times D}$. However, the ODE system (13) with large m does not express more functions than just taking $m = 1$. Nevertheless, choosing $m \neq 1$ may influence the associated function norm as well as the constant on the approximation bound in Theorem 23. We do not explore the effect of different m in this paper.

Beside $\mathcal{D}_p$, [31] introduced another class of function spaces $\tilde{\mathcal{D}}_p$ that contain functions for which the quantity $N_p(1)$ and the continuity of ρ_t with respect to t are controlled. We first provide the following definition of "Lipschitz condition" of ρ_t.

Definition 19. Given a family of probability distribution $\{\rho_t,\ t \in [0, 1]\}$, the "Lipschitz coefficient" of $\{\rho_t\}$, denoted by $Lip_{\{\rho_t\}}$, is defined as the infimum of all the numbers L that satisfies

$$|\mathbb{E}_{\rho_t} \boldsymbol{U}\sigma(\boldsymbol{W}\boldsymbol{z}) - \mathbb{E}_{\rho_s} \boldsymbol{U}\sigma(\boldsymbol{W}\boldsymbol{z})| \leq L|t - s||\boldsymbol{z}|, \tag{15}$$

and

$$\left| \|\mathbb{E}_{\rho_t}|\boldsymbol{U}||\boldsymbol{W}|\|_{1,1} - \|\mathbb{E}_{\rho_s}|\boldsymbol{U}||\boldsymbol{W}|\|_{1,1} \right| \leq L|t - s|, \tag{16}$$

for any $t, s \in [0, 1]$, where $\|\cdot\|_{1,1}$ is the sum of the absolute values of all the entries in a matrix. The "Lipschitz norm" of $\{\rho_t\}$ is defined as

$$\|\{\rho_t\}\|_{Lip} = \|\mathbb{E}_{\rho_0}|\boldsymbol{U}||\boldsymbol{W}|\|_{1,1} + Lip_{\{\rho_t\}}. \tag{17}$$

Definition 20 ([31])**.** Let f be a function that satisfies $f = f_{\boldsymbol{\alpha},\{\rho_t\}}$ for a pair of $(\boldsymbol{\alpha}, \{\rho_t\})$, then we define

$$\|f\|_{\tilde{\mathcal{D}}_p(\boldsymbol{\alpha},\{\rho_t\})} = |\boldsymbol{\alpha}|^T N_p(1) + \|N_p(1)\|_1 - D + \|\{\rho_t\}\|_{Lip}, \tag{18}$$

to be the $\tilde{\mathcal{D}}_p$ norm of f with respect to the pair $(\boldsymbol{\alpha}, \{\rho_t\})$. We define

$$\|f\|_{\tilde{\mathcal{D}}_p} = \inf_{f=f_{\boldsymbol{\alpha},\{\rho_t\}}} \|f\|_{\tilde{\mathcal{D}}_p(\boldsymbol{\alpha},\{\rho_t\})}. \tag{19}$$

to be the $\tilde{\mathcal{D}}_p$ norm of f. The space $\tilde{\mathcal{D}}_p$ is defined as all the functions that admit the representation $f_{\boldsymbol{\alpha},\{\rho_t\}}$ with finite $\tilde{\mathcal{D}}_p$ norm.

The space $\mathcal{D}_p$ and $\tilde{\mathcal{D}}_p$ are called *flow-induced spaces*.

One can easily see that the "norms" defined here are all non-negative quantities (despite the $-D$ term), even though it is not clear that they are really norms. The following embedding theorem shows that flow-induced function space is larger than Barron space.

Theorem 21. *For any function $f \in \mathcal{B}$, and $D \geq d+2$ and $m \geq 1$, we have $f \in \tilde{\mathcal{D}}_1$, and*

$$\|f\|_{\tilde{\mathcal{D}}_1} \leq 2\|f\|_{\mathcal{B}} + 1. \tag{20}$$

Finally, we define a discrete "path norm" for residual networks.

Definition 22. For a residual network defined by (12) with parameters $\Theta = \{\boldsymbol{\alpha}, \boldsymbol{U}_l, \boldsymbol{W}_l, l = 0, 1, \cdots, L-1\}$, we define the l_1 path norm of Θ to be

$$\|\Theta\|_{\mathcal{P}} = |\boldsymbol{\alpha}|^T \prod_{l=1}^{L} \left(I + \frac{1}{L}|\boldsymbol{U}_l||\boldsymbol{W}_l| \right) \mathbf{1}. \tag{21}$$

With the definitions above, we are ready to state the direct and inverse approximation theorems for the flow-induced function spaces [31].

Theorem 23 (Direct Approximation Theorem)**.** *Let $f \in \tilde{\mathcal{D}}_2$, $\delta \in (0,1)$. Then, there exists an absolute constant C, such that for any*

$$L \geq C \left(m^4 D^6 \|f\|_{\tilde{\mathcal{D}}_2}^5 (\|f\|_{\tilde{\mathcal{D}}_2} + D)^2 \right)^{\frac{3}{\delta}},$$

there is an L-layer residual network $f_L(\cdot;\Theta)$ that satisfies

$$\|f - f_L(\cdot;\Theta)\|^2 \leq \frac{\|f\|_{\tilde{\mathcal{D}}_2}^2}{L^{1-\delta}}, \tag{22}$$

and

$$\|\Theta\|_{\mathcal{P}} \leq 9\|f\|_{\tilde{\mathcal{D}}_1} \tag{23}$$

Theorem 24 (Inverse Approximation Theorem)**.** *Let f be a function defined on X. Assume that there is a sequence of residual networks $\{f_L(\cdot;\Theta_L)\}_{L=1}^{\infty}$ such that $\|f(\boldsymbol{x}) - f_L(\boldsymbol{x};\Theta_L)\| \to 0$ as $L \to \infty$. Assume further that the parameters in $\{f_L(\cdot;\Theta)\}_{L=1}^{\infty}$ are (entry-wise) bounded by c_0. Then, we have $f \in \mathcal{D}_\infty$, and*

$$\|f\|_{\mathcal{D}_\infty} \leq \frac{2e^{m(c_0^2+1)} D^2 c_0}{m}$$

Moreover, if there exists constant c_1 such that $\|f_L\|_{\mathcal{D}_1} \leq c_1$ holds for any $L > 0$, then we have

$$\|f\|_{\mathcal{D}_1} \leq c_1$$

The Rademacher complexity estimate is only established for a family of modified flow-induced function norms $\|\cdot\|_{\hat{\mathcal{D}}_p}$ (see the factor 2 in the definition below). It is not clear at this stage whether this is only a technical difficulty.

Let

$$\|f\|_{\hat{\mathcal{D}}_p} = \inf_{f=f_{\boldsymbol{\alpha},\{\rho_t\}}} |\boldsymbol{\alpha}|^T \hat{N}_p(1) + \|\hat{N}_p(1)\|_1 - D + \|\{\rho_t\}\|_{Lip}, \tag{24}$$

where $\hat{N}_p(t)$ is given by

$$\begin{aligned}\hat{N}_p(0) &= 2\mathbf{1},\\ \dot{\hat{N}}_p(t) &= 2\left(\mathbb{E}_{\rho_t}(|\boldsymbol{U}||\boldsymbol{W}|)^p\right)^{1/p}\hat{N}_p(t).\end{aligned}$$

Denote by $\hat{\mathcal{D}}_p$ the space of functions with finite $\hat{\mathcal{D}}_p$ norm. Then, we have

Theorem 25 ([31]). *Let $\hat{\mathcal{D}}_p^Q = \{f \in \hat{\mathcal{D}}_p : \|f\|_{\hat{\mathcal{D}}_p} \le Q\}$, then we have*

$$\mathrm{Rad}_n(\hat{\mathcal{D}}_2^Q) \le 18Q\sqrt{\frac{2\log(2d)}{n}}. \tag{25}$$

Next we turn to the generalization error estimates for the regularized estimator. At the moment, for the same reason as above, such estimates have only been proved when the empirical risk is regularized by a weighted path norm

$$\|\Theta\|_{\mathcal{WP}} = |\boldsymbol{\alpha}|^T \prod_{l=1}^{L}\left(I + \frac{2}{L}|\boldsymbol{U}_l||\boldsymbol{W}_l|\right)\mathbf{1}, \tag{26}$$

which is the discrete version of (24). This norm assigns larger weights to paths that pass through more non-linearities. Now consider the residual network (12) and the regularized empirical risk:

$$\mathcal{J}(\Theta) := \hat{\mathcal{R}}(\Theta) + 3\lambda\|\Theta\|_{\mathcal{WP}}\sqrt{\frac{2\log(2d)}{n}}, \tag{27}$$

Theorem 26 ([30]). *Let $f^* : X \to [0,1]$. Fix any $\lambda \ge 4 + 2/(3\sqrt{2\log(2d)})$. Assume that $\hat{\Theta}$ is an optimal solution of the regularized model (27). Then for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the random training samples, the population risk satisfies*

$$\mathcal{R}(\hat{\Theta}) \le \frac{3\|f\|_{\mathcal{B}}^2}{Lm} + (4\|f\|_{\mathcal{B}} + 1)\frac{3(4+\lambda)\sqrt{2\log(2d)} + 2}{\sqrt{n}} + 4\sqrt{\frac{2\log(14/\delta)}{n}}. \tag{28}$$

3.4 Multi-layer networks: Tree-like function spaces

A neural network with L hidden layers is a function of the form

$$f(\boldsymbol{x}) = \sum_{i_L=1}^{m_L} a_{i_L}^L \sigma\left(\sum_{i_{L-1}=1}^{m_{L-1}} a_{i_L i_{L-1}}^{L-1}\sigma\left(\dots\sigma\left(\sum_{i_1=1}^{m_1} a_{i_2 i_1}^1\,\sigma\left(\sum_{i_0=1}^{d+1} a_{i_1 i_0}^0\, x_{i_0}\right)\right)\right)\right) \tag{29}$$

where $a^{\ell}_{i_{\ell+1}i_\ell}$ with $i_{\ell+1} \in [m_{\ell+1}]$ and $i_\ell \in [m_\ell]$ are the weights of the neural network. Here the bias terms in the intermediate layers are omitted without loss of generality. Analogous to the Barron norm, we introduce the path-norm proxy by

$$\sum_{i_L,\dots,i_0} \left| a^L_{i_L}\, a^{L-1}_{i_L i_{L-1}} \cdots a^1_{i_2 i_1}\, a^0_{i_1 i_0} \right|.$$

and the path-norm of the function as the infimum of the path-norm proxies over all weights inducing the function.

$$\|f\|_{\mathcal{W}^L} = \inf \left\{ \sum_{i_L,\dots,i_0} \left| a^L_{i_L}\, a^{L-1}_{i_L i_{L-1}} \cdots a^1_{i_2 i_1}\, a^0_{i_1 i_0} \right| \,\middle|\, f \text{ satisfies (29)} \right\} \tag{30}$$

The natural function spaces for the purposes of approximation theory are the *tree-like function spaces* $\mathcal{W}^L$ of depth $L \in \mathbb{N}$ introduced in [36], where the norm can be extended in a natural way. For trees with a single hidden layer, Barron space and tree-like space agree, and the properties of tree-like function spaces are reminiscent of Barron space. Instead of stating all the results as theorems, we will just list them below.

- Rademacher complexity/generalization gap: Let $\mathcal{F}_Q = \{f \in C^0, \|f\|_{\mathcal{W}^L} \leq Q\}$ be the closed ball of radius $Q > 0$ in the tree-like function space. Then $\mathrm{Rad}_S(\mathcal{F}_Q) \leq 2^{L+1} Q \sqrt{\frac{2\ln(2d+2)}{n}}$. If a stronger version of the path-norm is controlled, then the dependence on depth can be weakened to L^3 instead of 2^L [12]. But remember, here we are thinking of keep L fixed at some finite value and increase the widths of the layers.

- The closed unit ball of the tree-like space of depth $L \geq 1$ is compact (in particular closed) in $C^0(K)$ for every compact set $K \subseteq \mathbb{R}^d$ and in $L^2(P)$ for every compactly supported probability measure P.

- In particular, an inverse approximation theorem holds: If $\|f_m\|_{\mathcal{W}^L} \leq C$ and $f_m \to f$ in $L^2(P)$, then f is in the tree-like function space of the same depth.

- The direct approximation theorem holds, but not with Monte Carlo rate. For f^* in a tree-like function space and $m \in \mathbb{N}$, there exists a network f_m with layers of width $m_\ell = m^{L-\ell+1}$ such that

  $$\|f_m - f^*\|_{L^2(P)} \leq \frac{2^L \|f^*\|_{\mathcal{W}^L}}{\sqrt{m}}.$$

  Note that the network has $O(m^{2L-1})$ weights. Part of this stems from the recursive definition, and by rearranging the index set into a tree-like structure, the number of free parameters (but dimension-independent) can be dropped to $O(m^{L+1})$.

  It is unclear whether a depth-independent (or less depth-dependent) approximation rate can be expected. Unlike two-layer networks and residual neural networks, layers of a multi-layer network discretize into conditional expectations, whereas the other function classes are naturally expressed as expectations. A larger number of parameters for multi-layer networks is therefore natural.

- Tree-like function spaces form a scale in the sense that if f is in the tree-like function space of depth ℓ, then it is also in the tree-like function space of depth $L > \ell$ and $\|f\|_{\mathcal{W}^L} \leq 2\,\|f\|_{\mathcal{W}^\ell}$.
- If $f : \mathbb{R}^d \to \mathbb{R}^k$ and $g : \mathbb{R}^k \to \mathbb{R}$ are tree-like functions in $\mathcal{W}^L$ and $\mathcal{W}^\ell$ respectively, then their composition $g \circ f$ is in $\mathcal{W}^{L+\ell}$.

In analogy to two-layer neural networks, we can prove a priori error estimates for multi-layer networks. Let P be a probability measure on X and $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be a set of iid samples drawn from P. For finite neural networks with weights $(a^L, \ldots, a^0) \in \mathbb{R}^{m_L} \times \cdots \times \mathbb{R}^{m_1 \times d}$ we denote

$$\widehat{\mathcal{R}}_n(a^L, \ldots, a^0) = \hat{\mathcal{R}}_n(f_{a^L,\ldots,a^0})$$

$$f_{a^L,\ldots,a^0}(\boldsymbol{x}) = \sum_{i_L=1}^{m_L} a^L_{i_L} \sigma \left(\sum_{i_{L-1}=1}^{m_{L-1}} a^{L-1}_{i_L i_{L-1}} \sigma \left(\sum_{i_{L-2}} \ldots \sigma \left(\sum_{i_1=1}^{m_1} a^1_{i_2 i_1} \sigma \left(\sum_{i_0=1}^{d+1} a^0_{i_1 i_0}\, x_{i_0} \right) \right) \right) \right).$$

Consider the regularized loss function: regularized risk functional

$$L_n(a^0, \ldots, a^L) = \widehat{\mathcal{R}}_n(a^L, \ldots, a^0) + \frac{9\,L^2}{m} \left[\sum_{i_L=1}^{m_L} \cdots \sum_{i_0=1}^{d+1} \left| a^L_{i_L}\, a^{L-1}_{i_L i_{L-1}} \ldots a^0_{i_1 i_0} \right| \right]^2$$

Theorem 27 (Generalization error)**.** *Assume that the target function satisfies $f^* \in \mathcal{W}^L$. Let $\mathcal{H}_m$ be the class of neural networks with architecture like in the direct approximation theorem for tree-like function spaces, i.e. $m_\ell = m^{L-\ell+1}$ for $\ell \geq 1$. Let f_m be the function given by $\operatorname{argmin}_{(a^0,\ldots,a^L)\in\mathcal{H}_m} L_n$. Then f_m satisfies the risk bound*

$$\mathcal{R}(f_m) \leq \frac{18\,L^2\,\|f^*\|^2_{\mathcal{W}^L}}{m} + 2^{L+3/2}\|f^*\|_{\mathcal{W}^L}\sqrt{\frac{2\,\log(2d+2)}{n}} + \bar{c}\,\sqrt{\frac{2\,\log(2/\delta)}{n}}. \tag{31}$$

In particular, there exists a function f_m satisfying the risk estimate. Using a natural cut-off, the constant $\bar{c}$ can be replaced with $\|f^*\|_{L^\infty} \leq C\,\|f^*\|_{\mathcal{W}^L}$.

3.5 Indexed representation and multi-layer spaces

Neural networks used in applications are not tree-like. To capture the structure of the functions represented by practical multi-layer neural networks, [36, 71] introduced an indexed representation of neural network functions.

Definition 28. For $0 \leq i \leq L$, let $(\Omega_i, \mathcal{A}_i, \pi^i)$ be probability spaces where $\Omega_0 = \{0, \ldots, d\}$ and π^0 is the normalized counting measure. Consider measurable functions $a^L : \Omega_L \to \mathbb{R}$ and $a^i : \Omega_{i+1} \times \Omega_i \to \mathbb{R}$ for $0 \leq i \leq L-1$. The *arbitrarily wide neural network* modeled on the index spaces $\{\Omega_i\}$ with weight functions $\{a^i\}$ is

$$f_{a^L,\ldots,a^0}(\boldsymbol{x}) = \int_{\Omega_L} a^{(L)}_{\theta_L} \sigma \left(\int_{\Omega_{L-1}} \ldots \sigma \left(\int_{\Omega_1} a^1_{\theta_2,\theta_1} \sigma \left(\int_{\Omega_0} a^0_{\theta_1,\theta_0}\, x_{\theta_0} \pi^0(\mathrm{d}\theta_0) \right) \pi^1(\mathrm{d}\theta_1) \right) \ldots \pi^{(L-1)}(\mathrm{d}\theta_{L-1}) \right) \pi^L(\mathrm{d}\theta_L). \tag{32}$$

If the index spaces are finite, this coincides with finite neural networks and the integrals are replaced with finite sums. From this we see that the class of arbitrarily wide neural networks modeled on certain index spaces may not be a vector space. If all weight spaces $\{\Omega_i\}$ are sufficiently expressive (e.g. the unit interval with Lebesgue measure), then the set of multi-layer networks modeled on $\Omega_L, \dots, \Omega_0$ becomes a vector space. The key property is that $(0,1)$ can be decomposed into two sets of probability $1/2$, each of which is isomorphic to itself, so adding two neural networks of equal depth is possible.

The space of arbitrarily wide neural networks is a subspace of the tree-like function space of depth L that consists of functions f with a finite path-norm

$$\|f\|_{\Omega_L,\dots,\Omega_0;K} = \inf\left\{ \int_{\prod_{i=0}^L \Omega_i} \left|a^{(L)}_{\theta_L} \dots a^{(0)}_{\theta_1\theta_0}\right| \left(\pi^L \otimes \dots \otimes \pi^0\right)\left(\mathrm{d}\theta_L \otimes \dots \otimes \mathrm{d}\theta_0\right) \,\middle|\, f = f_{a^L,\dots,a^0} \text{ on } K \right\}.$$

Since the coefficients of non-consecutive layers do not share an index, this may be a proper subspace for networks with mutiple hidden layers. If $\Omega_i = (0,1)$, the space of arbitrarily wide networks with one hidden layer coincides with Barron space.

In a network with two hidden layers, the (vector-valued) output of the zeroth layer is fixed independently of the second layer. Thus the output of the first layer is a vector whose coordinates lie in the subset of Barron space which have L^1-densities with respect to the (fixed) distribution of zeroth-layer weights on $\mathbb{R}^{d+1}$. This subspace is a separable subset of (non-separable) Barron space (see [37] for some functional analytic considerations on Barron space). It is, however, still unclear whether the tree-like space and the space of arbitrarily wide neural networks on sufficiently expressive index spaces agree. The latter contains all Barron functions and their compositions, including products of Barron functions.

The space of measurable weight functions which render the path-norm finite is inconveniently large when considering training dynamics. To allow a natural gradient flow structure, we consider the subset of functions with L^2-weights. This is partially motivated by the observation that the L^2-norm of weights controls the path-norm.

Lemma 29. *If $f = f_{a^L,\dots,a^0}$, then*

$$\|f\|_{\Omega_L,\dots,\Omega_0;K} \leq \inf\left\{ \|a^L\|_{L^2(\pi^L)} \prod_{i=0}^{L-1} \|a^i\|_{L^2(\pi^{i+1}\otimes\pi^i)} \,\middle|\, a^i \text{ s.t. } f = f_{a^L,\dots,a^0} \text{ on } K \right\}$$

Proof. We sketch the proof for two hidden layers.

$$\begin{aligned}
\int_{\Omega_2\times\Omega_1\times\Omega_0} \left|a^2_{\theta_2}\, a^1_{\theta_2\theta_1}\, a^0_{\theta_1\theta_0}\right| \mathrm{d}\theta_2\, \mathrm{d}\theta_1\, \mathrm{d}\theta_0 &= \int_{\Omega_2\times\Omega_1\times\Omega_0} \left|a^2_{\theta_2} a^0_{\theta_1\theta_0}\right| \left|a^1_{\theta_2\theta_1}\right| \mathrm{d}\theta_2\, \mathrm{d}\theta_1\, \mathrm{d}\theta_0 \\
&\leq \left(\int_{\Omega_2\times\Omega_1\times\Omega_0} \left|a^2_{\theta_2} a^0_{\theta_1\theta_0}\right|^2 \mathrm{d}\theta_2\, \mathrm{d}\theta_1\, \mathrm{d}\theta_0 \right)^{\frac{1}{2}} \left(\int_{\Omega_2\times\Omega_1\times\Omega_0} \left|a^1_{\theta_2\theta_1}\right|^2 \mathrm{d}\theta_2\, \mathrm{d}\theta_1\, \mathrm{d}\theta_0 \right)^{\frac{1}{2}} \\
&= \left(\int_{\Omega_2} \left|a^2_{\theta_2}\right|^2 \mathrm{d}\theta_2 \right)^{\frac{1}{2}} \left(\int_{\Omega_2\times\Omega_1} \left|a^1_{\theta_2\theta_1}\right|^2 \mathrm{d}\theta_2 \mathrm{d}\theta_1 \right)^{\frac{1}{2}} \left(\int_{\Omega_1\times\Omega_0} \left|a^0_{\theta_1\theta_0}\right|^2 \mathrm{d}\theta_1 \mathrm{d}\theta_0 \right)^{\frac{1}{2}}.
\end{aligned}$$

□

Note that the proof is entirely specific to network-like architectures and does not generalize to tree-like structures. We define the measure of complexity of a function (which is not a norm) as

$$Q(f) = \inf \left\{ \|a^L\|_{L^2(\pi^L)} \prod_{i=0}^{L-1} \|a^i\|_{L^2(\pi^{i+1} \otimes \pi^i)} \,\middle|\, a^i \text{ s.t. } f = f_{a^L,\dots,a^0} \text{ on } K \right\}.$$

We can equip the class of neural networks modeled on index spaces $\{\Omega_i\}$ with a metric which respects the parameter structure.

Remark 30. The space of arbitrarily wide neural networks with L^2-weights can be metrized with the Hilbert-weight metric

$$d_{HW}(f,g) = \inf \left\{ \sum_{\ell=0}^{L} \|a^{\ell,f} - a^{\ell,g}\|_{L^2(\pi^\ell)} \,\middle|\, a^{L,f}, \dots, a^{0,g} \text{ s.t. } f = f_{a^{L,f},\dots,a^{0,f}},\ g = f_{a^{L,g},\dots,a^{0,g}} \text{ and } \right.$$
$$\left. \|a^{\ell,h}\| \equiv \left(\prod_{i=0}^{L} \|a^{i,h}\|_{L^2} \right)^{\frac{1}{L+1}} \leq 2\, Q(h)^{\frac{1}{L+1}} \text{ for } h \in \{f,g\} \right\}. \tag{33}$$

The normalization across layers is required to ensure that functions in which one layer can be chosen identical do not have zero distance by shifting all weight to the one layer.

We refer to these metric spaces (metric vector spaces if $\Omega_i = (0,1)$ for all $i \geq 1$) as *multi-layer spaces*. They are complete metric spaces.

3.6 Depth separation in multi-layer networks

We can ask how much larger L-layer space is compared to $(L-1)$-layer space. A satisfying answer to this question is still outstanding, but partial answers have been found, mostly concerning the differences between networks with one and two hidden layers.

Example 31. The structure theorem for Barron functions Theorem 10 shows that functions which are non-differentiable on a curved hypersurface are not Barron. In particular, this includes distance functions from hypersurfaces like

$$f(\boldsymbol{x}) = \operatorname{dist}(\boldsymbol{x}, S^{d-1}) = \left|1 - \|\boldsymbol{x}\|_{\ell^2}\right|.$$

It is obvious, however, that f is the composition of two Barron functions and therefore can be represented exactly by a neural network with two hidden layers. This criterion is easy to check in practice and therefore of greater potential impact than mere existence results. But it says nothing about approximation by finite neural networks.

Remark 32. Neural networks with two hidden layers are significantly more flexible than networks with one hidden layer. In fact, there exists an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ which

is analytic, strictly monotone increasing and satisfies $\lim_{z\to\pm\infty}\sigma(z)=\pm 1$, but also has the surprising property that the finitely parametrized family

$$\mathcal{H}=\left\{\sum_{i=1}^{3d} a_i\,\sigma\left(\sum_{j=1}^{3d} b_{ij}\,\sigma\left(\boldsymbol{w}_j^T\boldsymbol{x}\right)\right)\;\middle|\; a_i,b_{ij}\in\mathbb{R}\right\}$$

is dense in the space of continuous functions on any compact set $K\subset\mathbb{R}^d$ [66]. The proof is based on the Kolmogorov-Arnold representation theorem, and the function is constructed in such a way that the translations

$$\frac{\sigma(z-3m)+\lambda_m\,\sigma(z-(3m+1))+\mu_m\,\sigma(z-(3m+2))}{\delta_m}$$

for $m\in\mathbb{N}$ form a countable dense subset of $C^0[0,1]$ for suitable $\lambda_m,\mu_m,\delta_m\in\mathbb{R}$. In particular, the activation function is virtually impossible to use in practice. However, the result shows that any approximation-theoretic analysis must be specific to certain activation functions and that common regularity requirements are not sufficient to arrive at a unified theory.

Example 33. A separation result like this can also be obtained with standard activation functions. If f is a Barron function, then there exists an $f_m(\boldsymbol{x})=\sum_{i=1}^m a_i\,\sigma(\boldsymbol{w}_i^T x)$ such that

$$\|f-f_m\|_{L^2(P)}\leq\frac{C}{\sqrt{m}}.$$

In [38], the authors show that there exists a function f such that

$$\|f-f_m\|_{L^2(P)}\geq\bar{c}\,m^{-1/(d-1)}.$$

but $f=g\circ h$ where g,h are Barron functions (whose norm grows like a low degree polynomial in d). The argument is based on Parseval's identity and neighbourhood growth in high dimensions. Intuitively, the authors argue that $\|f_m-f\|_{L^2(\mathbb{R}^d)}=\|\hat{f}_m-\hat{f}\|_{L^2(\mathbb{R}^d)}$ and that the Fourier transform of $\sigma(\boldsymbol{w}^T\boldsymbol{x})$ is concentrated on the line generated by w. If $\hat{f}$ is a radial function, its Fourier transform is radial as well and $f(\boldsymbol{x})=g(|\boldsymbol{x}|)$ can be chosen such that g is a Barron function and the Fourier transform of f has significant L^2-mass in high frequencies. Since small neighborhoods $B_\varepsilon(\boldsymbol{w}_i)$ only have mass $\sim\varepsilon^d$, we see that $\hat{f}$ and $\hat{f}_m$ cannot be close unless m is very large in high dimension. To make this intuition precise, some technical arguments are required since $\boldsymbol{x}\mapsto\sigma(\boldsymbol{w}^T\boldsymbol{x})$ is not an L^2-function.

There are some results on functions which can be approximated better with significantly deeper networks than few hidden layers, but a systematic picture is still missing. To the best of our knowledge, there are no results for the separation between L and $L+1$ hidden layers.

3.7 Tradeoffs between learnability and approximation

Example 33 and Remark 32 can be used to establish more: If $\tilde{f}$ is a Barron function and $\|f-\tilde{f}\|_{L^2(P)}<\varepsilon$, then there exists a dimension-dependent constant $c_d>0$ such that $\|\tilde{f}\|_{\mathcal{B}}\geq$

$c_d \varepsilon^{-\frac{d-3}{2}}$, i.e. f cannot be approximated to high accuracy by functions of low Barron norm. To see this, choose m such that $\frac{\bar{c}}{4} m^{-1/(d-1)} \leq \varepsilon \leq \frac{\bar{c}}{2} m^{-1/(d-1)}$ and let f_m be a network with m neurons. Then

$$2\varepsilon \leq \bar{c}\, m^{-1/d} \leq \|f_m - f\|_{L^2} \leq \|f_m - \tilde{f}\|_{L^2} + \|\tilde{f} - f\|_{L^2} \leq \|f_m - \tilde{f}\|_{L^2} + \varepsilon,$$

so if f_m is a network which approximates the Barron function $\tilde{f}$, we see that

$$\frac{\bar{c}}{4} m^{-1/(d-1)} \leq \varepsilon \leq \|f_m - \tilde{f}\|_{L^2} \leq \frac{\|\tilde{f}\|_{\mathcal{B}}}{m^{1/2}}.$$

In particular, we conclude that

$$\|\tilde{f}\|_{\mathcal{B}} \geq \frac{\bar{c}}{4} m^{1/2-1/(d-1)} = \frac{\bar{c}}{4} m^{\frac{d-3}{2(d-1)}} = \left(\frac{4}{\bar{c}}\right)^{\frac{d-1}{2}} \left(\frac{\bar{c}}{4} m^{-1/(d-1)}\right)^{-\frac{d-3}{2}} \geq \left(\frac{4}{\bar{c}}\right)^{\frac{d-1}{2}} \varepsilon^{-\frac{d-3}{2}}.$$

This is a typical phenomenon shared by *all* machine learning models of low complexity. Let P_d be Lebesgue-measure on the d-dimensional unit cube (which we take as the archetype of a truly 'high-dimensional' data distribution).

Theorem 34. *[35] Let Z be a Banach space of functions such that the unit ball B^Z in Z satisfies*

$$\mathbb{E}_{S \sim P_d^n} \operatorname{Rad}_S(B^Z) \leq \frac{C_d}{\sqrt{n}},$$

i.e. the Rademacher complexity on a set of N sample decays at the optimal rate in the number of data points. Then

1. *The* Kolmogorov width *of Z in the space of Lipschitz functions with respect to the L^2-metric is low in the sense that*

$$\limsup_{t \to \infty} \left[t^{\frac{2}{d-2}} \sup_{f(0)=0,\, f \text{ is 1-Lipschitz}} \inf_{\|g\|_Z \leq t} \|f - g\|_{L^2(P_d)} \right] \geq \bar{c} > 0.$$

2. *There exists a function f with Lispchitz constant 1 such that $f(0) = 0$, but*

$$\limsup_{t \to \infty} \left[t^{\gamma} \inf_{\|g\|_Z \leq t} \|f - g\|_{L^2(P_d)} \right] = \infty \qquad \forall\ \gamma > \frac{2}{d-2}.$$

This resembles the result of [65] for approximation by ridge functions under a constraint on the number of parameters, whereas here a complexity bound is assumed instead and no specific form of the model is prescribed (and the result thus applies to multi-layer networks as well).

Thus function spaces of low complexity are 'poor approximators' for general classes like Lipschitz functions since we need functions of large Z-norm to approximate functions to a prescribed level of accuracy. This includes all function spaces discussed in this review, although some spaces are significantly larger than others (e.g. there is a large gap between reproducing kernel Hilbert spaces, Barron space, and tree-like three layer space).

3.8 A priori vs. a posteriori estimates

The error estimate given above should be compared with a more typical form of estimate in the machine learning literature:

$$\mathcal{R}(\hat{\theta}_n) - \hat{\mathcal{R}}_n(\hat{\theta}_n) \lesssim \frac{\|\hat{\theta}_n\|}{\sqrt{n}} \tag{34}$$

where $\|\hat{\theta}_n\|$ is some suitably defined norm. Aside from the fact that (16) gives a bound on the total generalization error and (34) gives a bound on the generalization gap, there is an additional important difference: The right hand side of (16) depends only on the target function f^*, not the output of the machine learning model. The right hand side of (34) depends only on the output of the machine learning model, not the target function. In accordance with the practice in finite element methods, we call (16) a priori estimates and (34) a posteriori estimates.

How good and how useful are these estimates? A priori estimates discussed here tell us in particular that there exist functions in the hypothesis space for which the generalization error does not suffer from the CoD if the target function lies in the appropriate function space. It is likely that these estimates are nearly optimal in the sense that they are comparable to Monte Carlo error rates (except for multi-layer neural networks, see below). It is possible to improve these estimates, for example using standard tricks for Monte Carlo sampling for the approximation error and local Rademacher complexity [13] for the estimation error . However, these would only improve the exponents in m and n by $O(1/d)$ which diminishes for large d.

Regarding the quantitative value of these a priori estimates, the situation is less satisfactory. The first issue is that the values of the norms are not known since the target function is not known. One can estimate these values using the output of the machine learning model, but this does not give us rigorous bounds. An added difficulty is that the norms are defined as an infimum over all possible representations, the output of the machine learning model only gives one representation. But even if we use the exact values of these norms, the bounds given above are still not tight. For one thing, the use of Monte Carlo sampling to control the approximation error does not give a tight bound. This by itself is an interesting issue.

The obvious advantage of the a posteriori bounds is that they can be readily evaluated and give us quantitative bounds for the size of the generalization gap. Unfortunately this has not been borned out in practice: The values of these norms are so enormous that these bounds are almost always vacuous [29].

In finite element methods, a posteriori estimates are used to help refining the mesh in adaptive methods. Ideally one would like to do the same for machine learning models. However, little has been done in this direction.

Since the a posteriori bounds only controls the generalization gap, not the full generalization error, it misses an important aspect of the whole picture, namely, the approximation error. In fact, by choosing a very strong norm, one can always obtain estimates of the type in (34). However, with such strongly constrained hypothesis space, the approximation error

might be huge. This is indeed the case for some of the norm-based a posteriori estimates in the literature. See [30] for examples.

3.9 What's not known?

Here is a list of problems that we feel are most pressing.

1. Sharper estimates. There are two obvious places where one should be able to improve the estimates.

- In the current analysis, the approximation error is estimated with the help of Monte Carlo sampling. This gives us the typical size of the error for randomly picked parameters. However, in machine learning, we are only interested in the smallest error. This is a clean mathematical problem that has not received attention.

- The use of Rademacher complexity to bound the generalization gap neglects the fact that the integrand in the definition of the population risk (as well as the empirical risk) should itself be small, since it is the point-wise error. This should be explored further. We refer to [13] for some results in this direction.

2. The rate for the approximation error for functions in multi-layer spaces is not the same as Monte Carlo. Can this be improved?

More generally, it is not clear whether the multi-layer spaces defined earlier are the right spaces for multi-layer neural networks.

3. We introduced two kinds of flow-induced spaces for residual neural networks. To control the Rademacher complexity, we had to introduce the weighted path norm. This is not necessary for the approximation theory. It is not clear whether similar Rademacher complexity estimates can be proved for the spaces $\mathcal{D}_2$ or $\tilde{\mathcal{D}}_2$.

4. Function space for convolutional neural networks that fully explores the benefit of symmetry.

5. Another interesting network structure is the DenseNet [46]. Naturally one is interested in the natural function space associated with DenseNets.

4 The loss function and the loss landscape

It is a surprise to many that simple gradient descent algorithms work quite well for optimizing the loss function in common machine learning models. To put things into perspective, no one would dream of using the same kind of algorithms for protein folding – the energy landscape for am typical protein is so complicated with lots of local minima that gradient descent algorithms will not go very far. The fact that they seem to work well for machine learning models strongly suggests that the landscapes for the loss functions are qualitatively different. An important question is to quantify exactly how the loss landscape looks like. Unfortunately, theoretical results on this important problem is still quite scattered and there is not yet a general picture that has emerged. But generally speaking, the current

understanding is that while it is possible to find arbitrarily bad examples for finite sized neural networks, their landscape simplifies as the size increases.

To begin with, the loss function is non-convex and it is easy to cook up models for which the loss landscape has bad local minima (see for example [82]). Moreover, it has been suggested that for small size two-layer ReLU networks with teacher networks as the target function, nearly all target networks lead to spurious local minima, and the probability of hitting such local minima is quite high [74]. It has also been suggested the over-parametrization helps to avoid these bad spurious local minima.

The loss landscape of large networks can be very complicated. [79] presented some amusing numerical results in which the authors demonstrated that one can find arbitrarily complex patterns near the global minima of the loss function. Some theoretical results along this direction were proved in [25]. Roughly speaking, it was shown that for any $\varepsilon > 0$, every low-dimensional pattern can be found in a loss surface of a sufficiently deep neural network, and within the pattern there exists a point whose loss is within ε of the global minimum [25].

On the positive side, a lot is known for linear and quadratic neural network models. For linear neural network models, it has been shown that [49]: (1) every local minimum is a global minimum, (2) every critical point that is not a global minimum is a saddle point, (3) for networks with more than three layers there exist "bad" saddle points where the Hessian has no negative eigenvalue and (4) there are no such bad saddle points for networks with three layers.

Similar results have been obtained for over-parametrized two-layer neural network models with quadratic activation [81, 27]. In this case it has been shown under various conditions that (1) all local minima are global and (2) all saddle points are strict, namely there are directions of strictly negative curvature. Another interesting work for the case of quadratic activation function is [68]. In the case when the target function is a "single neuron", [68] gives an asymptotically exact (as $d \to \infty$) characterization of the number of training data samples needed for the global minima of the empirical loss to give rise to a unique function, namely the target function.

For over-parametrized neural networks with smooth activation function, the structure of the global minima is characterized in the paper of Cooper [23]: Cooper proved that the locus of the global minima is generically (i.e. possibly after an arbitrarily small change to the data set) a smooth $m - n$ dimensional submanifold of $\mathbb{R}^m$ where m is the number of free parameters in the neural network model and n is the training data size.

The set of minimizers of a convex function is convex, so unless the manifold of minimizers is an affine subspace of $\mathbb{R}^m$, the loss function is non-convex (as can be seen by convergence to poor local minimizers from badly chosen initial conditions). If there are two sets of weights such that both achieve minimal loss, but not all weights on the line segment between them do, then somewhere on the connecting line there exists a local maximum of the loss function (restricted to the line). In particular, if the manifold of minimizers is curved at a point, then arbitrarily closedby there exists a point where the Hessian of the loss function has a negative eigenvalue. This lack of positive definiteness in training neural networks is observed

in numerical experiments as well. The curvature of the set of minimizers partially explains why the weights of neural networks converge to different global minimizers depending on the initial condition.

For networks where half of the weights in every layer can be set to zero if the remaining weights are rescaled appropriately, the set of global minimizers is connected [54].

We also mention the interesting empirical work reported in [57]. Among other things, it was demonstrated that adding skip connection has a drastic effect on smoothing the loss landscape.

4.1 What's not known?

We still lack a good mathematical tool to describe the landscape of the loss function for large neural networks. In particular, are there local minima and how large is the basin of attraction of these local minima if they do exist?

For fixed, finite dimensional gradient flows, knowledge about the landscape allows us to draw conclusions about the qualitative behavior of the gradient descent dynamics independent of the detailed dynamics. In machine learning, the dimensionality of the loss function is m, the number of free parameters, and we are interested in the limit as m goes to infinity. So it is tempting to ask about the landscape of the limiting (infinite dimensional) problem. It is not clear whether this can be formulated as a well-posed mathematical problem.

5 The training process: convergence and implicit regularization

The results of Section 3 tell us that good solutions do exist in the hypothesis space. The amazing thing is that simple minded gradient descent algorithms are able to find them, even though one might have to be pretty good at parameter tuning. In comparison, one would never dream of using gradient descent to perform protein folding, since the landscape of protein folding is so complicated with lots of bad local minima.

The basic questions about the training process are:

- Optimization: Does the training process converge to a good solution? How fast?
- Generalization: Does the solution selected by the training process generalize well? In particular, is there such thing as "implicit regularization"? What is the mechanism for such implicit regularization?

At the moment, we are still quite far from being able to answering these questions completely, but an intuitive picture has started to emerge.

We will mostly focus on the gradient descent (GD) training dynamics. But we will touch upon some important qualitative features of other training algorithms such as stochastic gradient descent (SGD) and Adam.

5.1 Two-layer neural networks with mean-field scaling

"Mean-field" is a notion in statistical physics that describes a particular form of interaction between particles. In the mean-field situation, particles interact with each other only through a mean-field which every particle contributes to more or less equally. The most elegant mean-field picture in machine learning is found in the case of two-layer neural networks: If one views the neurons as interacting particles, then these particles only interact with each other through the function represented by the neural network, the mean-field in this case. This observation was first made in [20, 70, 73, 78]. By taking the hydrodynamic limit for the gradient flow of finite neuron systems, these authors obtained a continuous integral differential equation that describes the evolution of the probability measure for the weights associated with the neurons.

Let

$$I(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m) = \hat{\mathcal{R}}_n(f_m), \quad \boldsymbol{u}_j = (a_j, \boldsymbol{w}_j), \quad f_m(\boldsymbol{x}) = \frac{1}{m}\sum_j a_j \sigma(\boldsymbol{w}_j^T \boldsymbol{x})$$

Define the GD dynamics by:

$$\frac{d\boldsymbol{u}_j}{dt} = -m\nabla_{\boldsymbol{u}_j} I(\boldsymbol{u}_1, \cdots, \boldsymbol{u}_m), \quad \boldsymbol{u}_j(0) = \boldsymbol{u}_j^0, \quad j \in [m] \tag{35}$$

Lemma 35. *Let*

$$\rho(d\boldsymbol{u}, t) = \frac{1}{m}\sum_j \delta_{\boldsymbol{u}_j(t)}$$

then the GD dynamics (35) can be expressed equivalently as:

$$\partial_t \rho = \nabla(\rho \nabla V), \quad V = \frac{\delta \hat{\mathcal{R}}_n}{\delta \rho} \tag{36}$$

(36) is the mean-field equation that describes the evolution of the probability distribution for the weights associated with each neuron. The lemma above simply states that (36) is satisfied for finite neuron systems.

It is well-known that (36) is the gradient flow of $\hat{\mathcal{R}}_n$ under the Wasserstein metric. This brings the hope that the mathematical tools developed in the theory of optimal transport can be brought to bear for the analysis of (36) [86]. In particular, we would like to use these tools to study the qualitative behavior of the solutions of (36) as $t \to \infty$. Unfortunately the most straightforward application of the results from optimal transport theory requires that the risk functional be displacement convex [69], a property that rarely holds in machine learning (see however the example in [48]). As a result, less than expected has been obtained using optimal transport theory.

The one important result, due originally to Chizat and Bach [20], is the following. We will state the result for the population risk.

Theorem 36. *[20, 21, 87] Let $\{\rho_t\}$ be a solution of the Wasserstein gradient flow such that*

- *ρ_0 is a probability distribution on the cone $\Theta := \{|a|^2 \leq |w|^2\}$.*
- *Every open cone in Θ has positive measure with respect to ρ_0.*

Then the following are equivalent.

1. *The velocity potentials $\frac{\delta \mathcal{R}}{\delta \rho}(\rho_t, \cdot)$ converge to a unique limit as $t \to \infty$.*
2. *$\mathcal{R}(\rho_t)$ decays to the global infimum value as $t \to \infty$.*

If either condition is met, the unique limit of $\mathcal{R}(\rho_t)$ is zero. If ρ_t also converges in the Wasserstein metric, then the limit ρ_∞ is a minimizer.

Intuitively, the theorem is slightly stronger than the statement that $\mathcal{R}(\rho_t)$ converges to its infimum value if and only if its derivative converges to zero in a suitable sense (if we approach from a flow of omni-directional distributions). The theorem is more a statement about a training algorithm than the energy landscape, and specific PDE arguments are used in its proof. A few remarks are in order:

1. There are further technical conditions for the theorem to hold.
2. Convergence of subsequences of $\frac{\delta \mathcal{R}}{\delta \rho}(\rho_t, \cdot)$ is guaranteed by compactness. The question of whether they converge to a unique limit can be asked independently of the initial distribution and therefore may be more approachable by standard means.
3. The first assumption on ρ_0 is a smoothness assumption needed for the existence of the gradient flow.
4. The second assumption on ρ_0 is called *omni-directionality*. It ensures that ρ can shift mass in any direction which reduces risk. The cone formulation is useful due to the homogeneity of ReLU.

This is almost the only non-trivial rigorous result known on the global convergence of gradient flows in the nonlinear regime. In addition, it reveals the fact that having full support for the probability distribution (or something to that effect) is an important property that helps for the global convergence.

The result is insensitive to whether a minimizer of the risk functional exists (i.e. whether the target function is in Barron space). If the target function is not in Barron space, convergence may be very slow since the Barron norm increases sub-linearly during gradient descent training.

Lemma 37. *[87, Lemma 3.3] If $\{\rho_t\}$ evolves by the Wasserstein-gradient flow of $\mathcal{R}$, then*

$$\lim_{t \to \infty} \frac{\|f_{\rho_t}\|_{\mathcal{B}}}{t} = 0.$$

In high dimensions, even reasonably smooth functions are difficult to approximate with functions of low Barron norm in the sense of Theorem 34. Thus a "dynamic curse of dimensionality" may affect gradient descent training if the target function is not in Barron space.

Theorem 38. *Consider population and empirical risk expressed by the functionals*

$$\mathcal{R}(\rho) = \frac{1}{2}\int_X (f_\rho - f^*)^2(\boldsymbol{x})d\boldsymbol{x}, \qquad \hat{\mathcal{R}}_n(\rho) = \frac{1}{2n}\sum_{i=1}^{n}(f_\rho - f^*)^2(\boldsymbol{x}_i)$$

where the points $\{\boldsymbol{x}_i\}$ *are i.i.d samples from the uniform distribution on* X*. There exists* f^* *with Lipschitz constant and* L^∞*-norm bounded by* 1 *such that the parameter measures* $\{\rho_t\}$ *defined by the* 2*-Wasserstein gradient flow of either* $\hat{\mathcal{R}}_n$ *or* $\mathcal{R}$ *satisfy*

$$\limsup_{t\to\infty}\left[t^\gamma\,\mathcal{R}(\rho_t)\right] = \infty$$

for all $\gamma > \frac{4}{d-2}$.

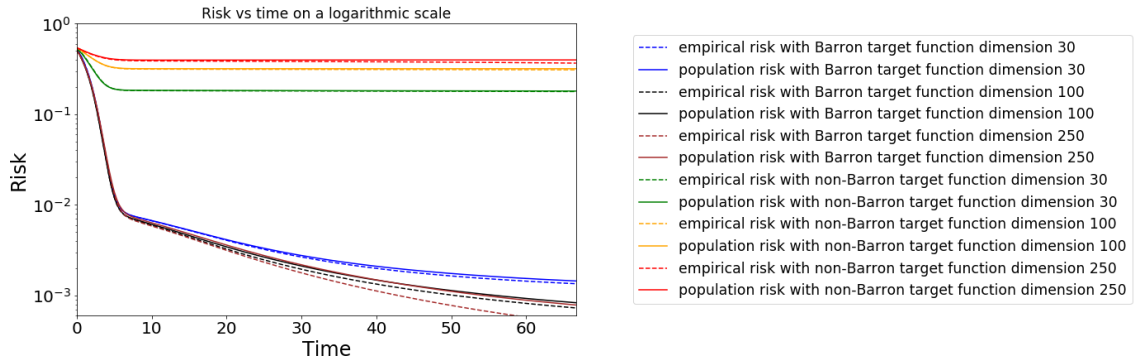


Figure 2: The rate of convergence of the gradient descent dynamics for Barron and non-Barron functions on a logarithmic scale. Convergence rates for Barron functions seem to be dimension-independent. This is not the case for non-Barron functions.

5.2 Two-layer neural networks with conventional scaling

In practice, people often use the conventional scaling (instead of the mean-field scaling) which takes the form:

$$f_m(\boldsymbol{x};\boldsymbol{a},\boldsymbol{B}) = \sum_{j=1}^{m} a_j\sigma(\boldsymbol{b}_j^T\boldsymbol{x}) = \boldsymbol{a}^T\sigma(\boldsymbol{B}\boldsymbol{x}),$$

A popular initialization [55, 43] is as follows

$$a_j(0) \sim \mathcal{N}(0,\beta^2), \qquad \boldsymbol{b}_j(0) \sim \mathcal{N}(0, I/d)$$

where $\beta = 0$ or $1/\sqrt{m}$. For later use, we define the Gram matrix $K = (K_{ij}) \in \mathbb{R}^{n\times n}$:

$$K_{i,j} = \frac{1}{n}\mathbb{E}_{\boldsymbol{b}\sim\pi_0}[\sigma(\boldsymbol{b}^T\boldsymbol{x}_i)\sigma(\boldsymbol{b}^T\boldsymbol{x}_j)].$$

With this "scaling", the one case where a lot is known is the so-called highly over-parametrized regime. There is both good and bad news in this regime. The good news is that one can prove exponential convergence to global minima of the empirical risk.

Theorem 39 ([28]). *Let $\lambda_n = \lambda_{\min}(K)$ and assume $\beta = 0$. For any $\delta \in (0,1)$, assume that $m \gtrsim n^2\lambda_n^{-4}\delta^{-1}\ln(n^2\delta^{-1})$. Then with probability at least $1-6\delta$ we have*

$$\hat{\mathcal{R}}_n(\boldsymbol{a}(t),\boldsymbol{B}(t)) \le e^{-m\lambda_n t}\hat{\mathcal{R}}_n(\boldsymbol{a}(0),\boldsymbol{B}(0)) \tag{37}$$

Now the bad news: the generalization property of the converged solution is no better than that of the associated random feature model, defined by freezing $\{\boldsymbol{b}_j\} = \{\boldsymbol{b}_j(0)\}$, and only training $\{a_i\}$.

The first piece of insight that the underlying dynamics in this regime is effectively linear is given in [26]. [47] termed the effective kernel the "neural tangent kernel". Later it was proved rigorously that in this regime, the entire GD path for the two-layer neural network model is uniformly close to that of the associated random feature model [32, 5].

Theorem 40 ([32]). *Let $\boldsymbol{B}_0 = \boldsymbol{B}(0)$. Denote by $f_m(\boldsymbol{x};\tilde{\boldsymbol{a}}(\cdot),\boldsymbol{B}_0))$ the solutions of GD dynamics for the random feature model. Under the same setting as Theorem 39, we have*

$$\sup_{\boldsymbol{x}\in S^{d-1}} |f_m(\boldsymbol{x};\boldsymbol{a}(t),\boldsymbol{B}(t)) - f_m(\boldsymbol{x};\tilde{\boldsymbol{a}}(t),\boldsymbol{B}_0)| \lesssim \frac{(1+\sqrt{\ln(1/\delta)})^2\lambda_n^{-1}}{\sqrt{m}}. \tag{38}$$

In particular, there is no "implicit regularization" in this regime.

Even though convergence of the training process can be proved, overall this is a disappointing result. At the theoretical level, it does not shed any light on possible implicit regularization. At the practical level, it tells us that high over-parametrization is indeed a bad thing for these models.

What happens in practice? Do less over-parametrized regimes exist for which implicit regularization does actually happen? Some insight has been gained from the numerical study in [63].

Let us look at a simple example: the single neuron target function: $f_1^*(\boldsymbol{x}) = \sigma(\boldsymbol{b}^*\cdot\boldsymbol{x})$, $\boldsymbol{b}^* = \boldsymbol{e}_1$. This is admittedly a very simple target function. Nevertheless, the training dynamics for this function is quite representative of target functions which can be accurately approximated by a small number of neurons (i.e. effectively "over-parametrized").

Figure 3 shows some results from [63]. First note that the bottom two figures represent exactly the kind of results stated in Theorem 40. The top two figures suggest that the training dynamics displays two phases. In the first phase, the training dynamics follows closely that of the associated random feature model. This phase quickly saturates. The training dynamics for the neural network model is able to reduce the training (and test)

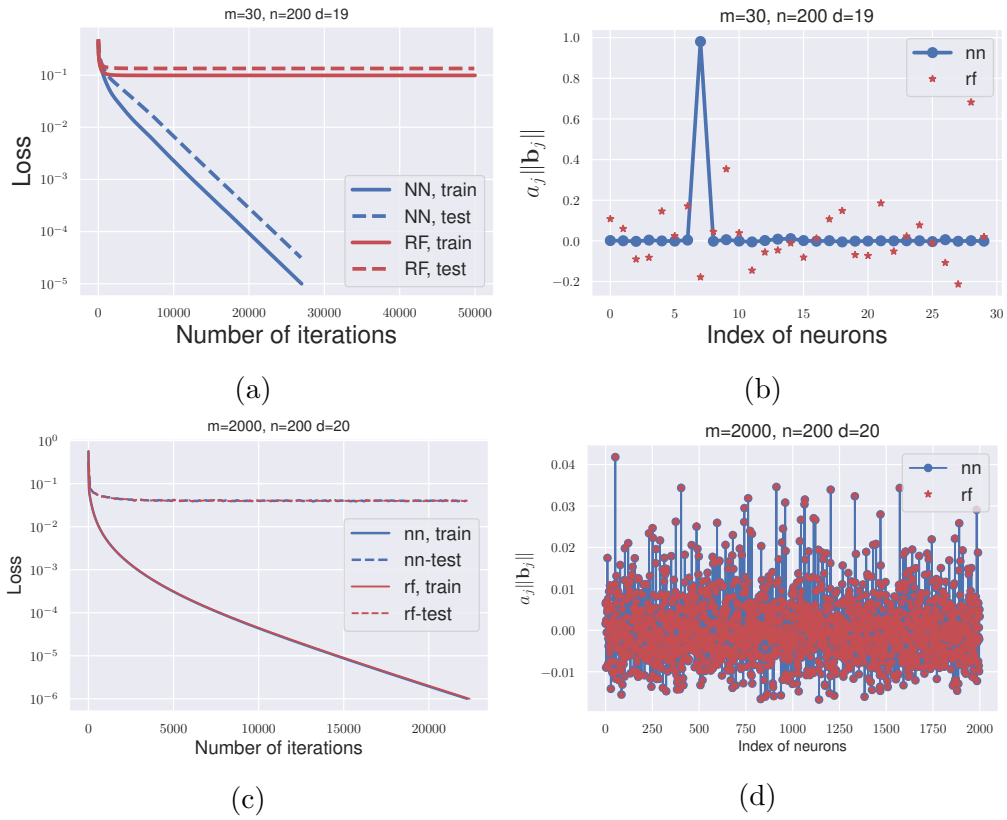


Figure 3: The dynamic behavior of learning single-neuron target function using GD with conventional scaling. We also show the results of the random feature model as a comparison. **Top:** the case when $m = 30, n = 200, d = 19$ in the mildly over-parametrized regime. **Bottom:** the case when $m = 2000, n = 200, d = 19$ in the highly over-parametrized regime. The learning rate $\eta = 0.001$. (a,c) The dynamic behavior of the training and test error. (b,d) The outer layer coefficient of each neuron for the converged solution.

error further in the second phase through a quenching-activation process: Most neurons are quenched in the sense that their outer layer coefficients a_j become very small, with the exception of only a few activated ones.

This can also be seen from the $m - n$ hyper-parameter space. Shown in Figure 4 are the heat maps of the test errors under the conventional and mean-field scaling, respectively. We see that the test error does not change much as m changes for the mean-field scaling. In contrast, there is a clear "phase transition" in the heat map for the conventional scaling when the training dynamics undergoes a change from the neural network-like to a random feature-like behavior. This is further demonstrated in Figure 5: One can see that the performance of the neural network models indeed becomes very close to that of the random feature model as the network width m increases. At the same time, the path norm also undergoes a sudden increase from the mildly over-parameterized/under-parameterized regime ($m \approx n/(d+1)$) to the highly over-parameterized regime ($m \approx n$). This may provide an explanation for the increase of the test error.

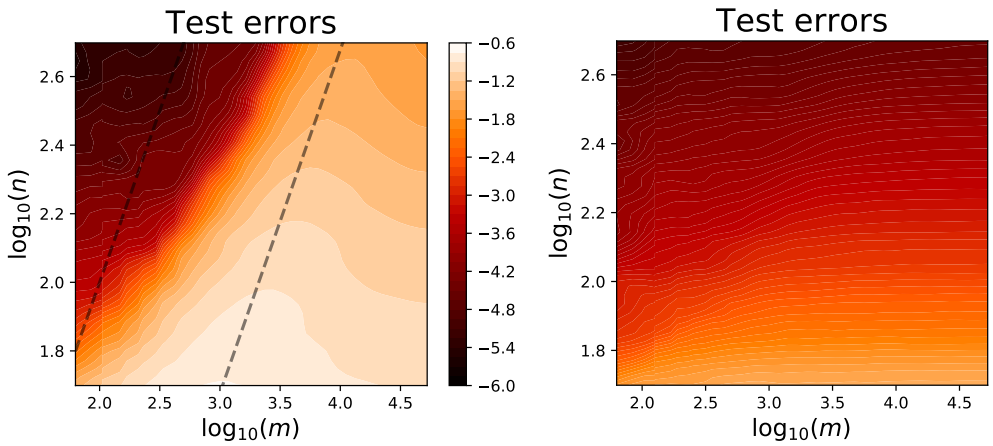


Figure 4: How the network width affects the test error of GD solutions. The test errors are given in logarithmic scale. These experiments are conducted on the single-neuron target function with $d = 20$ and learning rate $\eta = 0.0005$. The GD is stopped when the training loss is smaller than 10^{-7}. The two dashed lines correspond to $m = n/(d+1)$ (left) and $m = n$ (right), respectively. **Left:** Conventional scaling; **Right:** Mean-field scaling.

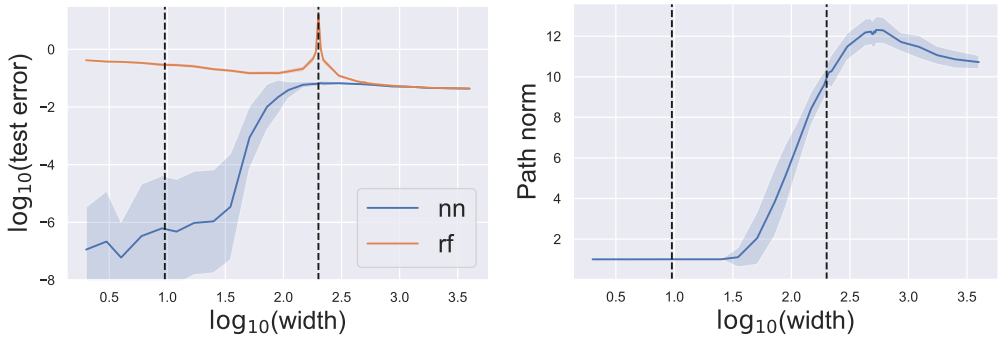


Figure 5: Here $n = 200, d = 20$, learning rate $= 0.001$. GD is stopped when the training error is smaller than 10^{-8}. The two dashed lines correspond to $m = n/(d+1)$ (left) and $m = n$ (right), respectively. Several independent experiments were performed. The mean (shown as the line) and variance (shown in the shaded region) are shown here. **Left:** test error; **Right**: path norm.

The existence of different kinds of behavior with drastically different generalization properties is one of the reasons behind the fragility of deep learning: If the network architecture happens to fall into the random feature-like regime, the performance will deteriorate.

5.3 Other convergence results for the training of neural network models

Convergence of GD for linear neural networks was proved in [14, 4, 89]. [59] analyzes the training of neural networks with quadratic activation function. It is proved that a modified GD with small initialization converges to the ground truth in polynomial time if the sample

size $n \geq O(dr^5)$ where d, r are the input dimension and the rank of the target weight matrix, respectively. [80] considers the learning of single neuron with SGD. It was proved that as long as the sample size is large enough, SGD converges to the ground truth exponentially fast if the model also consists of a single neuron. Similar analysis for the population risk was presented in [84].

5.4 Double descent and slow deterioration for the random feature model

At the continuous (in time) level, the training dynamics for the random feature model is a linear system of ODEs defined by the Gram matrix. It turns out that the generalization properties for this linear model is surprisingly complex, as we now discuss.

Consider a random feature model with features $\{\phi(\cdot; \boldsymbol{w})\}$ and probability distribution π over the feature vectors $\boldsymbol{w}$. Let $\{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_m\}$ be the random feature vectors sampled from π. Let Φ be an $n \times m$ matrix with $\Phi_{ij} = \phi(\boldsymbol{x}_i; \boldsymbol{w}_j)$ where $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\}$ is the training data, and let

$$\hat{f}(\boldsymbol{x}; \boldsymbol{a}) = \sum_{k=1}^{m} a_k \phi(\boldsymbol{x}; \boldsymbol{b}_k), \tag{39}$$

where $\boldsymbol{a} = (a_1, b_2, ..., a_m)^T$ are the parameters. To find $\boldsymbol{a}$, GD is used to optimize the following least squares objective function,

$$\min_{\boldsymbol{a} \in \mathbb{R}^m} \frac{1}{2n} \|\Phi \boldsymbol{a} - \boldsymbol{y}\|^2, \tag{40}$$

starting from the origin, where $\boldsymbol{y}$ is a vector containing the values of the target function at $\{\boldsymbol{x}_i,\ i = 1, 2, ..., n\}$. The dynamics of $\boldsymbol{a}$ is then given by

$$\frac{d}{dt}\boldsymbol{a}(t) = -\frac{1}{m}\frac{\partial}{\partial \boldsymbol{a}}\frac{1}{2n}\|\Phi \boldsymbol{a} - \boldsymbol{y}\|^2 = -\frac{1}{mn}\Phi^T(\Phi \boldsymbol{a} - \boldsymbol{y}). \tag{41}$$

Let $\Phi = U\Sigma V^T$ be the singular value decomposition of Φ, where

$$\Sigma = \text{diag}\{\lambda_1, \lambda_2, ..., \lambda_{\min\{n,m\}}\}$$

with $\{\lambda_i\}$ being the singular values of Φ, in descending order. Then the GD solution of (40) at time $t \geq 0$ is given by

$$\boldsymbol{a}(t) = \sum_{i:\lambda_i > 0} \frac{1 - e^{-\lambda_i^2 t/(mn)}}{\lambda_i} (\boldsymbol{u}_i^T \boldsymbol{y}) \boldsymbol{v}_i. \tag{42}$$

With this solution, we can conveniently compute the training and test error at any time t. Specifically, let $B = [\boldsymbol{w}_1, ..., \boldsymbol{w}_m]$, and with an abuse of notation let $\phi(\boldsymbol{x}; B) = (\phi(\boldsymbol{x}; \boldsymbol{w}_1), ..., \phi(\boldsymbol{x}; \boldsymbol{w}_m))^T$, the prediction function at time t is given by

$$\hat{f}_t(\boldsymbol{x}) = \phi(\boldsymbol{x}; B)^T \boldsymbol{a}(t) = \sum_{i:\lambda_i > 0} \frac{1 - e^{-\lambda_i^2 t/mn}}{\lambda_i} (\boldsymbol{u}_i^T \boldsymbol{y})(\boldsymbol{v}_i^T \phi(\boldsymbol{x}; B)). \tag{43}$$

Shown in the left figure of Figure 6 is the test error of the minimum norm solution (which is the limit of the GD path when initialized at 0) of the random feature model as a function of the size of the model m for $n = 500$ for the MNIST dataset [64]. Also shown is the smallest eigenvalue of the Gram matrix. One can see that the test error peaks at $m = n$ where the smallest eigenvalue of the Gram matrix becomes exceedingly small. This is the same as the "double descent" phenomenon reported in [16, 2]. But as can be seen from the figure (and discussed in more detail below), it is really a resonance kind of phenomenon caused by the appearance of very small eigenvalues of the Gram matrix when $m = n$.

Shown in right is the test error of the solution when the gradient descent training is stopped after different number of steps. One curious thing is that when training is stopped at moderately large values of training steps, the resonance behavior does not show up much. Only when training is continued to very large number of steps does resonance show up.

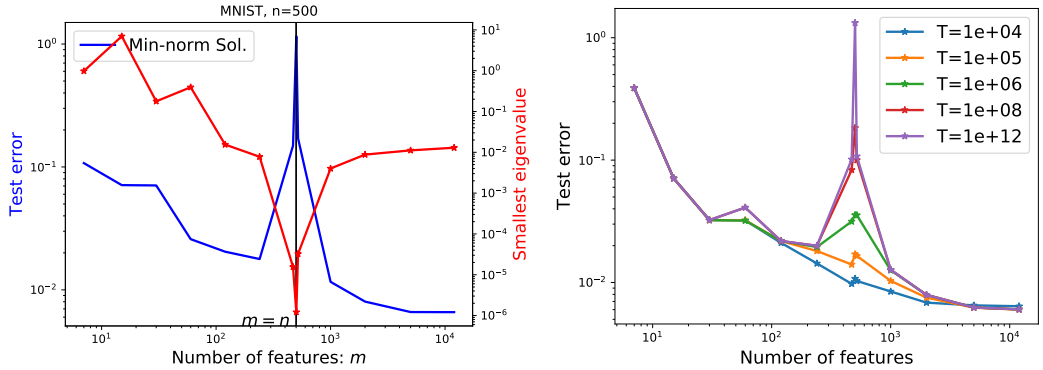
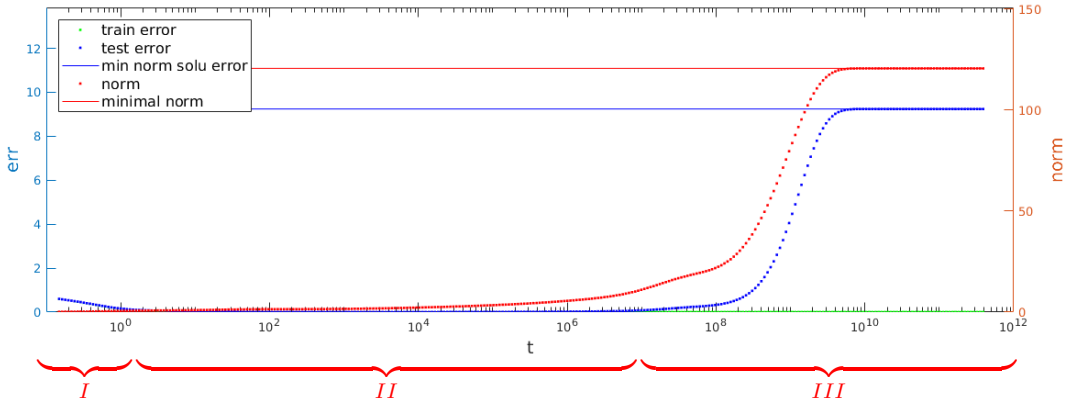


Figure 6: **Left:** The test error and minimal eigenvalue of Gram matrix as a function of m for $n = 500$. **Right:** The test error of GD solutions obtained by running different number of iterations. MNIST dataset is used.

Intuitively this is easy to understand. The large test error is caused by the small eigenvalues, because each term in (43) convergences to $1/\lambda_i$, and this limit is large when the corresponding eigenvalue is small. However, still by (43), the contribution of the eigenvalues enter through terms like $e^{-\lambda^2 t}$, and therefore shows up very slowly in time.

Some rigorous analysis of this can be found in [64] and will not be repeated here. Instead we show in Figure 7 an example of the detailed dynamics of the training process. One can see that the training dynamics can be divided into three regimes. In the first regime, the test error decreases from an $O(1)$ initial value to something small. In the second regime, which spans several decades, the test error remains small. It eventually becomes big in the third regime when the small eigenvalues start to contribute significantly. This slow deterioration phenomenon may also happen for more complicated models, like deep neural networks, considering that linear approximation of the loss function is effective near the global minimum, and hence in this regime the GD dynamics is similar to that of a linear model (e.g. random feature). Though, the conjecture needs to be confirmed by further theoretical and numerical study.

Figure 7



One important consequence of this resonance phenomenon is that it affects the training of two-layer neural networks under the conventional scaling. For example in Figure 4, the test error of neural networks under the conventional scaling peaks around $m = n$, with m being the width of the network. Though at $m = n$ the neural network has more parameters than n, it still suffers from this resonance phenomenon since as we saw before, in the early phase of the training, the GD path for the neural network model follows closely that of the corresponding random feature model.

5.5 Global minima selection

In the over-parametrized regime, there usually exist many global minima. Different optimization algorithms may select different global minima. For example, it has been widely observed that SGD tends to pick "flatter solutions" than GD [44, 51]. A natural question is which global minima are selected by a particular optimization algorithm.

An interesting illustration of this is shown in Figure 8. Here GD was used to train the FashionMNIST dataset to near completion, and was suddenly replaced by SGD. Instead of finishing the last step in the previous training process, the subsequent SGD path escapes from the vicinity of the minimum that GD was converging to, and eventually converges to a different global minimum, with a slightly smaller test error [51].

This phenomenon can be well-explained by considering the dynamic stability of the optimizers, as was done in [88]. It was found that the set of dynamically stable global minima is different for different training algorithm. In the example above, the global minimum that GD was converging to was unstable for SGD.

The gist of this phenomenon can be understood from the following simple one-dimensional optimization problem,

$$f(x) = \frac{1}{2n} \sum_{i=1}^{n} a_i x^2$$

with $a_i \geq 0 \; \forall i \in [n]$. The minimum is at $x = 0$. For GD with learning rate η to be stable,

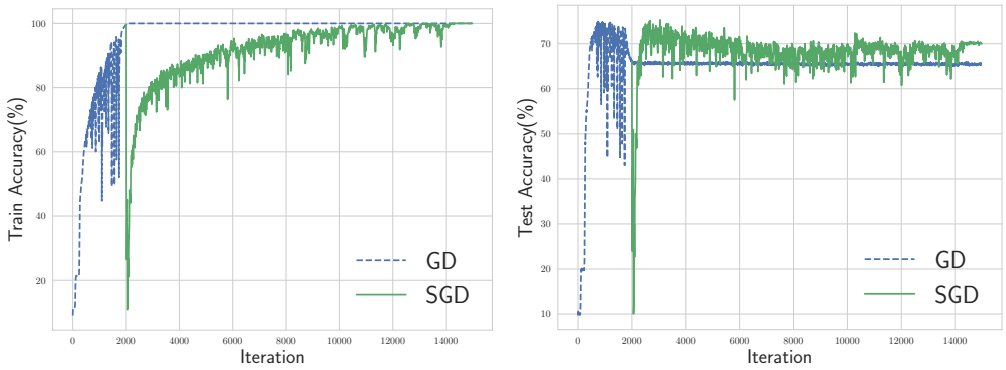


Figure 8: Escape phenomenon in fitting corrupted FashionMNIST. The GD solution is unstable for the SGD dynamics. Hence the path escapes after GD is suddenly replaced by SGD. **Left:** Training accuracy, **Right:** Test accuracy.

the following has to hold:

$$|1 - \eta a| \leq 1$$

SGD is given by:

$$x_{t+1} = x_t - \eta a_\xi x_t = (1 - \eta a_\xi) x_t, \tag{44}$$

where ξ is a random variable that satisfies $\mathbb{P}(\xi = i) = 1/n$. Hence, we have

$$\mathbb{E}x_{t+1} = (1 - \eta a)^t x_0, \tag{45}$$

$$\mathbb{E}x_{t+1}^2 = \left[(1 - \eta a)^2 + \eta^2 s^2\right]^t x_0^2, \tag{46}$$

where $a = \sum_{i=1}^n a_i/n$, $s = \sqrt{\sum_{i=1}^n a_i^2/n - a^2}$. Therefore, for SGD to be stable at $x = 0$, we not only need $|1 - \eta a| \leq 1$, but also $(1 - \eta a)^2 + \eta^2 s^2 \leq 1$. In particular, SGD can only converge with the additional requirement that $s \leq 2/\eta$.

The quantity a is called sharpness, and s is called non-uniformity in [88]. The above simple argument suggests that the global minima selected by SGD tend to be more uniform than the ones selected by GD.

This sharpness-non-uniformity criterion can be extended to multi-dimension. It turns out that this theoretical prediction is confirmed quite well by practical results. Figure 9 shows the sharpness and non-uniformity results of SGD solutions for a VGG-type network for the FashionMNIST dataset. We see that the predicted upper bound for the non-uniformity is both valid and quite sharp.

5.6 Qualitative properties of adaptive gradient algorithms

Adaptive gradient algorithms are a family of optimization algorithms widely used for training neural network models. These algorithms use a coordinate-wise scaling of the update direction (gradient or gradient with momentum) according to the history of the gradients. Two most popular adaptive gradient algorithms are RMSprop and Adam, whose update rules are

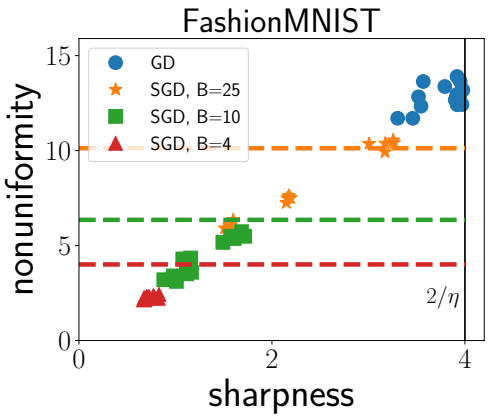


Figure 9: The sharpness-non-uniformity diagram for the minima selected by SGD applied to a VGG-type network for the FashionMNIST dataset. Different colors correspond to different set of hyper-parameters. B and μ in the figure stand for the batch size and the learning rate, respectively. The dashed line shows the predicted bound for the non-uniformity. One can see that (1) the data with different colors roughly lies below the corresponding dashed line and (2) the prediction given by the dashed line is quite sharp.

- **RMSprop:**

$$\begin{aligned} \boldsymbol{v}_{t+1} &= \alpha \boldsymbol{v}_t + (1-\alpha)(\nabla f(\boldsymbol{x}_t))^2 \\ \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \eta \frac{\nabla f(\boldsymbol{x}_t)}{\sqrt{\boldsymbol{v}_{t+1}} + \epsilon} \end{aligned} \tag{47}$$

- **Adam:**

$$\begin{aligned} \boldsymbol{v}_{t+1} &= \alpha \boldsymbol{v}_t + (1-\alpha)(\nabla f(\boldsymbol{x}_t))^2 \\ \boldsymbol{m}_{t+1} &= \beta \boldsymbol{m}_t + (1-\beta)\nabla f(\boldsymbol{x}_t) \\ \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t - \eta \frac{\boldsymbol{m}_{t+1}/(1-\beta^{t+1})}{\sqrt{\boldsymbol{v}_{t+1}/(1-\alpha^{t+1})} + \epsilon} \end{aligned} \tag{48}$$

In (47) and (48), ϵ is a small constant used to avoid division by 0, usually taken to be 10^{-8}. It is added to each component of the vector in the denominators of these equations. The division should also be understood as being component-wise. Here we will focus on Adam. For further discussion, we refer to [62].

One important tool for understanding these adaptive gradient algorithms is their continuous limit. Different continuous limits can be obtained from different ways of taking limits. If we let η tends to 0 while keeping α and β fixed, the limiting dynamics will be

$$\dot{\boldsymbol{x}} = -\frac{\nabla f(\boldsymbol{x})}{|\nabla f(\boldsymbol{x})| + \epsilon}. \tag{49}$$

This becomes signGD when $\epsilon = 0$. On the other hand, if we let $\alpha = 1 - a\eta$ and $\beta = 1 - b\eta$ and take $\eta \to 0$, then we obtain the limiting dynamics

$$\begin{aligned} \dot{\boldsymbol{v}} &= a(\nabla f(\boldsymbol{x})^2 - \boldsymbol{v}) \\ \dot{\boldsymbol{m}} &= b(\nabla f(\boldsymbol{x}) - \boldsymbol{m}) \\ \dot{\boldsymbol{x}} &= -\frac{(1-e^{-bt})^{-1}\boldsymbol{m}}{\sqrt{(1-e^{-at})^{-1}\boldsymbol{v}} + \epsilon} \end{aligned} \tag{50}$$

In practice the loss curves of Adam can be very complicated. The left panel of Figure 11 shows an example. Three obvious features can be observed from these curves:

1. **Fast initial convergence:** the loss curve decreases very fast, sometimes even super-linearly, at the early stage of the training.

2. **Small oscillations:** The fast initial convergence is followed by oscillations around the minimum.

3. **Large spikes:** spikes are sudden increase of the value of the loss. They are followed by an oscillating recovery. Different from small oscillations, spikes make the loss much larger and the interval between two spikes is longer.

The fast initial convergence can be partly explained by the convergence property of signGD, which attains global minimum in finite time for strongly convex objective functions. Specifically, we have the following proposition [62].

Proposition 41. *Assume that the objective function satisfies the Polyak-Lojasiewicz (PL) condition:* $\|\nabla f(\boldsymbol{x})\|_2^2 \geq \mu f(\boldsymbol{x})$*, for some positive constant* μ*. Define continuous signGD dynamics,*

$$\dot{\boldsymbol{x}}_t = -sign(\nabla f(\boldsymbol{x}_t)),$$

then we have

$$f(\boldsymbol{x}_t) \leq \left(\sqrt{f(\boldsymbol{x}_0)} - \frac{\sqrt{\mu}}{2}t\right)^2.$$

The small oscillations and spikes may potentially be explained by linearization around the stationary point, though in this case, linearization is quite tricky due to the singular nature of the stationary point [62].

The performance of Adam depends sensitively on the values of α and β. This has also been studied in [62]. Recall that $\alpha = 1 - a\eta$ and $\beta = 1 - b\eta$. Focusing on the region where a and b are not too large, three regimes with different behavior patterns were observed in the hyper-parameter space of (a, b):

1. **The spike regime**: This happens when b is sufficiently larger than a. In this regime large spikes appear in the loss curve, which makes the optimization process unstable.

2. **The oscillation regime**: This happens when a and b have similar magnitude (or in the same order). In this regime the loss curve exhibits fast and small oscillations. Small loss and stable loss curve can be achieved.

3. **The divergence regime**: This happens when a is sufficiently larger than b. In this regime the loss curve is unstable and usually diverges after some period of training. This regime should be avoided in practice since the training loss stays large.

In Figure 10 we show one typical loss curve for each regime for a typical neural network model.

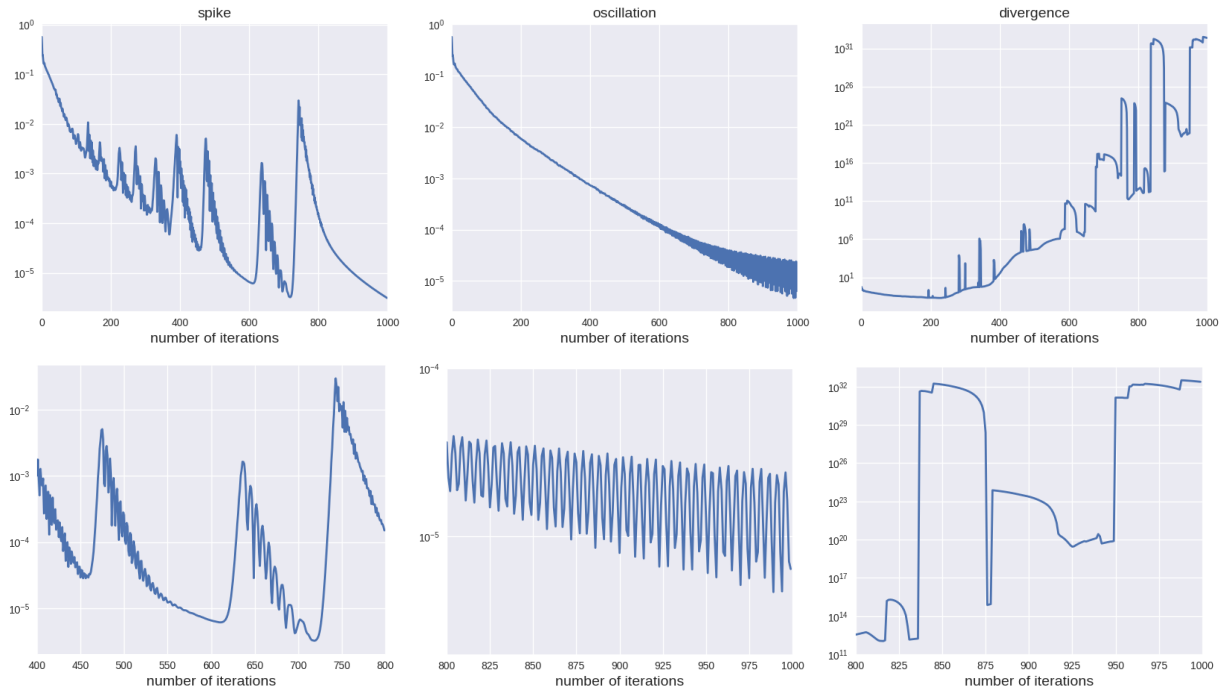


Figure 10: The three typical behavior patterns for Adam trajectories. Experiments are conducted on a fully-connected neural network with three hidden layers are $256, 128, 64$, respectively. The training data is taken from 2 classes of CIFAR-10 with 1000 samples per class. The learning rate is $\eta = 0.001$. The first row shows the loss curve of total 1000 iterations, the second row shows part of the loss curve (the last 200 iterations for oscillation and divergence regimes, and $400-800$ iterations for the spike regime). **Left:** $a = 1$, $b = 100$, large spikes appear in the loss curve; **Middle:** $a = 10$, $b = 10$, the loss is small and oscillates very fast, and the amplitude of the oscillation is also small; **Right:** $a = 100$, $b = 1$, the loss is large and blows up.

In Figure 11, we study Adam on a neural network model and show the final average training loss for different values of a and b. One can see that Adam can achieve very small loss in the oscillation regime. The algorithm is not stable in the spike regime and may blow up in the divergence regime. These observations suggest that in practice one should take $a \approx b$ with small values of a and b. Experiments on more complicated models, such as ResNet18, also support these conclusions [62].

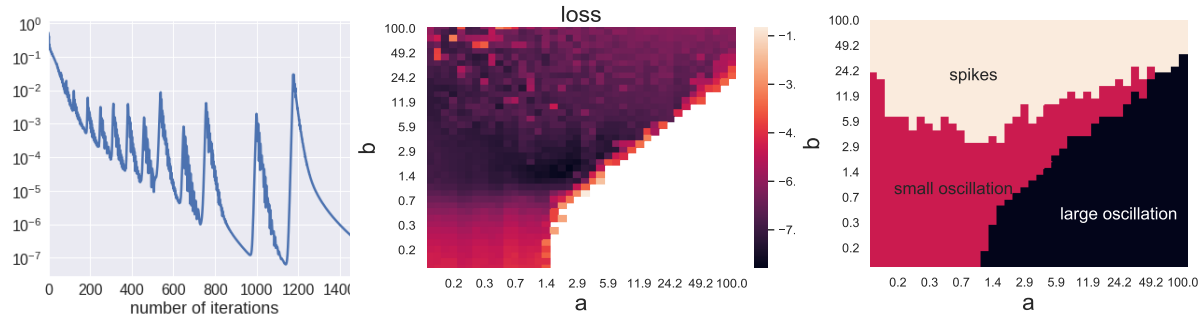


Figure 11: **Left:** The training curve of Adam for a multi-layer neural network model on CIFAR-10. The network has 3 hidden-layers whose widths are 256-256-128, with learning rate $1e-3$ and $(\alpha, \beta) = (0.9, 0.999)$ as default. 2 classes are picked from CIFAR-10 with 500 images in each class. Square loss function is used. **Middle:** Heat map of the average training loss (in logarithmic scale) of Adam on a multi-layer neural network model. The loss is the averaged over the last 1000 iterations. a and b range from 0.1 to 100 in logarithm scale. **Right:** The different behavior patterns.

5.7 Exploding and vanishing gradients for multi-layer neural networks

Exploding and vanishing gradients is one of the main obstacles for training of multi (many)-layer neural networks. Intuitively it is easy to see why this might be an issue. The gradient of the loss function with respect to the parameters involve a product of many matrices. Such a product can easily grow or diminish very fast as the number of products, namely the layers, increases.

Consider the multi-layer neural network with depth L:

$$\begin{aligned} \boldsymbol{z}_0 &= \boldsymbol{x}, \\ \boldsymbol{z}_l &= \sigma(W_l \boldsymbol{z}_{l-1} + \boldsymbol{c}_l), \quad l = 1, 2, \cdots, L, \\ f(\boldsymbol{x}; \theta) &= \boldsymbol{z}_L, \end{aligned} \tag{51}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the input data, σ is the ReLU activation and $\theta := (W_1, \boldsymbol{c}_1, \cdots, W_d, \boldsymbol{c}_L)$ denotes the collection of parameters. Here $W_l \in \mathbb{R}^{m_l \times m_{l-1}}$ is the weight, $\boldsymbol{c}_l \in \mathbb{R}^{m_l}$ is the bias. m_l is the width of the l-th hidden layer.

To make quantitative statements, we consider the case when the weights are i.i.d. random variables. This is typically the case when the neural networks are initialized. This problem has been studied in [41, 42]. Below is a brief summary of the results obtained in these papers.

Fix two collections of probability measures $\mu = (\mu^{(1)}, \mu^{(2)}, \cdots, \mu^{(L)})$, $\nu = (\nu^{(1)}, \nu^{(2)}, \cdots, \nu^{(L)})$ on $\mathbb{R}$ such that

- $\mu^{(l)}$, $\nu^{(l)}$ are symmetric around 0 for every $1 \le l \le L$;
- the variance of $\mu^{(l)}$ is $2/n_{l-1}$;
- $\nu^{(l)}$ has no atoms.

We consider the random network obtained by:

$$W_l^{i,j} \sim \mu^{(l)}, \quad \boldsymbol{c}_l^i \sim \nu^{(l)}, \qquad i.i.d. \tag{52}$$

for any $i = 1, 2, \cdots, m_l$ and $j = 1, 2, \cdots, m_{l-1}$, i.e. the weights and biases at layer l are drawn independently from $\mu^{(l)}$, $\nu^{(l)}$ respectively. Let

$$Z_{p,q} := \frac{\partial \boldsymbol{z}_L^q}{\partial \boldsymbol{x}^p}, \qquad p = 1, 2, \cdots, m_0 = d, \quad q = 1, 2, \cdots, m_L. \tag{53}$$

Theorem 42. *We have*

$$\mathbb{E}\left[Z_{p,q}^2\right] = \frac{1}{d}. \tag{54}$$

*In contrast, the fourth moment of $Z_{p,q}$ is **exponential** in $\sum_l \frac{1}{m_l}$:*

$$\frac{2}{d^2} \exp\left(\frac{1}{2}\sum_{l=1}^{L-1} \frac{1}{m_l}\right) \leq \mathbb{E}\left[Z_{p,q}^4\right] \leq \frac{C_\mu}{d^2} \exp\left(C_\mu \sum_{l=1}^{L-1} \frac{1}{m_l}\right), \tag{55}$$

where $C_\mu > 0$ is a constant only related to the (fourth) moment of $\mu = \{\mu^{(l)}\}_{l=1}^L$.

Furthermore, for any $K \in \mathbb{Z}_+$, $3 \leq K < \min_{1 \leq l \leq L-1} \{m_l\}$, we have

$$\mathbb{E}\left[Z_{p,q}^{2K}\right] \leq \frac{C_{\mu,K}}{d^K} \exp\left(C_{\mu,K} \sum_{l=1}^{L-1} \frac{1}{m_l}\right), \tag{56}$$

where $C_{\mu,K} > 0$ is a constant depending on K and the (first $2K$) moments of μ.

Obviously, by Theorem 42, to avoid the exploding and vanishing gradient problem, we want the quantity $\sum_{l=1}^{L-1} \frac{1}{m_l}$ to be small. This is also borne out from numerical experiments [42, 41].

5.8 What's not known?

There are a lot that we don't know about training dynamics. Perhaps the most elegant mathematical question is the global convergence of the mean-field equation for two-layer neural networks.

$$\partial_t \rho = \nabla(\rho \nabla V), \quad V = \frac{\delta \mathcal{R}}{\delta \rho}$$

The conjecture is that:

- if ρ_0 is a smooth distribution with full support, then the dynamics described by this flow should converge and the population risk $\mathcal{R}$ should converge to 0;
- if the target function f^* lies in the Barron space, then the Barron norm of the output function stays uniformly bounded.

Similar statements should also hold for the empirical risk.

Another core question is when two-layer neural networks can be trained efficiently [i]. The work [61, 77] shows that there exist target functions with Barron norms of size $poly(d)$, such that the training is exponentially slow in the statistical-query (SQ) setting [50]. A concrete example is $f^*(\boldsymbol{x}) = \sin(dx_1)$ with $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$, for which it is easy to verify that the Barron norm of f^* is $poly(d)$. However, [77] shows that the gradients of the corresponding neural networks are exponentially small, i.e. $O(e^{-d})$. Thus even for a moderate d, it is impossible to evaluate the gradients accurately on a finite-precision machine due to the floating-point error. Hence gradient-based optimizers are unlikely to succeed. This suggests that the Barron space is very likely too large for studying the training of two-layer neural networks. It is an open problem to identify the right function space, such that the functions can be learned in polynomial time by two-layer neural networks.

There are (at least) two possibilities for why the training becomes slow for certain (Barron) target functions in high dimension:

1. The training is slow in the continuous model due to the large parameter space or

2. The training is fast in the continuous model with dimension-independent rates, but the discretization becomes more difficult in high dimension.

It is unknown which of these explanations applies. Under strong conditions, it is known that parameters which are initialized according to a law π_0 and trained by gradient descent perform better at time $t > 0$ than parameters which are drawn from the distribution π_t given by the Wasserstein gradient flow starting at π_0 [19]. This might suggest that also gradient flows with continuous initial condition may not reduce risk at a dimension-independent rate.

One can ask similar questions for multi-layer neural networks and residual neural networks. However, it is much more natural to formulate them using the continuous formulation. We will postpone this to a separate article.

Even less is known for the training dynamics of neural network models under the conventional scaling, except for the high over-parametrized regime. This issue is all the more important since conventional scaling is the overwhelming scaling used in practice.

In particular, since the neural networks used in practice are often over-parametrized, and they seem to perform much better than random feature models, some form of implicit regularization must be at work. Identifying the presence and the mechanism of such implicit regularization is a very important question for understanding the success of neural network models.

We have restricted our discussion to gradient descent training. What about stochastic gradient descent and other training algorithms?

6 Concluding remarks

Very briefly, let us summarize the main theoretical results that have been established so far.

[i] The authors would like to thank Jason Lee for helpful conversations on the topic.

1. Approximation/generalization properties of hypothesis space:

    - Function spaces and quantitative measures for the approximation properties for various machine learning models. The random feature model is naturally associated with the corresponding RKHS. In the same way, the Barron norm is identified as the natural measure associated with two-layer neural network models (note that this is different from the spectral norm defined by Barron). For residual networks, the corresponding quantities are defined for the flow-induced spaces. For multi-layer networks, a promising candidate is provided by the multi-layer norm.
    - Generalization error estimates of regularized model. Dimension-independent error rates have been established for these models. Except for multi-layer neural networks, these error rates are comparable to Monte Carlo. They provide a way to compare these different machine learning models and serve as a benchmark for studying implicit regularization.

2. Training dynamics for highly over-parametrized neural network models:

    - Exponential convergence for the empirical risk.
    - Their generalization properties are no better than the corresponding random feature model or kernel method.

3. Mean-field training dynamics for two-layer neural networks:

    - If the initial distribution has full support and the GD path converges, then it must converge to a global minimum.

A lot has also been learned from careful numerical experiments and partial analytical arguments, such as:

- Over-parametrized networks may be able to interpolate any training data.
- The "double descent" and "slow deterioration" phenomenon for the random feature model and their effect on the corresponding neural network model.
- The qualitative behavior of adaptive optimization algorithms.
- The global minima selection mechanism for different optimization algorithms.
- The phase transition of the generalization properties of two-layer neural networks under the conventional scaling.

We have mentioned many open problems throughout this article. Besides the rigorous mathematical results that are called for, we feel that carefully designed numerical experiments should also be encouraged. In particular, they might give some insight on the difference between neural network models of different depth (for example, two-layer and three-layer neural neworks), and the difference between scaled and unscaled residual network models.

One very important issue that we did not discuss much is the continuous formulation of machine learning. We feel this issue deserves a separate article when the time is ripe.

Acknowledgement:. This work is supported in part by a gift to the Princeton University from iFlytek.

References

[1] Emmanuel Abbe and Colin Sandon. Poly-time universality and limitations of deep learning. *arXiv preprint arXiv:2001.02992*, 2020.

[2] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

[3] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[4] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. In *International Conference on Learning Representations*, 2018.

[5] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.

[6] Antonio Auffinger, Gérard Ben Arous, and Jiří Černỳ. Random matrices and complexity of spin glasses. *Communications on Pure and Applied Mathematics*, 66(2):165–201, 2013.

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

[8] Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *The Journal of Machine Learning Research*, 18(1):714–751, 2017.

[9] Jean Barbier, Florent Krzakala, Nicolas Macris, Léo Miolane, and Lenka Zdeborová. Optimal errors and phase transitions in high-dimensional generalized linear models. *Proceedings of the National Academy of Sciences*, 116(12):5451–5460, 2019.

[10] Andrew R Barron. Neural net approximation. In *Proc. 7th Yale Workshop on Adaptive and Learning Systems*, volume 1, pages 69–72, 1992.

[11] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[12] Andrew R Barron and Jason M Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv:1809.03090*, 2018.

[13] Peter L Bartlett, Olivier Bousquet, Shahar Mendelson, et al. Local Rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005.

[14] Peter L. Bartlett, David P. Helmbold, and Philip M. Long. Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks. *arXiv preprint arXiv:1802.06093*, 2018.

[15] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

[16] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[17] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[18] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[19] Zhengdao Chen, Grant Rotskoff, Joan Bruna, and Eric Vanden-Eijnden. A dynamical central limit theorem for shallow neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[20] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in neural information processing systems*, pages 3036–3046, 2018.

[21] Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. *arXiv preprint arXiv:2002.04486*, 2020.

[22] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

[23] Yaim Cooper. The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*, 2018.

[24] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[25] Wojciech Marian Czarnecki, Simon Osindero, Razvan Pascanu, and Max Jaderberg. A deep neural network's loss surface contains every low-dimensional pattern. *arXiv preprint arXiv:1912.07559*, 2019.

[26] Amit Daniely. SGD learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*, pages 2422–2430, 2017.

[27] Simon Du and Jason Lee. On the power of over-parametrization in neural networks with quadratic activation. In *International Conference on Machine Learning*, pages 1329–1338, 2018.

[28] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[29] Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI*, 2017.

[30] Weinan E, Chao Ma, and Qingcan Wang. A priori estimates of the population risk for residual networks. *arXiv preprint arXiv:1903.02154*, 2019.

[31] Weinan E, Chao Ma, and Lei Wu. Barron spaces and the compositional function spaces for neural network models. *arXiv preprint arXiv:1906.08039*, 2019.

[32] Weinan E, Chao Ma, and Lei Wu. A comparative analysis of the optimization and generalization property of two-layer neural network and random feature models under gradient descent dynamics. *Science China Mathematics, pages 1-24, 2020; arXiv:1904.04326*, 2019.

[33] Weinan E, Chao Ma, and Lei Wu. Machine learning from a continuous viewpoint. *arXiv preprint arXiv:1912.12777*, 2019.

[34] Weinan E, Chao Ma, and Lei Wu. A priori estimates of the population risk for two-layer neural networks. *Communications in Mathematical Sciences*, 17(5):1407–1425, 2019; arXiv:1810.06397, 2018.

[35] Weinan E and Stephan Wojtowytsch. Kolmogorov width decay and poor approximators in machine learning: Shallow neural networks, random feature models and neural tangent kernels. *arXiv:2005.10807 [math.FA]*, 2020.

[36] Weinan E and Stephan Wojtowytsch. On the Banach spaces associated with multi-layer ReLU networks of infinite width. *arXiv:2007.15623 [stat.ML]*, 2020.

[37] Weinan E and Stephan Wojtowytsch. Representation formulas and pointwise properties for barron functions. *arXiv preprint arXiv:2006.05982*, 2020.

[38] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940, 2016.

[39] Nicolas Fournier and Arnaud Guillin. On the rate of convergence in Wasserstein distance of the empirical measure. *Probability Theory and Related Fields*, 162(3-4):707–738, 2015.

[40] Sebastian Goldt, Galen Reeves, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. The gaussian equivalence of generative models for learning with two-layer neural networks. *arXiv preprint arXiv:2006.14709*, 2020.

[41] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? In *Advances in Neural Information Processing Systems*, pages 582–591, 2018.

[42] Boris Hanin and David Rolnick. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*, pages 571–581, 2018.

[43] 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.

[44] Sepp Hochreiter and Jurgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[45] Kaitong Hu, Zhenjie Ren, David Siska, and Lukasz Szpruch. Mean-field Langevin dynamics and energy landscape of neural networks. *arXiv:1905.07769 [math.PR]*, 2019.

[46] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[47] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8580–8589, 2018.

[48] Adel Javanmard, Marco Mondelli, and Andrea Montanari. Analysis of a two-layer neural network via displacement convexity. *arXiv preprint arXiv:1901.01375*, 2019.

[49] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in neural information processing systems*, pages 586–594, 2016.

[50] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[51] Nitish S. Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping T.P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *In International Conference on Learning Representations (ICLR)*, 2017.

[52] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[53] Jason M Klusowski and Andrew R Barron. Risk bounds for high-dimensional ridge function combinations including neural networks. *arXiv preprint arXiv:1607.01434*, 2016.

[54] Rohith Kuditipudi, Xiang Wang, Holden Lee, Yi Zhang, Zhiyuan Li, Wei Hu, Rong Ge, and Sanjeev Arora. Explaining landscape connectivity of low-cost solutions for multilayer nets. In *Advances in Neural Information Processing Systems*, pages 14601–14610, 2019.

[55] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[56] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[57] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

[58] Qianxiao Li, Long Chen, Cheng Tai, and Weinan E. Maximum principle based algorithms for deep learning. *The Journal of Machine Learning Research*, 18(1):5998–6026, 2017.

[59] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pages 2–47, 2018.

[60] Zhong Li, Chao Ma, and Lei Wu. Complexity measures for neural networks with general activation functions using path-based norms. *arXiv:2009.06132 [cs.LG]*, 2020.

[61] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in neural information processing systems*, pages 855–863, 2014.

[62] Chao Ma, Lei Wu, and Weinan E. A qualitative study of the dynamic behavior of adaptive gradient algorithms. *arXiv preprint arXiv:2009.06125*, 2020.

[63] Chao Ma, Lei Wu, and Weinan E. The quenching-activation behavior of the gradient descent dynamics for two-layer neural network models. *arXiv preprint arXiv:2006.14450*, 2020.

[64] Chao Ma, Lei Wu, and Weinan E. The slow deterioration of the generalization error of the random feature model. In *Mathematical and Scientific Machine Learning Conference*, 2020.

[65] VE Maiorov. On best approximation by ridge functions. *Journal of Approximation Theory*, 99(1):68–94, 1999.

[66] Vitaly Maiorov and Allan Pinkus. Lower bounds for approximation by MLP neural networks. *Neurocomputing*, 25(1-3):81–91, 1999.

[67] Y Makovoz. Uniform approximation by neural networks. *Journal of Approximation Theory*, 95(2):215–228, 1998.

[68] Stefano Sarao Mannelli, Eric Vanden-Eijnden, and Lenka Zdeborová. Optimization and generalization of shallow neural networks with quadratic activation functions. *arXiv preprint arXiv:2006.15459*, 2020.

[69] Robert J McCann. A convexity principle for interacting gases. *Advances in mathematics*, 128(1):153–179, 1997.

[70] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[71] Phan-Minh Nguyen and Huy Tuan Pham. A rigorous framework for the mean field limit of multilayer neural networks. *arXiv:2001.11443 [cs.LG]*, 2020.

[72] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8(1):143–195, 1999.

[73] Grant Rotskoff and Eric Vanden-Eijnden. Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks. In *Advances in neural information processing systems*, pages 7146–7155, 2018.

[74] Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer relu neural networks. In *International Conference on Machine Learning*, pages 4433–4441, 2018.

[75] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

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

[77] Ohad Shamir. Distribution-specific hardness of learning neural networks. *The Journal of Machine Learning Research*, 19(1):1135–1163, 2018.

[78] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *arXiv preprint arXiv:1808.09372*, 2018.

[79] Ivan Skorokhodov and Mikhail Burtsev. Loss landscape sightseeing with multi-point optimization. *arXiv preprint arXiv:1910.03867*, 2019.

[80] Mahdi Soltanolkotabi. Learning ReLUs via gradient descent. In *Advances in neural information processing systems*, pages 2007–2017, 2017.

[81] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[82] Grzegorz Swirszcz, Wojciech Marian Czarnecki, and Razvan Pascanu. Local minima in training of deep networks. *arXiv preprint arXiv:1611.06310*, 2016.

[83] Cheng Tai and Weinan E. Multiscale adaptive representation of signals: I. the basic framework. *The Journal of Machine Learning Research*, 17(1):4875–4912, 2016.

[84] Yuandong Tian. 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.

[85] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[86] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[87] Stephan Wojtowytsch. On the global convergence of gradient descent training for two-layer Relu networks in the mean field regime. *arXiv:2005.13530 [math.AP]*, 2020.

[88] Lei Wu, Chao Ma, and Weinan E. How SGD selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems*, pages 8279–8288, 2018.

[89] Lei Wu, Qingcan Wang, and Chao Ma. Global convergence of gradient descent for deep linear residual networks. In *Advances in Neural Information Processing Systems*, pages 13389–13398, 2019.

[90] Lenka Zdeborová and Florent Krzakala. Statistical physics of inference: Thresholds and algorithms. *Advances in Physics*, 65(5):453–552, 2016.

[91] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

A Proofs for Section 3.1

Proof of Theorem 6 Write the random feature model as

$$f_m(\boldsymbol{x}) = \int a\phi(\boldsymbol{x};\boldsymbol{w})\rho_m(da, d\boldsymbol{w}),$$

with

$$\rho_m(a,\boldsymbol{w}) = \frac{1}{m}\sum_{j=1}^{m}\delta(a-a_j)\delta(\boldsymbol{w}-\boldsymbol{w}_j).$$

Since supp $(\rho_m) \subseteq K := [-C, C]\times\Omega$, the sequence of probability measures (ρ_m) is tight. By Prokhorov's theorem, there exists a subsequence (ρ_{m_k}) and a probability measure $\rho^* \in \mathcal{P}(K)$ such that ρ_{m_k} converges weakly to ρ^*. Due to that $g(a,\boldsymbol{w};\boldsymbol{x}) = a\phi(\boldsymbol{x};\boldsymbol{w})$ is bounded and continuous with respect to $(a,\boldsymbol{w})$ for any $\boldsymbol{x}\in[0,1]^d$, we have

$$f^*(\boldsymbol{x}) = \lim_{k\to\infty}\int a\phi(\boldsymbol{x};\boldsymbol{w})\rho_{m_k}(da,d\boldsymbol{w}) = \int a\phi(\boldsymbol{x};\boldsymbol{w})\rho^*(da,d\boldsymbol{w}), \tag{57}$$

Denote by $a^*(\boldsymbol{w}) = \int a\rho^*(a|\boldsymbol{w})da$ the conditional expectation of a given $\boldsymbol{w}$. Then we have

$$f^*(\boldsymbol{x}) = \int a^*(\boldsymbol{w})\phi(\boldsymbol{x};\boldsymbol{w})\pi^*(d\boldsymbol{w}),$$

where $\pi^*(\boldsymbol{w}) = \int\rho(a,\boldsymbol{w})da$ is the marginal distribution. Since supp $(\rho^*)\subseteq K$, it follows that $|a^*(\boldsymbol{w})|\le C$. For any bounded and continuous function $g:\Omega\mapsto\mathbb{R}$, we have

$$\begin{aligned}\int g(\boldsymbol{w})\pi^*(\boldsymbol{w})d\boldsymbol{w} &= \int g(\boldsymbol{w})\rho^*(a,\boldsymbol{w})d\boldsymbol{w}da &(58)\\ &= \lim_{k\to\infty}\int g(\boldsymbol{w})\rho_m^*(a,\boldsymbol{w})dad\boldsymbol{w} &(59)\\ &= \lim_{k\to\infty}\frac{1}{m}\sum_{j=1}^{m_k} g(\boldsymbol{w}_j^0) = \int g(\boldsymbol{w})\pi_0(\boldsymbol{w})d\boldsymbol{w}. &(60)\end{aligned}$$

By the strong law of large numbers, the last equality holds with probability 1. Taking $g(\boldsymbol{w}) = a^*(\boldsymbol{w})\phi(\boldsymbol{x};\boldsymbol{w})$, we obtain

$$f^*(\boldsymbol{x}) = \int a^*(\boldsymbol{w})\phi(\boldsymbol{x};\boldsymbol{w})\pi_0(\boldsymbol{w})d\boldsymbol{w}.$$

To prove Theorem 7, we first need the following result ([33, Proposition 7]).

Proposition 43. *Given training set* $\{(\boldsymbol{x}_i, f(\boldsymbol{x}_i))\}_{i=1}^n$, *for any* $\delta \in (0,1)$, *we have that with probability at least* $1-\delta$ *over the random sampling of* $\{\boldsymbol{w}_j^0\}_{j=1}^m$, *there exists* $\boldsymbol{a} \in \mathbb{R}^m$ *such that*

$$\begin{aligned} \hat{\mathcal{R}}_n(\boldsymbol{a}) &\leq \frac{2\log(2n/\delta)}{m}\|f\|_{\mathcal{H}}^2 + \frac{8\log^2(2n/\delta)}{9m^2}\|f\|_\infty^2, \\ \frac{\|\boldsymbol{a}\|^2}{m} &\leq \|f\|_{\mathcal{H}}^2 + \sqrt{\frac{\log(2/\delta)}{2m}}\|f\|_\infty^2 \end{aligned} \tag{61}$$

Proof of Theorem 7 Denote by $\tilde{\boldsymbol{a}}$ the solution constructed in Proposition 43. Using the definition, we have

$$\hat{\mathcal{R}}_n(\hat{\boldsymbol{a}}_n) + \frac{\|\hat{\boldsymbol{a}}_n\|}{\sqrt{nm}} \leq \hat{\mathcal{R}}_n(\tilde{\boldsymbol{a}}) + \frac{\|\tilde{\boldsymbol{a}}\|}{\sqrt{nm}}.$$

Hence, $\frac{\|\hat{\boldsymbol{a}}_n\|}{\sqrt{m}} \leq C_{n,m} := \frac{\|\tilde{\boldsymbol{a}}\|}{\sqrt{m}} + \sqrt{n}\hat{\mathcal{R}}_n(\tilde{\boldsymbol{a}})$. Define $\mathcal{H}_C := \{\frac{1}{m}\sum_{j=1}^m a_j\phi(\boldsymbol{x};\boldsymbol{w}_j^0) : \frac{\|\boldsymbol{a}\|}{\sqrt{m}} \leq C\}$. Then, we have $\mathrm{Rad}_S(\mathcal{H}_C) \lesssim \frac{C}{\sqrt{n}}$ [76]. Moreover, for any $\delta \in (0,1)$, with probability $1-\delta$ over the choice of training data, we have

$$\begin{aligned} \mathcal{R}(\hat{\boldsymbol{a}}_n) &\lesssim \hat{\mathcal{R}}_n(\hat{\boldsymbol{a}}_n) + \mathrm{Rad}_S(\mathcal{H}_{C_{n,m}}) + \sqrt{\frac{\log(2/\delta)}{n}} \\ &\leq \hat{\mathcal{R}}_n(\tilde{\boldsymbol{a}}) + \frac{\|\tilde{\boldsymbol{a}}\|}{\sqrt{nm}} + \frac{\|\tilde{\boldsymbol{a}}\|}{\sqrt{nm}} + \hat{\mathcal{R}}_n(\tilde{\boldsymbol{a}}) + \sqrt{\frac{\log(2/\delta)}{n}} \\ &\leq 2\hat{\mathcal{R}}_n(\tilde{\boldsymbol{a}}) + 2\frac{\|\tilde{\boldsymbol{a}}\|}{\sqrt{nm}} + \sqrt{\frac{\log(2/\delta)}{n}}. \end{aligned}$$

By inserting Eqn. (61), we complete the proof. □

B Proofs for Section 3.2

Proof of Theorem 12. Let $f(\boldsymbol{x}) = \mathbb{E}_{(a,w)\sim\rho}\big[a\,\sigma(\boldsymbol{w}^T\boldsymbol{x})\big]$. Using the homogeneity of the ReLU activation function, we may assume that $\|\boldsymbol{w}\|_{\ell^1} \equiv 1$ and $|a| \equiv \|f\|_{\mathcal{B}}$ ρ-almost everywhere. By [76, Lemma 26.2], we can estimate

$$\mathbb{E}_{(a,\boldsymbol{w})\sim\rho^m}\left[\sup_{\boldsymbol{x}\in[0,1]^d}\frac{1}{m}\sum_{i=1}^m\big(a_i\,\sigma(\boldsymbol{w}_i^T\boldsymbol{x}) - f(\boldsymbol{x})\big)\right] \leq 2\,\mathbb{E}_{(a,w)\sim\pi^m}\mathbb{E}_\xi\left[\sup_{\boldsymbol{x}\in[0,1]^d}\frac{1}{m}\sum_{i=1}^m\xi_i\,a_i\,\sigma(\boldsymbol{w}_i^T\boldsymbol{x})\right],$$

where $\xi_i = \pm 1$ with probability $1/2$ independently of ξ_j are Rademacher variables. Now we bound

$$\begin{aligned}
\mathbb{E}_\xi \left[\sup_{\boldsymbol{x}\in[0,1]^d} \frac{1}{m} \sum_{i=1}^m \xi_i \, a_i \, \sigma(\boldsymbol{w}_i^T \boldsymbol{x}) \right] &= \mathbb{E}_\xi \left[\sup_{\boldsymbol{x}\in[0,1]^d} \frac{1}{m} \sum_{i=1}^m \xi_i \, |a_i| \, \sigma\big(\boldsymbol{w}_i^T \boldsymbol{x}\big) \right] \\
&= \mathbb{E}_\xi \left[\sup_{\boldsymbol{x}\in[0,1]^d} \frac{1}{m} \sum_{i=1}^m \xi_i \, \sigma\big(|a_i| \, \boldsymbol{w}_i^T \boldsymbol{x}\big) \right] \\
&\leq \mathbb{E}_\xi \left[\sup_{\boldsymbol{x}\in[0,1]^d} \frac{1}{m} \sum_{i=1}^m \xi_i \, |a_i| \, \boldsymbol{w}_i^T \boldsymbol{x} \right] \\
&= \frac{1}{m} \mathbb{E}_\xi \left[\sup_{\boldsymbol{x}\in[0,1]^d} \boldsymbol{x}^T \sum_{i=1}^m \xi_i \, |a_i| \, \boldsymbol{w}_i \right] \\
&= \frac{1}{m} \mathbb{E}_\xi \left\| \sum_{i=1}^m \xi_i \, |a_i| \, \boldsymbol{w}_i \right\|_{\ell^1} .
\end{aligned}$$

In the first line, we used the symmetry of ξ_i to eliminate the sign of a_i, which we then take into the (positively one-homogenous) ReLU activation function. The next inequality follows from the contraction lemma for Rademacher complexities, [76, Lemma 26.9], while the final equality follows immediately from the duality of ℓ^1- and ℓ^∞-norm. Recalling that

$$\|\boldsymbol{y}\|_{\ell^2} \leq \|\boldsymbol{y}\|_{\ell^1} \leq \sqrt{d+1} \, \|\boldsymbol{y}\|_{\ell^2} \quad \forall \, \boldsymbol{y} \in \mathbb{R}^{d+1}, \qquad \|a_i \, \boldsymbol{w}_i\|_{\ell^1} = \|f\|_{\mathcal{B}} \quad \forall \, 1 \leq i \leq m$$

we bound

$$\begin{aligned}
\mathbb{E}_{(a,\boldsymbol{w})\sim\rho^m} \left[\sup_{\boldsymbol{x}\in[0,1]^d} \frac{1}{m} \sum_{i=1}^m \big(a_i \, \sigma(\boldsymbol{w}_i^T \boldsymbol{x}) - f(\boldsymbol{x}) \big) \right] &\leq 2 \sup_{\|\boldsymbol{y}_i\|_{\ell^1} \leq \|f\|_{\mathcal{B}}} \mathbb{E}_\xi \left\| \frac{1}{m} \sum_{i=1}^m \xi_i \boldsymbol{y}_i \right\|_{\ell^1} \\
&\leq 2 \, \|f\|_{\mathcal{B}} \sup_{\|\boldsymbol{y}_i\|_{\ell^1} \leq 1} \mathbb{E}_\xi \left\| \frac{1}{m} \sum_{i=1}^m \xi_i \boldsymbol{y}_i \right\|_{\ell^1} \\
&\leq 2\sqrt{d+1} \, \|f\|_{\mathcal{B}} \sup_{\|\boldsymbol{y}_i\|_{\ell^2} \leq 1} \mathbb{E}_\xi \left\| \frac{1}{m} \sum_{i=1}^m \xi_i \boldsymbol{y}_i \right\|_{\ell^2} \\
&\leq 2 \, \|f\|_{\mathcal{B}} \sqrt{\frac{d+1}{m}}
\end{aligned}$$

by using the Rademacher complexity of the unit ball in Hilbert spaces [76, Lemma 26.10]. Applying the same argument to $-\big[\frac{1}{m} \sum_{i=1}^m a_i \sigma(\boldsymbol{w}_i^T \boldsymbol{x}) - f(\boldsymbol{x})\big]$, we find that

$$\mathbb{E}_{(a,w)\sim\pi^m} \sup_{\boldsymbol{x}\in[0,1]^d} \left| \frac{1}{m} \sum_{i=1}^m \big(a_i \, \sigma(\boldsymbol{w}_i^T \boldsymbol{x}) - f(\boldsymbol{x}) \big) \right| \leq 4 \, \|f\|_{\mathcal{B}} \sqrt{\frac{d+1}{m}}.$$

In particular, there exists weights $(a_i, \boldsymbol{w}_i)_{i=1}^m$ such that the inequality is true.

□