Some remarks about deep neural networks

Mark Peletier

September 19, 2022

Contents

Chapter 1

Introduction

There is a vast literature on machine learning in general and neural networks in particular. For many years the 'bible' in this field was the book by Goodfellow, Bengio, and Courville [GBC16], and this book still is a great place to start. More mathematical treatments can be found inn e.g. [SSBD14, CC18, DHP21].

1.1 Basic definitions

The simplest type of neural network is the *feedforward* network, also called a *multi-layer perceptron.*

Definition 1.1.1. A *Feed-forward neural network* is a tuple $(\mathbb{R}^d, \mu, \mathcal{F}, g)$ where

- $\mu \in \mathcal{P}(\mathbb{R}^d)$ is the 'data measure';
- $\mathcal{F}$ is a class of functions from $\mathbb{R}^d$ to $\mathbb{R}$;
- g is a given function from $\mathbb{R}^d$ to $\mathbb{R}$.

Given this tuple, the *cost* or *risk* or *loss function* is defined as

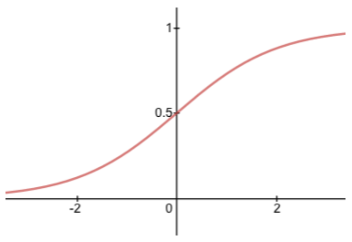
$$\mathcal{R} : \mathcal{F} \to \mathbb{R}, \qquad \mathcal{R}(f) := \int_{\mathbb{R}^d} \big|f(x) - g(x)\big|^2 \,\mu(dx).$$

The *optimization problem* is to find the minimizer of $\mathcal{R}$ over $\mathcal{F}$:

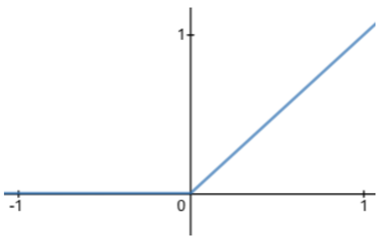
$$\min_{f\in\mathcal{F}} \mathcal{R}(f). \tag{1.1}$$

The name 'neural network' in this definition derives from a specific choice for the set of functions $\mathcal{F}$, which we now describe.

This is a modified version of lecture notes that were collectively written during a 'Research Topic' on Neural Networks at Eindhoven University of Technology in the first half of 2021. Much of it was written by me, but various bits have been contributed by other authors; in particular I am grateful to my co-lecturer Oxana Manita, and (in alphabetical order) Thijs Beurskens, Deen Colenbrander, Thijs Jenneskens, Sanne van Kempen, Jan Moraal, Georg Prokert, Antonio Reggio, and Paul Sanders. Of course I take full responsibility for any mistakes that there might be in these notes.



(a) Sigmoid $\sigma(x) = 1/(1+e^{-x})$



(b) Rectified Linear Unit (ReLU) $\sigma(x) = x_+$

Figure 1.1: Example activation functions

Definition 1.1.2. An L-layer neural network is a class of functions $\mathcal{F}$ such that each $f \in \mathcal{F}$ is of the form

$$f_\theta(x) := \Phi_\theta^{(L)} \circ \Phi_\theta^{(L-1)} \circ \cdots \circ \Phi_\theta^{(1)}(x),$$

where

- $\sigma : \mathbb{R} \to \mathbb{R}$ is called the 'activation function';
- for each $\ell = 1, \ldots, L$, $\Phi_\theta^{(\ell)}$ is a map from $\mathbb{R}^{d_{\ell-1}}$ to $\mathbb{R}^{d_\ell}$; here $(d_\ell)_\ell$ are a set of dimensions, with $d_0 = d$ and $d_L = 1$. $\Phi_\theta^{(\ell)}$ has the structure

  $$\Phi_\theta^{(\ell)}(z) := \sigma\Big(W^{(\ell)} z + b^{(\ell)}\Big), \qquad z \in \mathbb{R}^{d_{\ell-1}},$$

  or equivalently in coordinates,

  $$\big(\Phi_\theta^{(\ell)}(z)\big)_i := \sigma\left(\sum_{j=1}^{d_{\ell-1}} W_{ij}^{(\ell)} z_j + b_i^{(\ell)}\right), \qquad i = 1, \ldots, d_\ell, \qquad z \in \mathbb{R}^{d_{\ell-1}}.$$

  Here

  $$W^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell-1}} \qquad \text{and} \qquad b^{(\ell)} \in \mathbb{R}^{d_\ell}.$$

  Note that the scalar nonlinearity $\sigma : \mathbb{R} \to \mathbb{R}$ is applied coordinate-wise, i.e. if $z \in \mathbb{R}^n$ then $\sigma(z)$ is the n-vector with i^{th} coordinate $\sigma(z_i)$.
- The 'weights' $W^{(\ell)}$ and 'biases' $b^{(\ell)}$ are collected all together in the big parameter vector θ, and the subscript θ indicates dependence on (some of) the weights and biases.

Typical choices for σ are the 'sigmoid' function and the 'rectified linear unit' function (see Figure 1.1).

For fixed choice σ, fixed number of layers L and fixed dimensions d_ℓ, the elements of the class $\mathcal{F}$ are fully determined by the parameter vector θ. In a practical sense the optimization problem (1.1) then reduces to

$$\min_{\theta \in \Theta} \mathcal{R}(f_\theta)$$

where Θ is some set of admissible parameter values. Note that although $\mathcal{R}$ is a convex function of f (even a quadratic function of f), the nonlinearity of σ makes $\mathcal{R}(f_\theta)$ a non-convex function of θ.

Terminology

- We use the term *network architecture* for the structure of the functions $\Phi^{(\ell)}$, $\ell = 1, \dots, L$, without taking into account the specific parameter point θ;
- We use the term *network* for a network architecture at a particular parameter point θ.

Therefore a *network architecture* can generate many functions of z; a *network* implements exactly one function of z.

1.2 Sampling

In practice, μ and g are unknown, and instead we have access to a finite set of samples $(x_i, g(x_i)) =: (x_i, y_i)$. Since we do not know how to calculate $\mathcal{R}(f)$, the best we can do is calculate the *empirical loss*,

$$\mathcal{R}_N(f) := \frac{1}{N}\sum_{i=1}^{N} \bigl|f(x_i) - y_i\bigr|^2.$$

This choice can also be seen as replacing the true data measure μ in $\mathcal{R}$ by the empirical approximation

$$\mu_N(dx) := \frac{1}{N}\sum_{i=1}^{N} \delta_{x_i}(dx).$$

If x_i are i.i.d. samples of μ, then as N tends to infinity, μ_N converges to μ in the sense of measures (this is known as the Glivenko-Cantelli theorem [Dud02, Th. 11.4.1]; you can think of it as the 'law of large numbers for measures'). This implies that for a fixed $f \in \mathcal{F}$ we expect $\lim_{N\to\infty} \mathcal{R}_N(f) = \mathcal{R}(f)$.

On the other hand, for a fixed set of samples, minimizing $\mathcal{R}_N$ over $\mathcal{F}$ may produce a minimizer f that performs arbitrarily badly on points x that are not in the set $\{x_i\}_{i=1}^N$. If the support of μ is much larger than this set of points $\{x_i\}$, then f may also perform arbitrarily badly in $\mathcal{R}$. This phenomenon is called *overfitting*, and avoiding overfitting is a major challenge in all of machine learning. (See e.g. [GBC16, Ch. 7] for a discussion of the many ways that one may try to prevent it).

1.3 Training and testing

A common method to help deal with the difference between $\mathcal{R}$, which we want to minimize, and $\mathcal{R}_N$, which we actually minimize, is to divide the samples into a *training* and a *testing* set; this leads to two empirical loss functions, $\mathcal{R}_N^{\text{training}}$ and $\mathcal{R}_N^{\text{testing}}$. One then constructs an algorithm to iteratively reduce $\mathcal{R}_N^{\text{training}}$, while keeping track of the value of $\mathcal{R}_N^{\text{testing}}$. Figure 1.2 shows an example.

This approach is based on the following idea: the fact that the testing dataset is not used in the optimization algorithm means that we can consider $\mathcal{R}_N^{\text{testing}}$ to be a better indicator of $\mathcal{R}$ than $\mathcal{R}_N^{\text{training}}$. It does remain a very imperfect indicator of $\mathcal{R}$.

1.4 Examples

Example 1: Linear regression. If σ is affine, then for any choice of L and θ, f_θ is a linear map from $\mathbb{R}^d$ to $\mathbb{R}$. Minimizing $\mathcal{R}_N(f_\theta)$ over θ then corresponds to finding the best fit (in the sense of least

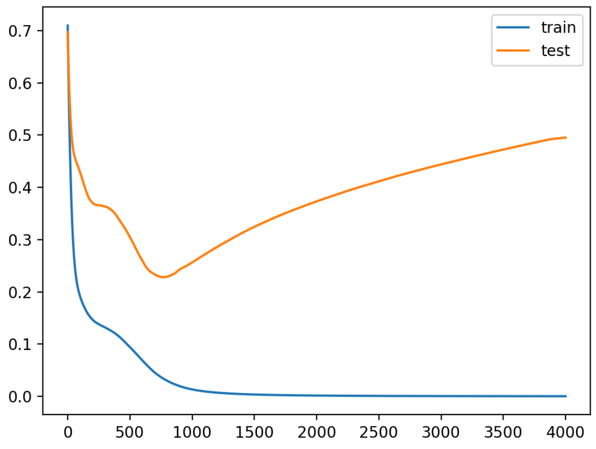


Figure 1.2: An example of loss curves for training and testing data, showing the joint decrease in early stages and divergence in later stages. Source: `https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/`.

squares) of the function $y = f_\theta(x)$ to the point cloud $\{(x_i, y_i)\}_{i=1}^N$; this is known as *linear regression*, because the dependence on the parameters θ is linear. (In this specific case, the dependence on x also is linear, but from the point of view of optimization that linearity plays no role).

Example 2: A single neuron. A network consisting of a single neuron generates all functions of the form

$$f_\theta(x) := \sigma(w^T x + b),$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$. If σ is the ReLU function, then each of these functions is continuous and piecewise affine, with one 'piece' being the zero function and the other piece an affine function.

Example 3: A parallel collection of neurons: one-and-a-half-layer networks. One can combine multiple neurons together in a function of the form

$$f_\theta(x) := \sum_{i=1}^{d_1} W_{1i}^{(2)} \sigma\left(\sum_{j=1}^{d_0} W_{ij}^{(1)} x_j + b_i^{(1)}\right) \tag{1.2}$$

I call this 'one-and-a-half layer' since the second-layer activation function and biases are missing (or have been chosen to be the identity and zero).

Since each term in the sum over i is a continuous piecewise affine function of x, this function f_θ again is a continuous piecewise affine function of x. In fact, if the input dimension d is 1, then it is easy to see that this class of functions can represent any continuous piecewise affine function from $\mathbb{R}$ to $\mathbb{R}$ with at most d_1 affine pieces (and even $d_1 + 1$ pieces, if one of them is zero).

Example 4: A full two-layer network. A two-layer network has the structure

$$\begin{aligned} f_\theta(x) &:= \sigma\left(\sum_{i=1}^{d_1} W_{1i}^{(2)} \sigma\left(\sum_{j=1}^{d_0} W_{ij}^{(1)} x_j + b_i^{(1)}\right) + b_1^{(2)}\right) \\ &= \sigma\Big(W^{(2)} \sigma\big(W^{(1)} x + b^{(1)}\big) + b^{(2)}\Big). \end{aligned}$$

The second formulation above makes use of matrix-vector-multiplication notation to make the expression easier to read.

Chapter 2

Approximation theory

2.1 Basic questions

The basic questions in approximation theory are the following. Given a metric space $(\mathcal{X}, d)$ and a subset $F \subset \mathcal{X}$,

1. Is F dense in $\mathcal{X}$?
2. If F is dense, and $F = \bigcup_{n\in\mathbb{N}} F_n$, then how well can we approximate elements of $\mathcal{X}$ by F_n? This might be characterized by the *best approximation*

   $$E_n(g) := \inf_{f\in F_n} d(f, g) = \operatorname{dist}(g, F_n), \qquad \text{for } g \in \mathcal{X}.$$

   One question would be to determine properties of g that are sufficient for $E_n(g)$ to decay at a certain rate as $n \to \infty$,
3. Conversely, given $g \in \mathcal{X}$, can we deduce properties of g from the decay rate of the sequence $(E_n(g))_n$?

The literature on approximation theory is vast. A good reference for the specific case of neural networks is the recent review by DeVore, Hanin, and Petrova [DHP21].

2.2 Regularity, dimension, and approximation rate

As a general principle, functions with higher regularity can be approximated more accurately with a limited number of components; but the rate deteriorates when increasing the dimension. This can be seen in a number of ways:

1. If f is a k times differentiable and periodic function on $[0, 1]$, and F_n the sum of the first n terms of the Fourier series, then $\|f - F_n\|_2$ decays as n^{-k} and $\|f - F_n\|_\infty$ as n^{-k+1} (see e.g. [Isk18, Theorems 6.44 and 6.47]);
2. In the same case but on $[0, 1]^d$, the approximation quality worsens; the error with n terms decays as $n^{-k/d}$ and $n^{-(k-1)/d}$.

3. If f is a function on $\mathbb{R}^d$ that is analytic in $B(0,2)$ and F_m its Taylor series truncated at order m, then F_m converges uniformly on $B(0,1)$ at *exponential* rate, e^{-cm} for some c depending on f. Note that since F_m contains $n = O(m^d)$ terms, this rate becomes $e^{-cn^{1/d}}$ when written as function of the number of terms.

4. For estimation based on sampling very similar rates hold. Consider a pair of random variables $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$, and set $f(X) := \mathbb{E}[Y|X]$, $f : \mathbb{R}^d \to \mathbb{R}$. A classical result by Stone [Sto82] states that if one only assumes that f is differentiable of order k, then the optimal rate of convergence of any estimator of f that is based on n samples of (X, Y) is $n^{-k/(2k+d)}$.

These examples show a general structure: for n 'components' in an approximation, one expects an error of the size of $n^{-k/d}$, where k is the number of derivatives of the function and d the dimension of the underlying space. For infinite smoothness $k = \infty$, the analytic case, the 'infinitely high power' $n^{-k/d}$ becomes $e^{-cn^{1/d}}$.

Sometimes these rates are given in opposite form: to achieve an error of size $\varepsilon > 0$, one needs $\varepsilon^{-d/k}$ or $\big(\log(1/\varepsilon)\big)^d$ terms.

2.3 Cybenko and Hornik

This section and many of the following sections revolve around the class of 'one-hidden-layer networks', also called 'shallow networks'. Above we called these 'one-and-a-half-layer networks' (see (1.2)), because they consist of a nonlinear hidden layer and a linear output layer. In shorthand notation these are functions of the form

$$f : \mathbb{R}^d \to \mathbb{R}, \qquad f(x) = \sum_{j=1}^{n} \alpha_j \sigma(w_j^T x + b_j) \tag{2.1}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$, $w_j \in \mathbb{R}^d$ for each j, and $\alpha_j, b_j \in \mathbb{R}$.

The central problem of this section is the following:

Under which conditions is the set

$$\mathcal{G} := \left\{ f : \mathbb{R}^d \to \mathbb{R} : f(x) = \sum_{j=1}^{N} \alpha_j \sigma\big(w_j^T x + b_j\big) \right\} \tag{2.2}$$

dense in $C_b([0,1]^d)$? Or in other sets?

This problem is known as the 'universal approximation question' and is treated by Cybenko [Cyb89] and Hornik [Hor91], as well as many others.

2.3.1 Initial results

A central concept is that of a *discriminatory function.* Set $I_d := [0,1]^d$ and let $\mathcal{M}(I_d)$ be the set of finite signed regular Borel measures[1].

[1]Recall that a Borel measure is a map $\mu : \mathcal{B} \to \mathbb{R}$ defined on the Borel σ-algebra $\mathcal{B}(I_d)$; the *finiteness* means that $\mu(A)$ is finite (not $\pm\infty$) on each $A \in \mathcal{B}$. A *regular* Borel measure is a measure that satisfies the regularity condition that each $A \subset \mathcal{B}$ can be approximated from inside and outside by open and closed sets:

$$\sup\{|\mu|(C) : C \subset A \text{ closed }\} = |\mu|(A) = \inf\{|\mu|(O) : O \supset B \text{ open }\}.$$

Definition 2.3.1. A function $\sigma : \mathbb{R} \to \mathbb{R}$ is *discriminatory* if for $\mu \in \mathcal{M}(I_d)$,

$$\left[\int_{I_d} \sigma\big(y^T x + \theta\big)\mu(dx) = 0 \quad \forall y \in \mathbb{R}^d, \theta \in \mathbb{R} \right] \quad \Longrightarrow \quad \mu = 0.$$

A central result in both Cybenko and Hornik's work is the following.

Theorem 2.3.2. *Let σ be continuous and discriminatory. Then the set $\mathcal{G}$ is dense in $C_b(I_d)$.*

Proof. We prove that the annihilator $\mathcal{G}^\perp := \{\xi \in C_b(I_d)^* : \langle \xi, f \rangle = 0\}$ is equal to $\{0\}$; since $\mathcal{G}$ is a linear subspace, this proves that $\mathcal{G}$ is dense in $C_b(I_d)$ (see e.g. [Bre11, Cor. 1.8]).

Let $\xi \in \mathcal{G}^\perp$. By the Riesz Representation theorem (e.g. [Rud87, Th. 6.19]) there exists a $\mu \in \mathcal{M}(I_d)$ such that

$$\langle \xi, f \rangle = \int_{I_d} f(x)\, \mu(dx) \qquad \text{for all } f \in C_b(I_d).$$

Applying ξ to functions $f(x) = \sigma(y^T x + \theta) \in \mathcal{G}$ we find that

$$\int_{I_d} \sigma(y^T x + \theta)\, \mu(dx) = 0 \qquad \text{for all } y \in \mathbb{R}^d, \theta \in \mathbb{R}.$$

Since σ is assumed to be discriminatory, it follows that $\mu = 0$ and therefore $\xi = 0$. $\square$

A central subsequent result in [Cyb89] then is

Lemma 2.3.3. *If $\sigma : \mathbb{R} \to \mathbb{R}$ is Lebesgue measurable and bounded, and is* sigmoidal, *i.e.*

$$\sigma(x) \longrightarrow \begin{cases} 1 & \text{as } x \to \infty \\ 0 & \text{as } x \to -\infty \end{cases}$$

then it is discriminatory.

The combination of this lemma with the theorem above gives the main result:

Corollary 2.3.4. *Let σ be continuous and sigmoidal; then $\mathcal{G}$ is dense in $C_b(I_d)$.*

2.3.2 Decision regions

Cybenko then uses this corollary to show that one-hidden-layer networks can approximate any measurable 'decision regions' to arbitrary accuracy.

Let I_d be partitioned into measurable sets $P_1, \ldots, P_k$. We define the *decision function* $g : I_d \to \{1, \ldots, k\}$ to be the function g that maps any $x \in P_k$ to k.

Theorem 2.3.5. *Let σ be a continuous sigmoidal function, and let g be a decision function as above. Let $\varepsilon > 0$.*

Then there exists a subset $D \subset I_d$ such that

- *the Lebesgue measure of D is larger than $1 - \varepsilon$;*
- *$|f(x) - g(x)| < \varepsilon$ for all $x \in D$.*

2.3.3 Additions by Hornik

Hornik [Hor91] then adds a few further results. First, he strengthens the characterization of discriminatory functions:

Lemma 2.3.6. *Any bounded non-constant measurable function is discriminatory.*

Hornik also discusses density in L^p spaces with respect to some fixed finite non-negative measure μ on $\mathbb{R}^d$. Note that now the domain is $\mathbb{R}^d$ rather than I_d.

Theorem 2.3.7. *Let σ be bounded and non-constant. Then $\mathcal{G}$ is dense in $L^p(\mu)$.*

For density in the space of continuous functions Hornik extends the result to all compact subsets X of $\mathbb{R}^d$:

Theorem 2.3.8. *Let σ be continuous, bounded, and non-constant, and let X be a compact subset of $\mathbb{R}^d$. Then $\mathcal{G}$ is dense in $C_b(X)$.*

Of the two improvements in this theorem with respect to Cybenko's Corollary 2.3.4, only the generalization to general non-constant σ is significant; the other generalization, to general non-compact sets X, can be done in Cybenko's proof with only notational changes.

2.3.4 More general universal approximation results

Leshno, Lin, Pinkus, and Schocken [LLPS93] go beyond the sufficiency results by Cybenko and Hornik in three ways:

1. They consider activations σ that are not necessarily bounded (and therefore allow for e.g. the ReLU function);
2. They also allow for sup-norm approximation of continuous functions by non-continuous approximants (by having σ non-continuous);
3. They give a full equivalence, showing that density is *equivalent* to the property that the activation function is not a polynomial.

Deep in the bowels of the proof the following property is used:

Theorem 2.3.9. *Let M be the set of all Lebesgue measurable and locally bounded functions such that the closure of the set of points of discontinuity has zero Lebesgue measure.*

If $\sigma \in M$, then $\mathcal{F}_n := span\{\sigma(w \cdot x + \theta) : w \in \mathbb{R}^n, \theta \in \mathbb{R}\}$ is dense in $C(\mathbb{R}^n)$ if and only if σ is not a polynomial (a.e.).

Here density in $C(\mathbb{R}^n)$ is defined in the sense of sup-norm approximation on each compact set.

Proposition 2.3.10. *Let $\mu \geq 0$ be a finite measure on $\mathbb{R}^n$ with compact support, $\mu \ll \mathcal{L}$. Then $\mathcal{F}_n$ is dense in $L^p(\mu)$ ($1 \leq p < \infty$) if and only if σ is not a polynomial (a.e.).*

The proof of Proposition 2.3.10 follows from Theorem 2.3.9.

Remark 2.3.11. The statement and proof of Theorem 2.3.9 concern 1-hidden layer networks, but can be easily extended to deep networks in the following way. The function is approximated already in the first hidden layer, and the further layers are tuned to approximate the identity function. This is not a reasonable NN structure from the practitioner's point of view: in practice we want nodes to distinguish meaningful features of the input data. $\square$

2.3.5 Outline of the proof of Theorem 2.3.9

For any $f : \mathbb{R} \to \mathbb{R}$ we define the sets

$$A(f) := \overline{\operatorname{span}\{x \mapsto f(wx+\theta) : w, \theta \in \mathbb{R}\}}$$
$$A_n(f) = \overline{\operatorname{span}\{x \mapsto f(w \cdot x+\theta) : w \in \mathbb{R}^n, \theta \in \mathbb{R}\}}$$

The closures above are in the supremum-norm-on-compact-sets mentioned above. In terms of these sets, Theorem 2.3.9 states that for any $\sigma \in M$, we have that σ is non-polynomial iff $A_n(\sigma) = C(\mathbb{R}^n)$.

1. It is sufficient to prove the theorem for $n = 1$. This fact is already present in the Cybenko-Hornik proofs and hinges on the fact that for any f,

$$A(f) = C(\mathbb{R}) \iff A_n(f) = C(\mathbb{R}^n).$$

In turn, this equivalence above follows from Weierstrass' polynomial approximation theorem if we can show that the span of the 'ridge polynomials' $f(w \cdot x + \theta)$ for one-dimensional polynomials f equals the set of all n-dimensional polynomials. This is the content of the next section.

We now continue with the proof that σ is non-polynomial iff $A(\sigma) = C(\mathbb{R})$.

2. If f is a polynomial, then $A(f)$ consists only of polynomials of the same degree. This follows from first observing that any element in $\operatorname{span}\{x \mapsto f(wx+\theta) : w, \theta \in \mathbb{R}\}$ is a polynomial of degree at most $\deg(f)$. Next, to show that the limit again is a polynomial of degree at most $\deg(f)$, note that if a sequence of polynomials of degree k converges at $k+1$ distinct points x_k, then this convergence implies convergence of the coefficients of the polynomials (see e.g. `https://math.stackexchange.com/questions/685472/find-n-degree-polynomial-from-n1-points`). Therefore the sup-norm convergence on any interval implies that the coefficients of the polynomial converge, and this implies convergence to a polynomial of no higher degree.

3. One establishes the following facts:

1. For any $f : \mathbb{R} \to \mathbb{R}$ and $\varphi \in C_c^\infty(\mathbb{R})$, $A(f) = A(f * \varphi)$;
2. If f is not a polynomial, then there exists $\varphi \in C_c^\infty(\mathbb{R})$ such that $f * \varphi$ also is not a polynomial;
3. If $f \in C^\infty(\mathbb{R})$ and f is not a polynomial, then $A(f) = C(\mathbb{R})$.

With these, the result follows::

$$\text{If } \sigma \text{ is not a polynomial, then } A(\sigma) \overset{1}{=} A(\sigma * \varphi) \underset{\exists\varphi}{\overset{2,3}{=}} C(\mathbb{R}).$$

4. Proof of the points under 3. The first point follows by approximating the convolution by summation over a finite number of points. Here the condition that $\sigma \in M$ becomes important, to make the convergence argument work.

The second point uses Baire's theorem to show that

$$\text{Each } f * \varphi \text{ is a polynomial of some degree} \implies \sup_{\varphi} \deg(f * \varphi) < \infty.$$

Assuming that $f * \varphi$ is a polynomial then implies that these polynomials have a joint degree m; by taking a sequence φ_n converging to a Dirac delta function in the sense of distributions shows that f then also is a polynomial of degree at most m. This establishes point 2.

The third point follows from a differentiation argument: by taking the limit of difference quotients, the function $(d/dw)f(wx+\theta) = xf'(wx+\theta)$ is an element of $A(f)$ for each w. For the choice $w = 0$ this function is $x \mapsto xf'(\theta)$, and since f is not a polynomial there exists $\theta \in \mathbb{R}$ such that $f'(\theta) \neq 0$. Therefore the function $x \mapsto x$ is an element of $A(f)$. By repeating this argument we find that all powers are in $A(f)$, and therefore $A(f)$ contains all polynomials; by Weierstrass we find that $A(f) = C(\mathbb{R})$.

2.3.6 A remark on representation of polynomials in several variables

Let $n \in \mathbb{N}_+$ and let P_n be the space of (real-valued) polynomials on $\mathbb{R}^n$. The aim of this section is to show that

$$P_n = \operatorname{span}\{[x \mapsto g(a \cdot x)] \,|\, a \in \mathbb{R}^n,\ g \in P_1\} =: M_n. \tag{2.3}$$

In particular, by the (n-dimensional version of) Weierstrass approximation theorem, this implies that M_n is dense in $C(\mathbb{R}^n)$ with respect to the topology of uniform convergence on compact subsets. The proof is a simplified part of the proof of [LP93, Theorem 2.1].

Obviously $M_n \subset P_n$. For $k \in \mathbb{N}$, let $H_{n,k}$ be the space of polynomials on $\mathbb{R}^n$ that are homogeneous of degree k. In multiindex notation, these are the polynomials of the form

$$p(x) = \sum_{|\alpha|=k} c_\alpha x^\alpha$$

with coefficients $c_\alpha \in \mathbb{R}$. Define further the subspace $S_{n,k} \subset H_{n,k}$ by

$$S_{n,k} := \operatorname{span}\{[x \mapsto (a \cdot x)^k] \,|\, a \in \mathbb{R}^n\}.$$

Since $P_n = \operatorname{span} \bigcup_{k\in\mathbb{N}} H_{n,k}$ and $S_{n,k} \subset M_n$ for all $k \in \mathbb{N}$, (2.3) follows if we prove

$$H_{n,k} = S_{n,k} \qquad \text{for all } k \in \mathbb{N}. \tag{2.4}$$

To show (2.4) we fix k and note that the partial derivatives ∂^β of order $|\beta| = k$ can be interpreted as linear maps from $H_{n,k}$ to the scalars, since they map each k-degree polynomial to a constant function. In particular, for $|\alpha| = |\beta| = k$,

$$\partial^\beta x^\alpha = \begin{cases} \alpha! & \text{if } \alpha = \beta, \\ 0 & \text{if } \alpha \neq \beta. \end{cases}$$

(Recall that for a multi-index $\alpha = (\alpha_1, \dots, \alpha_n)$ the generalized factorial $\alpha!$ is defined as $\prod_{i=1}^n \alpha_i!$). So $\{\partial^\beta \,|\, |\beta| = k\}$ is a basis for the dual space $H'_{n,k}$, and any $\lambda \in H'_{n,k}$ can be represented by some $q_\lambda \in H_{n,k}$ via

$$\langle \lambda, p\rangle = q_\lambda(\partial)p, \qquad \text{where} \quad q_\lambda(\partial) := \sum_{|\beta|=k} \frac{1}{\beta!} \langle \lambda, [x \mapsto x^\beta]\rangle\, \partial^\beta.$$

The map $[\lambda \mapsto q_\lambda]$ is an isomorphism from $H'_{n,k}$ to $H_{n,k}$.

We can write a polynomial of the form $p(x) = (a \cdot x)^k$, $a \in \mathbb{R}^n$, in the 'multinomial form'

$$p(x) = \sum_{|\alpha|=k} \frac{k!}{\alpha!} a^\alpha x^\alpha.$$

For any β with $|\beta| = k$ we therefore have

$$\partial^\beta \left[(a \cdot x)^k\right] = \sum_{|\alpha|=k} \frac{k!}{\alpha!} a^\alpha \partial^\beta x^\alpha = k!\, a^\beta$$

and more generally

$$q(\partial)\left[(a\cdot x)^k\right] = k!\,q(a), \qquad q\in H_{n,k},\ a\in\mathbb{R}^n. \tag{2.5}$$

Suppose now $\lambda \in S_{n,k}^{\perp}$. Then, by (2.5), $q_\lambda \equiv 0$, and hence $\lambda = 0$. This shows (2.4), and (2.3) is proved.

2.4 Intermezzo: The Kolmogorov-Arnold theorem

The results in the previous section give sufficient conditions under which one can approximate an 'arbitrary' function to any accuracy by finite shallow networks. Naturally, if the required error is smaller, then one will need larger networks. Surprisingly, if one is willing to *adapt the activation function to the target function* then one can achieve *exact equality* with only a finite number of terms. This is a consequence of the celebrated Kolmogorov-Arnold theorem.

While intriguing, this result is completely irrelevant for understanding and working with neural networks. Therefore we just present it here as an intermezzo, for the interested reader.

2.4.1 The Kolmogorov-Arnold Theorem

Theorem 2.4.1 (Kolmogorov-Arnold Theorem [Kol57])**.** *Fix $d \ge 2$. There exists a set of continuous functions $\psi_{pq} : [0,1] \to \mathbb{R}$, $1 \le p \le d$, $1 \le q \le 2d+1$, with the following property:*

For each $f \in C([0,1]^d)$ there exist $\chi_q \in C(\mathbb{R})$, $1 \le q \le 2d+1$, such that

$$f(x_1,\dots,x_d) = \sum_{q=1}^{2d+1} \chi_q\Bigl(\sum_{p=1}^{d} \psi_{pq}(x_p)\Bigr). \tag{2.6}$$

The resulting network is depicted in Figure 2.1. One aspect of this theorem is the fact that the functions χ_q are obtained as the limits of sequences that converge in the supremum norm. For this reason it is not easy to give useful characterizations of these functions χ_q.

The next table summarizes some differences between this result and the density statements of Cybenko and Hornik (Theorem 2.3.2 and Lemma 2.3.6)

Cybenko and Hornik	**Kolmogorov-Arnold**
$f(x) = \sum_{q=1}^{k} \alpha_q \sigma(w_q^T x + b_q)$	$f(x) = \sum_{q=1}^{2d+1} \chi_q\Bigl(\sum_{p=1}^{d} \psi_{pq}(x_p)\Bigr)$
dense subset	exact representation
σ any fixed bounded non-constant function	χ_q depends on f
smoothness of σ can be chosen	χ_q are highly non-smooth

It turns out that non-smoothness of χ_q and ψ_{pq} is essential, as shown in the following theorem:

Theorem 2.4.2 (Vitushkin Theorem [GP89, Th. 2.1])**.** *Fix $d \ge 2$ and $r \in \{1,2,\dots\}$. There exists an r times continuously differentiable function of d variables, that has no* exact *representation as linear combination and composition of r times continuously differentiable functions of less than d variables.*

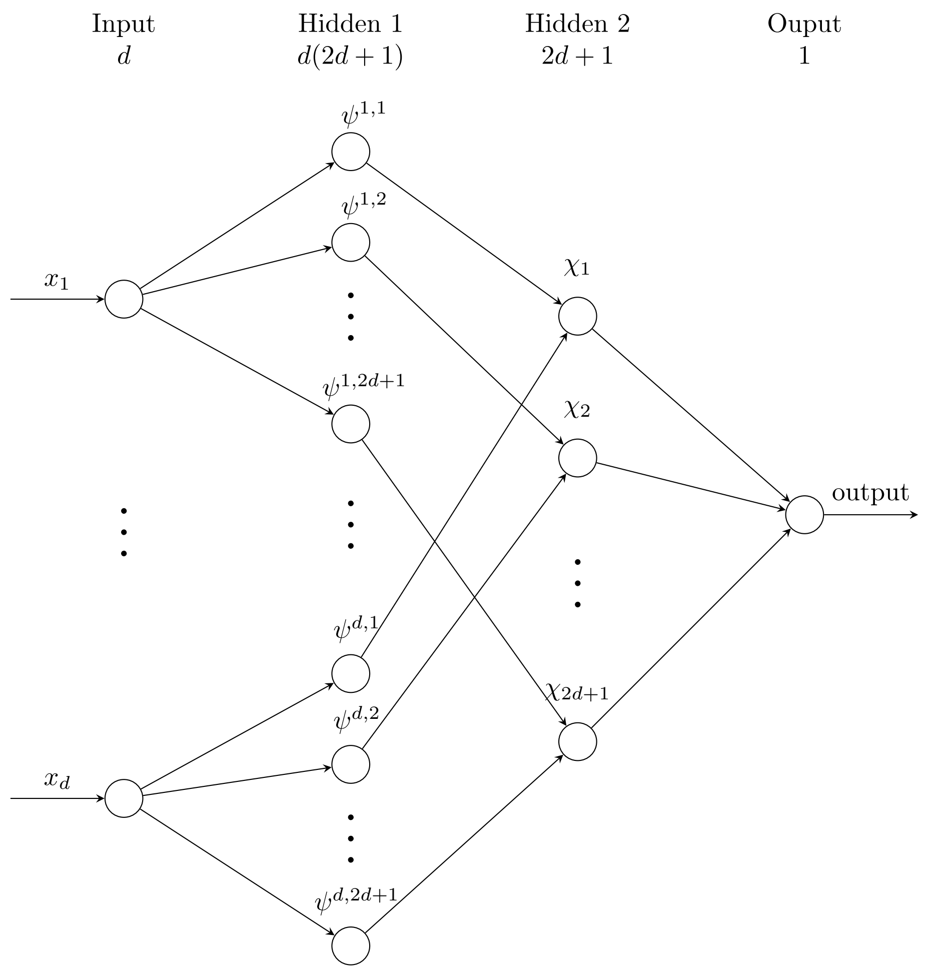


Figure 2.1: A graphical representation of the Kolmogorov-Arnold Theorem

Remark 2.4.3. One can interpret this theorem as follows: if one wants to represent functions on $\mathbb{R}^d$ by linear combinations of functions of one variable, then the one-dimensional functions *have to be less smooth* than the d-dimensional function you started with. Note that 'representation' here means a representation with *equality*, as in (2.6). This is different from representation up to some approximation error, as in Cybenko and Hornik (Section 2.3).

Note that this limitation only applies for 1, 2, or higher numbers of derivatives; it does not hold for *zero* derivatives, i.e. for continuity: indeed, the Kolmogorov-Arnold theorem represents continuous functions of d variables by linear combinations and compositions of continuous functions.

□

2.4.2 Quantifying the reduction of smoothness

Johannes Schmidt-Hieber proved the following quantification of the loss of smoothness described above. In the theorem below, we write $\mathbf{x} = (x_1, \ldots, x_d)$ for an element in $[0,1]^d$, and x for an element of the Cantor set $\mathcal{C}$, with similar convention for $\mathbf{y}$ and y.

Theorem 2.4.4 ([SH21])**.** *For fixed dimension $d \geq 2$, there exists a monotone function $\phi : [0,1] \to \mathcal{C}$ such that for any function $f : [0,1]^d \to \mathbb{R}$, we can find a function $g : \mathcal{C} \to \mathbb{R}$ such that*

1. *We have*
$$f(x_1, \dots, x_d) = g\Big(3 \sum_{p=1}^{d} 3^{-p} \phi(x_p)\Big);$$
2. *if f is continuous, then $g : \mathcal{C} \to \mathbb{R}$ also is continuous;*
3. *if f is β-Hölder continuous, i.e. if there exists $\beta \leq 1$ and a constant Q such that*
$$|f(\mathbf{x}) - f(\mathbf{y})| \leq Q|\mathbf{x} - \mathbf{y}|_\infty^\beta \qquad \textit{for all } \mathbf{x}, \mathbf{y} \in [0,1]^d,$$
then g is Hölder continuous with exponent $\beta \log 2/d \log 3$, i.e.
$$|g(x) - g(y)| \leq 2^\beta Q|x - y|^{\frac{\beta \log 2}{d \log 3}}, \qquad \textit{for all } x, y \in \mathcal{C}.$$

This result can be interpreted as a quantification of the loss-of-smoothness mentioned in Remark 2.4.3: if f is β-Hölder continuous, then one can not expect g to be Hölder continuous with the same exponent—but g *will* be Hölder continuous with the smaller exponent $\beta \log 2/d \log 3$. It is interesting to see that again the loss of smoothness vanishes in the limit case of 'simple' continuity, corresponding to $\beta \downarrow 0$; in an approximate sense, f and g become of similar 'smoothness' in that limit.

2.5 Width *vs.* depth

The previous sections dealt with the question 'can I approximate any function by a neural network?' The answer turns out to be Yes, provided the activation functions are not polynomials; this even is true with respect to several different norms, and this property only requires a single hidden layer. If one gives oneself the freedom to tune the activation to the function to be approximated, then one can even match a given function *exactly* (Section 2.4).

At the same time, it is clear from the preceding sections that approximation to a small error $\varepsilon > 0$ requires a large number of nodes. Therefore the question is natural how the worst-case error depends on the number of nodes, and in addition, whether this dependence can be improved by using more than one layer: by capitalizing on *depth*.

The main claim of this section, and a crucial reason for the use of neural networks in general, is this:

> By making use of the depth we can approximate functions *exponentially more efficiently* than without.

The remainder of this section explains this.

2.5.1 Basic principle I: Good approximation requires availability of high oscillation

Frenzen, Sasao, and Butler [FSB10] prove the following theorem, which is also used as one of the main results in [PGEB18].

Theorem 2.5.1. *Let $f \in C^3([a,b])$ and set*

$$c := \frac{1}{4}\int_a^b \sqrt{|f''(x)|}\,dx.$$

Let CPA_ε *be defined as*

$$\mathrm{CPA}_\varepsilon := \Big\{ g : [a,b] \to \mathbb{R} \textit{ continuous and piecewise affine satisfying } \|f-g\|_\infty \le \varepsilon \Big\}.$$

Let $s(\varepsilon)$ be defined as the smallest number of segments for an element of CPA_ε*. Then*

$$s(\varepsilon) \sim \frac{c}{\sqrt{\varepsilon}} \qquad \textit{as } \varepsilon \to 0. \tag{2.7}$$

It is intuitively obvious that the number of segments of a piecewise-affine approximation of a *nonlinear* function f has to increase when the required approximation accuracy improves. This theorem above gives a *quantification* of the number of segments in terms of the nonlinearity of the target function. It is intriguing that the relevant measure of nonlinearity of f is exactly the integral that defines c.

Idea of the proof. Assume that $g \in \mathrm{CPA}_\varepsilon$ is optimal; let $\Delta_i := [x_{i-1}, x_i]$ be the $s(\varepsilon)$ segments of g, with length δ_i, and since g is optimal it is not unreasonable to assume that

$$x \mapsto f(x) - g(x) \quad \text{achieves both } -\varepsilon \text{ and } \varepsilon \text{ on the interval } \Delta_i.$$

(Frenzen *et al.* prove the theorem without making this assumption.)

Focusing on Δ_i only, we assume for simplicity of notation that $g = 0$ on Δ_i. We also assume for simplicity that $f'' > 0$ on Δ_i, and set $d_i := \inf_{\Delta_i} f''$. Setting x_i^m to be the midpoint of Δ_i, we find by Taylor development that

$$\begin{aligned}
2\varepsilon = \max_{\Delta_i} f - \min_{\Delta_i} f &\ge \frac{1}{2}\Big(f(x_{i-1}) - 2f(x_i^m) + f(x_i)\Big)\\
&\ge \frac{1}{2}\Big(\frac{1}{2}(x_i^m - x_{i-1})^2 + \frac{1}{2}(x_i - x_i^m)^2\Big)d_i\\
&\ge \frac{1}{2}\Big(\frac{\delta_i}{2}\Big)^2 d_i = \frac{\delta_i^2 d_i}{8}.
\end{aligned}$$

Repeating this for the opposite sign of f'' we find

$$\varepsilon \ge \frac{\delta_i^2}{16}\inf_{\Delta_i}|f''|.$$

This inequality is the crucial ingredient: it limits the size δ_i of the segment Δ_i in terms of ε and $|f''|$. Rewriting this inequality to make it linear in δ_i we find

$$\frac{\delta_i}{4}\sqrt{\inf_{\Delta_i}|f''|} \le \sqrt{\varepsilon},$$

and by summing over i we obtain

$$\frac{1}{4}\sum_{i=1}^{s(\varepsilon)} \delta_i \sqrt{\inf_{\Delta_i}|f''|} \le s(\varepsilon)\sqrt{\varepsilon}.$$

On the left-hand side we recognize a Riemann sum approximating c, and we find that $\liminf_{\varepsilon\to 0}\sqrt{\varepsilon}\,s(\varepsilon) \ge c$. This is half of the characterization (2.7); the opposite inequality follows from an explicit construction. □

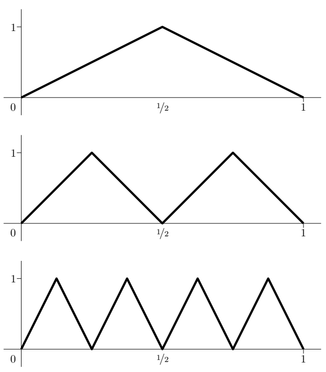


Figure 2.2: The m-fold composition $f_m = f \circ \cdots \circ f$ is a sawtooth function on $[0,1]$ with 2^{m-1} periods; shown are $f_1 = f$, f_2, and f_3. (Figure from [Tel15]).

2.5.2 Basic principle II: Composition creates exponentially many oscillations, addition only linearly many

Telgarsky [Tel15] remarks that a sawtooth function $[0,1]$ with 2^{m-1} periods can be constructed by taking the m-fold composition of a single sawtooth function f,

$$f(x) := \begin{cases} 2x & \text{if } 0 \le x \le 1/2 \\ 2 - 2x & \text{if } 1/2 \le x \le 1. \end{cases}$$

By contrast, a linear superposition of m scaled and translated copies of functions consisting of at most k affine segments has no more than mk segments. These basic examples illustrate why it is possible that the 'freedom to oscillate' increases exponentially with depth and only linear with width.

2.5.3 Concrete theorems

Telgarsky considers binary classification, and proves that deeper networks achieve functions that can not be approximated by shallower ones:

Theorem 2.5.2 ([Tel15, Th. 1.1]). *Let σ be the ReLU function. For $f : \mathbb{R} \to \mathbb{R}$ and a given set of points $D = \{(x_i, y_i)\}_{i=1}^n \subset [0,1] \times \{-1, 1\}$, define*

$$\mathcal{R}(f, D) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{\operatorname{sign}(f(x_i)) \neq y_i\}.$$

Choose $k \in \mathbb{N}$. Then there exists a choice of D with $n = 2^k$ points such that

$$\min_f \Big\{ \mathcal{R}(f, D) : f \text{ has } 2k \text{ layers and } 2 \text{ nodes per layer} \Big\} = 0, \tag{2.8}$$

while for any $L \in \mathbb{N}$,

$$\min_f \Big\{ \mathcal{R}(f, D) : f \text{ has } L \text{ layers and at most } 2^{\frac{k-3}{L} - 1} \text{ nodes per layer} \Big\} \ge \frac{1}{6}. \tag{2.9}$$

These assertions can be interpreted as follows:

- Property (2.8) states that for this choice of data it is sufficient to fix the number of nodes per layer and grow the depth logarithmically in the number of data points ($2k = 2\log_2 2^k$), to achieve perfect matching;
- Property (2.9) implies that for the same data points with a fixed number of layers L even exponential growth $2^{k/L}$ in k is not sufficient to reduce the error.

Morally, therefore, this result shows that deeper networks can realize functions that shallower networks can not represent.

Yarotsky [Yar17] considers approximation in the L^∞-norm of functions in the unit ball $F_{d,k}$ of the Sobolev space $W^{k,\infty}([0,1]^d)$ on the d-dimensional unit cube $[0,1]^d$. (This is the unit ball with respect to the $W^{k,\infty}$ Sobolev norm, even though the quality of approximation is measured in the L^∞-norm).

Theorem 2.5.3 ([Yar17, Th. 1]). *For $d \geq 1$, $k \geq 1$, and $\varepsilon > 0$, there exists a ReLU network architecture with depth $O(\log(1/\varepsilon))$ and $O(\varepsilon^{-d/k}\log(1/\varepsilon))$ parameters, that approximates each $f \in F_{d,k}$ to L^∞-error ε.*

This result is very much in line with the general principle described in Section 2.2. On the other hand, for fixed depth any nonlinear function can only be approximated with a number of nodes that is linear in the error:

Theorem 2.5.4 ([Yar17, Th. 6]). *Let $f \in C^2([0,1]^d)$ be nonlinear and fix the depth L. Any ReLU network approximating f to L^∞-error $\varepsilon > 0$ has at least $\Omega(\varepsilon^{-1}/2(L-2))$ nodes.*

This last statement means that the number of nodes is bounded from below by $C\varepsilon^{-1}/2(L-2)$ where C is a constant depending on f.

2.5.4 Approximation of analytic functions

We mentioned in Section 2.1 that analytic functions can be approximated very efficiently by truncating their Taylor development. It turns out analytic functions also can be approximated efficiently by deep neural networks:

Theorem 2.5.5 ([EW18, Cor. 11]). *Let $f : (-1,1)^d \to \mathbb{R}$ have a power series $f(x) = \sum_{k\in\mathbb{N}_0^d} a_k x^k$ that is absolutely convergent in $[-1,1]^d$. Then for any $\varepsilon, \delta > 0$, there exists a function $\hat{f}$ that can be represented by a deep ReLU network with depth $L = \left[e\left(\frac{1}{d\delta}\log\frac{1}{\varepsilon}+1\right)\right]^{2d}$ and width $d+4$, such that*

$$|f(x) - \hat{f}(x)| < 2\varepsilon \sum_{k\in\mathbb{N}_0^d} |a_k| \qquad \text{for all } x \in [-1+\delta, 1-\delta]^d.$$

Since the number of parameters in a fixed-width network is linear in the depth, this result shows that in order to improve the error ε by a *factor* 2, the number of parameters only needs to be increased by *addition* of a number of layers equal to

$$\left[e\left(\frac{1}{d\delta}\log\frac{2}{\varepsilon}+1\right)\right]^{2d} - \left[e\left(\frac{1}{d\delta}\log\frac{1}{\varepsilon}+1\right)\right]^{2d} \approx \frac{2de^{2d}}{(d\delta)^{2d}}\log 2.$$

Montanelli, Yang, and Du [MYD19] prove a similar result for a related class of *bandlimited* functions.

2.6 Intermezzo: Lower bounds on complexity for approximations with continuous parameters

De Vore, Howard, and Miccheli describe in a very readable article [DHM89] some ideas around optimal approximation. Fix a normed space X and an n-dimensional manifold $\mathcal{M}_n$ generated by a function M_n:

$$\mathcal{M}_n := \{M_n(a) : a \in \mathbb{R}^n\}.$$

De Vore *et al.* define the 'nonlinear n-width'

$$d_n(K)_X := \inf_{A,M_n} \sup_{f\in K} \|f - M_n(A(f))\|_X,$$

where the infimum runs over all continuous functions $A : K \to \mathbb{R}^n$ and $M_n : \mathbb{R}^n \to X$. This expression characterizes the best approximation of the set K by n-dimensional manifolds $\mathcal{M}_n$, with the proviso that there must also exist a continuous reverse mapping A that achieves this 'best' approximation.

One of the results by De Vore *et al.* is

Theorem 2.6.1 ([DHM89, Th. 4.2]). *Let $\Omega = [0,1]^d$ and $X = L^q(\Omega)$, with $1 \le q \le \infty$. Let K be the unit ball in $W^{k,p}(\Omega)$ for $k \ge 1$ and $1 \le p \le q$. Then*

$$d_n(K)_X \ge Cn^{-k/d},$$

with a constant C that only depends on k.

This result shows that the best approximation *under the condition that the parameter points $A(f)$ depend continuously on f* requires $O(\varepsilon^{-d/k})$ parameter dimensions to achieve error ε. For approximation with continuous parameters the curse of dimension therefore is completely unavoidable.

2.7 Intermezzo: Approximation with *random* neural networks

2.7.1 Random choice of parameters

Cybenko, Hornik, and others proved universal-approximation results for one-hidden-layer networks of the form (2.1), i.e.

$$f : \mathbb{R}^d \to \mathbb{R}, \qquad f(x) = \sum_{j=1}^{n} \alpha_j \sigma(w_j^T x + b_j). \tag{2.10}$$

These results identify conditions on σ and the function-to-be-approximated g such that g can be approximated by arbitrary precision by choosing α_j, w_j, and b_j appropriately. In this situation, one may consider α_j, w_j, and b_j to be implicitly defined functions of the target function g.

In this section we discuss results by Igelnik and Pao [IP95] and Rahimi and Recht [RR08]. These authors also consider one-hidden-layer networks of the form (2.10), but instead choose w_j and b_j *randomly and independently of g.* Given the choices of w_j and b_j, the parameters α_j are then chosen such as to make f in (2.10) close to g.

Both sets of authors emphasize that this way of choosing parameters leads to much more efficient training, since for fixed w_j and b_j, writing $\phi_j(x)$ for $\sigma(w_j^T x - b_j)$, the norm

$$\|f - g\|^2_{L^2_\mu} = \int \Big| \sum_{j=1}^n \alpha_j \phi_j(x) - g(x) \Big|^2 \mu(dx)$$
$$= \sum_{j,k=1}^n \alpha_k \alpha_j \int \phi_j(x)\phi_k(x)\,\mu(dx) - 2\sum_{j=1}^n \alpha_j \int g(x)\phi_j(x)\,\mu(dx) + \int g(x)^2 \mu(dx) \quad (2.11)$$

is a quadratic function of the parameters α_j. Optimization of α_j for given ϕ_j therefore is straightforward and explicit, in contrast to optimization of α_j, w_j, and b_j simultaneously, which is a highly nonlinear problem.

Igelnik and Pao's main result is

Theorem 2.7.1 ([IP95, Th. 3]). *Let $\sigma : \mathbb{R} \to \mathbb{R}$ satisfy*

$$\text{either} \int_{\mathbb{R}} \sigma(x)^2\,dx < \infty \quad \text{or} \quad \int_{\mathbb{R}} \sigma'(x)^2\,dx < \infty.$$

Let $g : [0,1]^d \to \mathbb{R}$ be Lipschitz continuous with Lipschitz constant κ (with respect to the ℓ^1-norm on $[0,1]^d$). Then there exists a constant C with the following property: for each $n \in \mathbb{N}$ there exists a probability measure μ_n on the set $(\mathbb{R}^d)^n \times \mathbb{R}^n$, such that drawing (w, b) from μ_n and after optimizing (2.11) *over α_j, we have*

$$\mathbb{E}\|g - f_{w,b,\alpha}\|^2_{L^2([0,1]^d)} \le \frac{C}{n}.$$

Remark 2.7.2. Note that as Theorem 2.7.1 stands, nothing prevents us from choosing μ_n to be a Dirac delta at a single parameter point, for instance one given by Hornik's result. This would make the discussion about randomness meaningless, because the network then is deterministic. In reality the construction by Igelnik and Pao yields a measure μ_n that has a certain degree of independence, but writing this out here would take too much space.

Also note that this formulation of Theorem 3 in [IP95] is fully my (Mark Peletier's) own; the version in the paper makes even less sense than this one, and the proof is riddled with errors, and therefore I (Mark) find it difficult to see exactly what this statement *should* be. □

Rahimi and Recht have a similar approximation theorem, but with a number of differences. They consider functions on some compact set $X \subset \mathbb{R}^d$ that are parametrised by a parameter space Θ.

1. The class of target functions is

$$\mathcal{F} := \left\{ f(x) = \int_\Theta \alpha(\theta)\phi(x;\theta)\,d\theta \,\Big|\, \|f\|_\nu < \infty \right\},$$

where $\nu \in \mathcal{P}_{\mathrm{ac}}(\Theta)$ is some probability measure that is absolutely continuous with respect to Lebesgue, and the norm is defined as

$$\|f\|_\nu := \sup_{\theta\in\Theta} \left| \frac{\alpha(\theta)}{\nu(\theta)} \right|.$$

2. The approximation is in L^∞ with high probability instead of in L^2.

Theorem 2.7.3 ([RR08, Th. 3.2])**.** *Let $\phi(x;\theta) = \sigma(\theta^T x)$, with $\sigma : \mathbb{R} \to \mathbb{R}$ L-Lipschitz, $\sigma(0) = 0$, and $\|\sigma\|_\infty \le 1$. Also assume that $m_2 := \int_\Theta |\theta|^2 \, \nu(d\theta) < \infty$. Fix $f \in \mathcal{F}$. Then for any $\delta > 0$, the following holds true with probability at least $1 - \delta$:*

After drawing $\theta_1, \dots, \theta_n$ i.i.d. from ν, there exist $c_1, \dots, c_n \in \mathbb{R}$ such that

$$\hat{f}(x) := \sum_{i=1}^n c_i \sigma(\theta_i^T x)$$

satisfies

$$\|f - \hat{f}\|_{L^\infty(X)} \le \frac{\|f\|_\nu}{\sqrt{n}} \left(\sqrt{\log \frac{1}{\delta}} + 4LB\sqrt{m_2} \right),$$

where $B := \sup_{x \in X} |x|_2$.

Remark 2.7.4. In fact, the proof makes a very explicit choice for the c_i:

$$c_i := \frac{1}{n} \frac{\alpha(\theta_i)}{\nu(\theta_i)}.$$

This is a very reasonable choice, since then

$$\alpha^n := \sum_{i=1}^n c_i \delta_{\theta_i} = \frac{1}{n} \sum_{i=1}^n \frac{\alpha(\theta_i)}{\nu(\theta_i)} \delta_{\theta_i}$$

is an approximation of α under ν: for $A \subset \Theta$,

$$\mathbb{E}_\nu \alpha^n(A) = \frac{1}{n} \sum_{i=1}^n \mathbb{E}_\nu \frac{\alpha(\theta_i)}{\nu(\theta_i)} \delta_{\theta_i}(A) = \mathbb{E}_\nu \frac{\alpha(\theta_1)}{\nu(\theta_1)} \delta_{\theta_1}(A) = \int_{\theta_1 \in \Theta} \frac{\alpha(\theta_1)}{\nu(\theta_1)} \mathbb{1}\{\theta_1 \in A\} \nu(d\theta_1) = \int_{\theta_1 \in A} \alpha(\theta_1).$$

Effectively, α^n becomes a Monte-Carlo sampler for α under ν. □

Remark 2.7.5. The class $\mathcal{F}$ is defined in a very implicit way; the finiteness of $\|f\|_\nu$ is a very strong requirement. Rahimi and Recht show that $\mathcal{F}$ is a dense subset of a reproducing-kernel Hilbert space $\mathcal{H}$ defined by the kernel functions

$$k(x, y) := \int_\Theta \nu(\theta) \phi(x;\theta) \phi(y;\theta) \, d\theta.$$

They also say that in many cases of interest, $\mathcal{H}$ is dense in the set of continuous functions. □

Chapter 3

Training neural networks

3.1 Minimizing convex functions

3.1.1 Convex functions

Definition 3.1.1. A function $f : \mathbb{R}^d \to \mathbb{R}$ is called *convex* if it satisfies

$$\forall x, y \in \mathbb{R}^d,\ \forall 0 \le \alpha \le 1, \qquad f\bigl(\alpha x + (1-\alpha)y\bigr) \le \alpha f(x) + (1-\alpha) f(y).$$

A function f is called λ-*convex*, for $\lambda \in \mathbb{R}$, if

$$\forall x, y \in \mathbb{R}^d,\ \forall 0 \le \alpha \le 1, \qquad f\bigl(\alpha x + (1-\alpha)y\bigr) \le \alpha f(x) + (1-\alpha) f(y) - \frac{\lambda}{2}\alpha(1-\alpha)\|x-y\|^2.$$

Consequently ‘convex’ is the same as ‘0-convex’. 0-convexity means that the graph of f lies below the straight lines connecting any two points on the graph; λ-convexity for $\lambda > 0$ means that the graph even lies beneath a parabola connecting the two points with second derivative λ. Similarly λ-convexity for $\lambda < 0$ means that the graph lies below a parabola with negative second derivative λ. If $\lambda_1 \le \lambda_2$, then λ_2-convexity implies λ_1-convexity; the concept of λ-convexity is stronger when λ is larger.

Lemma 3.1.2 (Alternative characterizations)**.** *The property that $f : \mathbb{R}^d \to \mathbb{R}$ is convex [λ-convex] is equivalent to any of the following properties:*

1. *(assuming $f \in C^2(\mathbb{R}^d)$)*

$$\forall x \in \mathbb{R}^d: \qquad d^2 f(x) \ge 0 \quad [d^2 f(x) \ge \lambda],$$

   where $d^2 f(x)$ is the second derivative (Hessian matrix) at x. The matrix inequality ‘$A \ge \lambda$’ means that the matrix $A - \lambda I$ is positive semidefinite.

2. *(assuming $f \in C^1(\mathbb{R}^d)$)*

$$\forall x, y \in \mathbb{R}^d: \qquad (\nabla f(x) - \nabla f(y)) \cdot (x-y) \ge 0 \quad [\ \ge \lambda\|x-y\|^2],$$

   where $\cdot$ is the inner product in $\mathbb{R}^d$.

3. For each $x \in \mathbb{R}^d$ there exists $p \in \mathbb{R}^d$ (depending on x) such that

$$\forall y \in \mathbb{R}^d : \qquad f(y) \geq f(x) + p \cdot (y - x) \quad \Big[\geq f(x) + p \cdot (y-x) + \frac{\lambda}{2}\|y - x\|^2 \Big]. \tag{3.1}$$

Definition 3.1.3. Let f be convex. For each $x \in \mathbb{R}^d$ the *subdifferential* $\partial f(x)$ is defined as the subset of $\mathbb{R}^d$ given by

$$\partial f(x) := \Big\{ \, p \in \mathbb{R}^d : \ p \text{ satisfies (3.1)} \, \Big\} \subset \mathbb{R}^d.$$

Lemma 3.1.4. *Let f be convex. We have the equivalence*

$$\partial f(x) \text{ is a singleton} \qquad \Longleftrightarrow \qquad f \text{ is differentiable at } x.$$

In this case $\partial f(x) = \{\nabla f(x)\}$.

Example 3.1.5. The function $f : \mathbb{R}^d \to \mathbb{R}$ given by the d-dimensional Euclidean norm $f(x) = \|x\|$ is a convex function, since by the triangle inequality

$$f(\alpha x + (1-\alpha)y) = \|\alpha x + (1-\alpha)y\| \leq \|\alpha x\| + \|(1-\alpha)y\| = \alpha\|x\| + (1-\alpha)\|y\|.$$

The norm is differentiable at any $x \neq 0$ with derivative

$$\nabla f(x) = \Big(\frac{x_1}{\|x\|}, \ldots, \frac{x_d}{\|x\|}\Big) = \frac{x}{\|x\|}.$$

For each $x \neq 0$ therefore

$$\partial f(x) = \Big\{ \frac{x}{\|x\|} \Big\}.$$

For $x = 0$ we have

$$\partial f(0) = \overline{B(0,1)} = \{z \in \mathbb{R}^d : \|z\| \leq 1\}.$$

To verify this: if $\|z\| \leq 1$, then we check (3.1):

$$\|y\| \overset{?}{\geq} 0 + z \cdot (y - 0),$$

which is true by $|z \cdot y| \leq \|z\| \, \|y\| \leq \|y\|$. Vice versa, if $z \in \mathbb{R}^d$ satisfies

$$\|y\| \geq z \cdot y \qquad \text{for all } y \in \mathbb{R}^d,$$

then by taking $y = z$ we obtain $\|z\| \geq \|z\|^2$ which implies $\|z\| \leq 1$. □

3.1.2 The Polya-Łojasiewicz inequality

A commonly studied alternative property is the Polya-Łojasiewicz inequality.

Definition 3.1.6. The function $f : \mathbb{R}^d \to \mathbb{R}$ satisfies a *Polya-Łojasiewicz inequality* with parameter $\Lambda > 0$ if

$$\Lambda(f(x) - \inf f) \leq |\nabla f(x)|^2 \qquad \text{for all } x \in \mathbb{R}^d. \tag{3.2}$$

The Polya-Łojasiewicz inequality is implied by uniform convexity:

Lemma 3.1.7. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be λ-convex for some $\lambda > 0$. Then f satisfies the Polya-Łojasiewicz inequality with parameter $\Lambda = \lambda$.*

Proof. Assume without loss of generality that $f(0) = 0$, and assume for simplicity that f is C^2 (the proof can be modified to deal with the more general case). From convexity we obtain immediately that for all x,

$$f(x) \le x \cdot \nabla f(x).$$

Since f is λ-convex, the Hessian satisfies $d^2 f(x) \ge \lambda I$, and therefore

$$\begin{aligned} |x||\nabla f(x)| \ge x \cdot f(x) &= \int_0^1 x \cdot \partial_t\big(\nabla f(tx)\big)\,dt = \int_0^1 x \cdot d^2 f(tx) x\,dt \\ &\ge \lambda \int_0^1 x \cdot x\,dt = \lambda |x|^2, \end{aligned}$$

and therefore $\lambda|x| \le |\nabla f(x)|$. Combining these inequalities we find

$$f(x) \le x \cdot \nabla f(x) \le |x||\nabla f(x)| \le \frac{1}{\lambda}|\nabla f(x)|^2.$$

This proves the result. □

3.2 Descent methods

Given a function $f : \mathbb{R}^d \to \mathbb{R}$, the aim of a *descent method* is to find minimizers of f. This class of algorithms iteratively constructs the next candidate u_{k+1} from the previous candidate u_k by

$$u_{k+1} = u_k + \eta_k \Delta u_k, \tag{3.3}$$

where $\Delta u_k = (\Delta u)_k$ is an 'update direction' and $\eta_k > 0$ can be considered a step size. A natural choice for Δu_k is the negative gradient of f,

$$\Delta u_k = -\nabla f(u_k), \tag{3.4}$$

but other choices also are possible. The name 'descent method' comes from the fact that Δu_k and η_k are chosen such that $f(u_{k+1}) \le f(u_k)$.

The role of $\eta_k > 0$ is to tune the size of the step that we take in the direction Δu_k. The choice of η_k can be done in different ways:

- η_k can be fixed in advance;
- η_k can be chosen as a given function of u_k or of $\{u_1, \dots, u_k\}$ (these are called *adaptive methods);*
- η_k can be chosen to minimize $\eta \mapsto f(u_k + \eta \Delta u_k)$ for given Δu_k *(exact line search);*
- η_k can be chosen an approximation of this minimizer *(approximate line search).*

3.2.1 Wolfe conditions and the Zoutendijk Theorem

Stepsizes can be too large, which leads to 'zig-zagging', but they can also be too small, leading to inefficient algorithms. Assuming a fixed descent direction ρ_k, we now look for a suitable step size along this direction. As a consequence, we define $L(\eta) = f(u_k + \eta\rho_k)$. Given $0 < a < b < 1$, the Wolfe Conditions are:

1. A condition of 'sufficient decrease': $L(\eta_k) \le L(0) + a\eta_k L'(0)$;

2. A curvature condition: $L'(\eta_k) \geq bL'(0)$.

We now turn to the feasibility of such conditions: is it always possible to find an η such that they are satisfied? As far as the sufficient decrease condition is concerned, the most widely used algorithm to find a suitable η is *backtracking line search.* This algorithm always finds a suitable step size.

Algorithm 1: Backtracking Line Search

```
Result: $\eta$ satisfying the sufficient decrease condition
Let $\bar{\eta} > 0, \tau \in (0,1), a \in (0,1)$ $\eta = \bar{\eta}$;
while $L(\eta) > L(0) + a\eta L'(0)$ do
    $\eta = \tau\eta$;
end
```

The Wolfe conditions play a key role in Zoutendijk's Theorem, which guarantees convergence of the gradient descent algorithm.

Theorem 3.2.1. *Let $f \in C^1$ and let ρ_k be the direction at iteration k. Let θ_k be the angle between ρ_k and $-\nabla f(u_k)$. Furthermore, assume that at every iteration the Wolfe conditions hold. Then,*

$$\sum_{k=1}^{\infty} \cos^2(\theta_k)\|\nabla f(u_k)\| < \infty. \tag{3.5}$$

This means that, as long as $\theta_k < \frac{\pi}{2}$ at every iteration, we get that the gradient converges to 0.

3.2.2 Rates of convergence

Let $x_k \to x$. We say that this convergence is

linear if	$\dfrac{\|x_k - x\|}{\|x_{k-1} - x\|} \longrightarrow r \in (0,1)$	as $k \to \infty$,
superlinear if	$\dfrac{\|x_k - x\|}{\|x_{k-1} - x\|} \longrightarrow 0$	as $k \to \infty$
quadratic if	$\dfrac{\|x_k - x\|}{\|x_{k-1} - x\|^2} \longrightarrow M \in (0,\infty)$	as $k \to \infty$.

Examples are

linear:	$x_k = 1 + \left(\frac{1}{2}\right)^k$	(and then $r = 1/2$)
superlinear:	$x_k = 1 + \left(\frac{1}{k}\right)^k$	
quadratic:	$x_k = 1 + \left(\frac{1}{2}\right)^{2^k}$	(and then $M = 1$).

3.2.3 Gradient descent methods and gradient flows

Given a function $f : \mathbb{R}^d \to \mathbb{R}$, the *gradient flow* of f is the ordinary differential equation

$$\dot{u} = -\nabla f(u), \tag{3.6}$$

whose solutions are curves $t \mapsto u(t) \in \mathbb{R}^d$. These curves move 'downhill' in the landscape defined by f, since for any solution u,

$$\frac{d}{dt} f(u(t)) = \nabla f(u(t)) \cdot \dot{u} = -\|\nabla f(u(t))\|^2 \leq 0.$$

In fact, this calculation implies that they continue to move downhill until they reach a stationary point of f, i.e. a point where $\nabla f = 0$.

The *forward-Euler discretization* of the ODE (3.6) is defined as follows: given a sequence of step sizes $\eta_k > 0$ one defines the times t_k by $t_0 = 0$ and $t_{k+1} = t_k + \eta_k$. Then a single discretized step is

$$u(t_{k+1}) = u(t_k) - \eta_k \nabla f(u(t_k)). \tag{3.7}$$

This forward-Euler discretization does not necessarily reduce f, as a simple example shows: taking $f(u) = u^2$, $u(0) = 1$, and $\eta_k = \eta = 1$, we find that $u(t_k) = (-1)^k$ and $f(u(t_k)) = 1$ for all k. If we make η larger than 1, then we even have $f(u(t_k)) \to \infty$ as $k \to \infty$.

However, in the limit $\eta \to 0$ the solutions of the gradient-descent algorithm (3.7) do converge to the solution of (3.6), under mild conditions on the regularity of f (e.g. [HNW93, Th. 3.4]).

The forward-Euler discretization (3.7) coincides with the general descent method (3.3) with the choice (3.4), and is often called (an example of) a *gradient descent method*.

Remark 3.2.2. Note that an algorithmic point of view, as in (3.3), suggests to number the iterates u_k by $k \in \mathbb{N}$, while the connection with ODEs in (3.7) suggests to think of u_k as an approximation of the value of u at $t = t_k$, where $t_k = \sum_{k'=1}^{k} \eta_{k'}$.

The difference between the two views becomes especially important when considering variation in step size η, for instance when choosing η_k not constant. In this context the connection with the ODE provides the intuition how to map different choices of η to a single 'time' axis. This is important when considering proofs of convergence as $\eta \to 0$. □

3.2.4 Example convergence theorem

The convergence theorem below makes very strong use of the uniform convexity of the function f, expressed through the upper and lower bounds

$$m\|\xi\|^2 \leq \xi^T d^2 f(x) \xi \leq M\|\xi\|^2 \qquad \text{for all } x, \xi \in \mathbb{R}^d, \tag{3.8}$$

where $m, M > 0$ are constants independent of x. For such uniformly convex functions the mismatch between an arbitrary point x and the global minimizer x^* can be expressed in three different ways:

$f(x) - f(x^*)$	the degree to which x does not minimize f,
$\|x - x^*\|$	the distance in the underlying normed vector space,
$\|\nabla f(x)\|$	the degree to which x is not a stationary point of f.

The following lemma shows how strong convexity bounds make these three quantities 'equivalent' to each other, somewhat like the concept of equivalent norms. These inequalities form the core of the convergence theorem below.

Lemma 3.2.3. *Let f satisfy* (3.8) *and let $x^* \in \mathbb{R}^d$ be the global minimizer of f. Then*

$$\frac{1}{2M}\|\nabla f(x)\|^2 \le f(x) - f(x^*) \le \frac{1}{2m}\|\nabla f(x)\|^2 \qquad \text{for all } x \in \mathbb{R}^d \tag{3.9}$$

$$\frac{m}{2}\|x-x^*\|^2 \le f(x) - f(x^*) \le \frac{M}{2}\|x-x^*\|^2 \qquad \text{for all } x \in \mathbb{R}^d \tag{3.10}$$

We also have the following convexity bounds:

$$f(x) \ge f(x_0) + \nabla f(x_0)\cdot(x-x_0) + \frac{m}{2}\|x-x_0\|^2 \qquad \text{for all } x, x_0 \in \mathbb{R}^d. \tag{3.11}$$

$$f(x) \le f(x_0) + \nabla f(x_0)\cdot(x-x_0) + \frac{M}{2}\|x-x_0\|^2 \qquad \text{for all } x, x_0 \in \mathbb{R}^d. \tag{3.12}$$

Proof. The left-hand inequality in (3.8) implies that f is m-convex, and using Lemma 3.1.2 we then find (3.11). Choosing x_0 in (3.11) to be the global minimizer x^* of f, we find

$$\frac{m}{2}\|x-x^*\|^2 \le f(x) - f(x^*) \qquad \text{for all } x \in \mathbb{R}^d.$$

This proves the first inequality in (3.10). To prove the second inequality in (3.9), we observe that

$$\begin{aligned} f(x) - f(x_0) &\ge \nabla f(x_0)\cdot(x-x_0) + \frac{m}{2}\|x-x_0\|^2 \\ &\ge \inf_{y\in\mathbb{R}^d} \nabla f(x_0)\cdot(y-x_0) + \frac{m}{2}\|y-x_0\|^2 \\ &= -\frac{1}{2m}\|\nabla f(x_0)\|^2, \end{aligned}$$

where the final identity follows from observing that the infimum is achieved at $y = x_0 - \nabla f(x_0)$. Choosing $x_0 = x$ and $x = x^*$ yields the second inequality in (3.9).

Defining $g(x) := (M/2)\|x\|^2 - f(x)$ we find that $d^2g(x) = MI - d^2f(x) \ge 0$, and applying Lemma 3.1.2 to g we find the upper bound (3.12). The remaining inequalities follow analogously. □

Theorem 3.2.4 ([BV04, Sec. 9.3])**.** *Let $f \in C^2(\mathbb{R}^d)$ satisfy* (3.8). *Then Gradient Descent with exact line search converges in the 'linear' sense above; more precisely,*

$$\frac{f(u_{k+1}) - f(x^*)}{f(u_k) - f(x^*)} \le r := 1 - \frac{m}{M};$$

$$\|u_k - x^*\| \le Cr^{k/2}, \qquad \textit{with } C^2 = \frac{2}{m}\big(f(u_0) - f(x^*)\big).$$

Proof. We apply the estimates of Lemma 3.2.3 to the sequence u_k generated by gradient descent with exact line search. Gradient descent implies that $\Delta u_k = -\nabla f(u_k)$. For any choice $\eta > 0$ (not necessarily optimal in the line search) we have

$$\begin{aligned} f\big(u_k - \eta\nabla f(u_k)\big) &\overset{(3.12)}{\le} f(u_k) + \nabla f(u_k)\cdot\big(-\eta\nabla f(u_k))\big) + \frac{M}{2}\|\eta\nabla f(u_k)\|^2 \\ &= f(u_k) - \eta\|\nabla f(u_k)\|^2 + \frac{M\eta^2}{2}\|\nabla f(u_k)\|^2. \end{aligned}$$

The right-hand side of this inequality is a quadratic function of η, and it is minimized at $\eta = M^{-1}$, with a value we can calculate explicitly. We find that

$$f\big(u_k - M^{-1}\nabla f(u_k)\big) \le f(u_k) - \frac{1}{2M}\|\nabla f(u_k)\|^2.$$

Since η_k is chosen to minimize $f\big(u_k - \eta\nabla f(u_k)\big)$, it follows that

$$f(u_{k+1}) = f\big(u_k - \eta_k\nabla f(u_k)\big) \le f\big(u_k - M^{-1}\nabla f(u_k)\big) \le f(u_k) - \frac{1}{2M}\|\nabla f(u_k)\|^2. \tag{3.13}$$

We then calculate

$$\begin{aligned} f(u_{k+1}) - f(x^*) &\overset{(3.13)}{\le} f(u_k) - f(x^*) - \frac{1}{2M}\|\nabla f(u_k)\|^2 \overset{(3.9)}{\le} f(u_k) - f(x^*) - \frac{m}{M}\big(f(u_k) - f(x^*)\big) \\ &= \underbrace{\Big(1 - \frac{m}{M}\Big)}_{r}\big(f(u_k) - f(x^*)\big). \end{aligned}$$

This implies that the sequence $f(u_k)$ converges to $f(x^*)$ at linear rate, with $r = 1 - m/M < 1$. By iterating the inequality we find

$$f(u_k) - f(x^*) \le \big(f(u_0) - f(x^*)\big)\, r^k.$$

We also want convergence of the points u_k themselves. We find this by applying inequality (3.10):

$$\|u_k - x^*\|^2 \le \frac{2}{m}\big(f(u_k) - f(x^*)\big) \le \frac{2}{m}\big(f(u_0) - f(x^*)\big)\, r^k =: C^2 r^k.$$

This concludes the proof. □

3.3 Stochastic Gradient Descent

Stochastic Gradient Descent is a modification of (3.7) in which the update $-\eta\nabla f(u)$ is replaced by a *random* approximation of this update:

$$u(t_{k+1}) = u(t_k) + \eta_k g(u(t_k), Z_k). \tag{3.14}$$

Here g is a (deterministic) function of two arguments, and Z_k is a sequence of i.i.d. random variables. It is standard to assume that g is an unbiased approximation of $-\nabla f$:

$$\forall u \in \mathbb{R}^d : \qquad \mathbb{E}g(u, Z) = -\nabla f(u).$$

The most common version of this in the context of the training of neural networks arises from *mini-batching*. To conform with common practice we switch notation: we write θ for the parameter for which we are seeking a value that minimizes the loss $\mathcal{R}_N(\theta)$ (so u is replaced by θ and f by $\mathcal{R}_N$).

The function $\mathcal{R}_N$ is defined by a data set $\{(x_i, y_i)\}_{1\le i\le N}$ as

$$\mathcal{R}_N(\theta) = \frac{1}{N}\sum_{i=1}^N \big|y_i - \psi(x_i;\theta)\big|^2.$$

A straightforward gradient descent on $\mathcal{R}_N$ therefore has the form

$$\theta(t_{k+1}) = \theta(t_k) - \eta\nabla\mathcal{R}_N(\theta(t_k)).$$

The stochastic version of this is obtained by defining each Z_k to be an independent subset of size $M \ll N$ of the set $\{1, 2, \dots, N\}$, with uniform distribution over such subsets (a *mini-batch*). Then we set for $\theta \in \mathbb{R}^d$ and for z such a subset

$$\mathcal{R}_{\text{mb}}(\theta, z) := \frac{1}{|z|} \sum_{i \in z} \big| y_i - \psi(x_i; \theta) \big|^2 \qquad \text{and} \qquad g(\theta, z) := -\nabla_\theta \mathcal{R}_{\text{mb}}(\theta, z), \tag{3.15}$$

where $|z|$ is the number of elements of z. Since each combination of elements of $\{1, \dots, N\}$ has the same probability of being selected, this choice indeed satisfies the bias-free property

$$\mathbb{E} g(\theta, Z) = -\nabla \mathcal{R}_N(\theta).$$

3.4 The Robbins-Monro theorem

3.4.1 Gronwall-based convergence proofs

Gronwall's lemma is a staple of many proofs of convergence. There are many different versions [Dra03] but the following one is sufficient for us:

Lemma 3.4.1. *Let $z : [0, \infty) \to [0, \infty)$ satisfy*

$$z(t) \le \alpha(t) + \int_0^t \beta(s) z(s)\, ds \qquad \textit{for all } t \ge 0, \tag{3.16}$$

for some functions $\alpha, \beta : [0, \infty) \to \mathbb{R}$. Assume that α is non-decreasing. Then

$$z(t) \le \alpha(t) \exp\Big(\int_0^t \beta(s)\, ds \Big) \qquad \textit{for all } t \ge 0.$$

See Wikipedia for a proof.

This lemma can for instance be used to estimate differences between solutions of ODEs, as follows. Let u_1 and u_2 be solutions of

$$\dot u_1 = F_1(u_1), \qquad \dot u_2 = F_2(u_2),$$

or in integral form,

$$u_1(t) = u_1(0) + \int_0^t F_1(u_1(s))\, ds, \qquad u_2(t) = u_2(0) + \int_0^t F_2(u_2(s))\, ds.$$

We want to apply Gronwall's lemma to $z(t) := |u_1(t) - u_2(t)|$ Assuming for simplicity that they start at the same point, $u_1(0) = u_2(0)$, then we estimate

$$\begin{aligned} z(t) = |u_1(t) - u_2(t)| &= \left| \int_0^t (F_1(u_1(s)) - F_2(u_2(s)))\, ds \right| \\ &\le \int_0^t |F_1(u_1(s)) - F_1(u_2(s))|\, ds + \int_0^t |F_1(u_2(s)) - F_2(u_2(s))|\, ds. \end{aligned}$$

We now need to manipulate—estimate—the two integrals into something resembling (3.16). For the first integral we assume that f_1 is Lipschitz continuous,

$$|F_1(u) - F_1(v)| \le L|u - v| \qquad \text{for all } u, v,$$

by which we find

$$\int_0^t |F_1(u_1(s)) - F_1(u_2(s))|\, ds \le L \int_0^t |u_1(s) - u_2(s)|\, ds = L \int_0^t z(s)\, ds.$$

For the second integral we make an assumption on $F_1 - F_2$, for instance

$$\sup_v |F_1(v) - F_2(v)| =: S < \infty.$$

Then

$$\int_0^t |F_1(u_2(s)) - F_2(u_2(s))|\, ds \le St.$$

Combining we find

$$z(t) \le St + L \int_0^T z(s)\, ds$$

which by Lemma 3.4.1 implies

$$z(t) \le S\, t\, e^{Lt} \qquad \text{for } t \ge 0. \tag{3.17}$$

This estimate is the main result of this calculation.

From this we learn a number of things:

- Since in our case $L > 0$, this estimate (3.17) is very bad for large t. This is a general feature of Gronwall estimates: unless you are lucky and have $L < 0$, Gronwall estimates are only useful for finite time, not for $t \to \infty$. (See [RRT17] for an example of how to overcome this limitation: they add explicit, isotropic noise to SGD and leverage logarithmic Sobolev inequalities).
- For fixed t, the estimate (3.17) gets better when $S \to 0$. Since S measures the deviation between F_1 and F_2, this estimate characterizes how much the difference $F_1 \ne F_2$ can lead to differences in the solutions, even if they start at the same point.
- Combining these two remarks: if we want to use Gronwall's Lemma to prove a convergence statement of the type 'u_2 converges to u_1', then (a) this can only be done over a finite length of time, $0 \le t \le T$, and (b) we will need to show that $S \to 0$.

3.4.2 The Robbins-Monro theorem

The Robbins-Monro theorem uses exactly the Gronwall-type argument outlined above. It is trickier than the example discussed above because of two aspects:

1. The Robbins-Monro algorithm does not generate a solution of an ODE but a sequence of points;
2. It also has a random component.

Nonetheless, both these additional difficulties can be dealt with.

The Robbins-Monro algorithm is

$$\theta_{n+1} = \theta_n + \alpha_n \Big[F(\theta_n) + \eta(\theta_n, \zeta) \Big].$$

Here

$$F : \mathbb{R}^d \to \mathbb{R}^d \qquad \text{is Lipschitz continuous,} \tag{3.18}$$

the step sizes $\alpha_n > 0$ satisfy

$$\sum_{n=1}^{\infty} \alpha_n = +\infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2 < \infty, \tag{3.19}$$

and η is a function of θ and of a random variable ζ, satisfying

$$\forall \theta : \qquad \mathbb{E}\eta(\theta, \zeta) = 0 \qquad \text{and} \qquad \text{Var } \eta(\theta, \zeta) \le K(1 + \|\theta\|^2)$$

for some $K > 0$ independent of θ.

The algorithm is closely connected to the ODE

$$\dot{x} = F(x). \tag{3.20}$$

The theorem characterizes the long-term behaviour of θ_n in terms of the long-term behaviour of this ODE, which in turn is characterized by a *globally asymptotically stable set* H. This set H is characterized by two properties:

1. Every solution of the ODE (3.20) converges to H as $t \to \infty$ (H is *globally attracting*);
2. Solutions that start near H remain close to H (H is *stable*).

More precisely, we take H to be *the smallest set* with these properties. (Without this condition $H = \mathbb{R}^d$ would also do). Note that H contains every equilibrium, including every saddle point; in general H can also contain non-equilibria, for instance limit cycles, although in the gradient case $F = -\nabla L$ these typically do not happen.

Theorem 3.4.2 ([BPP13, Th. 3.3], [RM51])**.** *Let the assumptions above be satisfied. Then the sequence θ_n converges to H as $n \to \infty$.*

Idea of the proof. The proof has two main elements:

1. One shows that interpolated versions of θ_n stay close to solutions of the ODE $\dot{x} = F(x)$ over finite stretches of time $[t, t+T]$;
2. One then uses this to show that θ_n converges to H.

The second point seems reasonable at first sight: if θ_n behaves like x, then the long-term behaviour of θ_n should behave like the long-term behaviour of x. However, this actually is non-trivial, and it hinges on the stability of H.

We only discuss the first point, which uses a Gronwall argument. We switch notation from the sequence of iterates θ_n to a function $\overline{x}$ defined for all $t \ge 0$. Define the sequence of times t_n by $t_0 = 0$ and $t_{n+1} = t_n + \alpha_n$, i.e.

$$t_n := \sum_{i=1}^{n-1} \alpha_i.$$

The conditions (3.19) on α_n imply that $t_n \to \infty$ as $n \to \infty$ and that $t_{n+1} - t_n \to 0$.

Fix an initial point θ_0. We define $x : [0, \infty) \to \mathbb{R}^d$ as the solution of the ODE (3.20) with initial datum θ_0:

$$\dot{x} = F(x), \qquad x(0) = \theta_0.$$

We write $\overline{x} : [0, \infty) \to \mathbb{R}^d$ for the linearly-interpolated version of θ_n, by setting

$$\overline{x}(t_n) := \theta_n, \qquad \text{and } \overline{x} \text{ is affine for } t_n \le t \le t_{n+1}.$$

We will show below that we can control an expression like

$$\sup_{0\le t\le T} |x(t) - \overline{x}(t)|$$

in terms of two properties of the step sizes,

$$\max_{n\ge 1} \alpha_n \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2. \tag{3.21}$$

By repeating the argument not starting at θ_0 but at θ_m, the corresponding quantities

$$\max_{n\ge m} \alpha_n \qquad \text{and} \qquad \sum_{n=m}^{\infty} \alpha_n^2$$

can be made as small as necessary. This is the basis of the first claim, that interpolated versions of θ_n stay close to solutions of the ODE.

To give an idea of the proof without too many technical restrictions, we make the following simplifications:

- $\sup_t |x(t)| =: C_1 < \infty$;
- the noise term $\eta(\theta, \zeta)$ is given by a θ-independent sequence of independent random variables X_n, with $\mathbb{E}X_n = 0$ and $\operatorname{Var} X_n = 1$.

Since F is (Lipschitz) continuous, the bound on x implies that

$$\sup_t |F(x(t))| =: C_2 < \infty.$$

With the setup above, we have

$$x(t) = \theta_0 + \int_0^t F(x(s))\, ds \tag{3.22}$$

$$\overline{x}(t) = \theta_0 + \int_0^t F(\overline{x}([s]))\, ds + \int_0^t Y_s\, ds. \tag{3.23}$$

The second line needs some explanation. By the definition of $\overline{x}$ and θ_n we have for $t_n \le t \le t_{n+1}$

$$\frac{\overline{x}(t_{n+1}) - \overline{x}(t_n)}{\alpha_n} = F(\overline{x}(t_n)) + X_n \overset{(*)}{=} F(\overline{x}([t]) + Y_t. \tag{3.24}$$

In $(*)$ we have introduced the notation $[t]$ for the largest t_n satisfying $t_n \le t$, and we have defined Y_t to be the t-dependent version of X_n:

$$Y_t := X_n \text{ for } n \text{ such that } t_n = [t].$$

Since $\overline{x}$ is affine on the interval $t_n \le t \le t_{n+1}$, the left-hand side in (3.24) equals $\dot{\overline{x}}(t)$, and by integrating we find (3.23).

Following the Gronwall setup above it would be natural to consider

$$z(t) = |x(t) - \overline{x}(t)|,$$

but if we try this then we are forced to estimate $|\overline{x}(t) - \overline{x}([t])|$, which has a random component that may be large. Instead it is slightly simpler to consider the function

$$z(t) := \big|x([t]) - \overline{x}([t])\big|,$$

because then we will need to control $|x(t) - x([t])|$, which is nonrandom. We estimate

$$\begin{aligned} z(t) &= \left| \int_0^{[t]} \Big[F(x(s)) - F(\overline{x}([s])) - Y_s \Big] \, ds \right| \\ &\le \int_0^{[t]} \Big| F(x(s)) - F(\overline{x}([s])) \Big| \, ds + \left| \int_0^{[t]} Y_s \, ds \right| \end{aligned} \tag{3.25}$$

The first integral is estimated by using

$$\begin{aligned} |F(x(s)) - F(\overline{x}([s]))| &\le L|x(s) - \overline{x}([s])| \\ &\le L|x(s) - x([s])| + L|x([s]) - \overline{x}([s])| \\ &\le L|x(s) - x([s])| + Lz(s). \end{aligned}$$

We estimate the first term by using the bound on F,

$$|x(s) - x([s])| = \left| \int_{[s]}^s \dot{x}(\sigma) \, d\sigma \right| = \left| \int_{[s]}^s F(x(\sigma)) \, d\sigma \right| \le C_2(s - [s]) \le C_2 \max_n \alpha_n.$$

We estimate the integral of Y in (3.25) by writing

$$\int_0^{[t]} Y_s \, ds = \sum_{n: t_n \le [t]} \alpha_n X_n$$

which implies that the expectation of the integral is zero, and that

$$\operatorname{Var} \int_0^{[t]} Y_s \, ds = \sum_{n: t_n \le [t]} \alpha_n^2 \operatorname{Var} X_n = \sum_{n: t_n \le [t]} \alpha_n^2 \le \sum_{n=1}^\infty \alpha_n^2.$$

Combining the various estimates we find

$$z(t) \le L \int_0^t z(s) \, ds + C_2 L t \max_n \alpha_n + \sum_{n=1}^\infty \alpha_n^2 = at + b + L \int_0^t z(s) \, ds.$$

Gronwall's Lemma 3.4.1 gives the estimate

$$z(t) \le (at + b) e^{Lt} = e^{Lt} \Big[t C_2 L \max_n \alpha_n + \sum_{n=1}^\infty \alpha_n^2 \Big].$$

If t ranges over an interval $[0, T]$, then this provides the uniform-in-time estimate

$$\sup_{0 \le t \le T} z(t) \le e^{LT} \Big[T C_2 L \max_n \alpha_n + \sum_{n=1}^\infty \alpha_n^2 \Big].$$

For a fixed interval length T, this provides the estimate of z in terms of the two measures of the step sizes in (3.21), as announced. □

3.5 SGD is close to an SDE

In (3.14–3.15) we introduced the SGD algorithm

$$\theta(t_{k+1}) = \theta(t_k) + \eta_k g(\theta(t_k), Z_k), \tag{3.26}$$

where the sequence of random variables $(Z_k)_k$ are i.i.d. with the property that for each θ

$$\mathbb{E}g(\theta, Z) = -\nabla\mathcal{R}_N(\theta).$$

In this section we compare the iterates of this SGD algorithm with the solutions of the SDE

$$d\vartheta_t = b(\vartheta_t)dt + \sigma(\vartheta_t, t)dW_t. \tag{3.27}$$

If these two stochastic processes are 'close' in some sense, then we will be able to generate understanding of the SGD algorithm by studying the solutions of the SDE, as we shall see below.

In (3.27) the parameters b and σ are still to be chosen. Since the expectation of $\sigma(\vartheta_t)dW_t$ is zero (this is part of the definition of the Itô integral, which in turn is part of the interpretation of (3.27)) we have

$$\frac{d}{dt}\mathbb{E}\,\vartheta_t = \mathbb{E}\,b(\vartheta_t).$$

For the SGD (3.26) we have

$$\frac{\mathbb{E}\big[\theta(t_{k+1}) - \theta(t_k)\big|\theta(t_k)\big]}{t_{k+1} - t_k} = \frac{\mathbb{E}\big[\theta(t_{k+1}) - \theta(t_k)\big|\theta(t_k)\big]}{\eta_k} = \mathbb{E}\big[g(\theta(t_k), Z_k)\big|\theta(t_k)\big] = -\nabla\mathcal{R}_N(\theta(t_k)).$$

Therefore it seems natural to choose

$$b(\vartheta) := -\nabla\mathcal{R}_N(\vartheta). \tag{3.28}$$

As for the choice of σ, we have the following characterization of the 'noise' generated by g in the case of the minibatch SGD:

Lemma 3.5.1 ([HLLL19, Prop. 1])**.** *Let $g(\theta, Z)$ be given as in* (3.15)*, i.e.*

$$\mathcal{R}_{\mathrm{mb}}(\theta, z) := \frac{1}{|z|}\sum_{i\in z}\big|y_i - \psi(x_i;\theta)\big|^2 \qquad \text{and} \qquad g(\theta, z) := -\nabla_\theta\mathcal{R}_{\mathrm{mb}}(\theta, z).$$

Then for each θ, writing $g = g(\theta, Z)$ for short,

$$\operatorname{cov} g(\theta, Z) := \mathbb{E}\big[(g - \mathbb{E}g)(g - \mathbb{E}g)^T\big] = \Big(\frac{1}{M} - \frac{1}{N}\Big)\Sigma_0(\theta),$$

where Σ_0 is the unbiased estimator of the covariance matrix based on the full set of N data points,

$$\Sigma_0(\theta) := \frac{1}{N-1}\sum_{i=1}^N \nabla_\theta\psi(x_i;\theta)\,\big(\nabla_\theta\psi(x_i;\theta)\big)^T.$$

This suggests the choice

$$\sigma(\uptheta, t) := \sqrt{\eta_t\Big(\frac{1}{M} - \frac{1}{N}\Big)\Sigma_0(\uptheta)}, \tag{3.29}$$

where η_t is the value of η_k at k corresponding to t. Here the square root $\sqrt{A}$ of a positive semidefinite symmetric matrix A is the unique positive semidefinite symmetric matrix B such that $B^2 = A$ (the 'square root' of A).

Hu, Li, Li, and Liu [HLLL19] prove that (3.26) and (3.27) are close. They describe that the choices (3.28) and (3.29) work, but that a slight modification of (3.28) works even better:

Theorem 3.5.2 ([HLLL19, Prop. 1])**.** *Fix $T > 0$, and assume that $\mathcal{R}_N$ is sufficiently differentiable. For $\eta > 0$, let $\uptheta$ be the solution of* (3.26) *on* $[0, T]$ *with σ given by* (3.29) *and b given by*

$$b(\uptheta) := -\nabla\mathcal{R}_N(\uptheta) - \frac{1}{4}\eta\nabla|\nabla\mathcal{R}_N(\uptheta)|^2. \tag{3.30}$$

Let θ be the solution of the SGD algorithm (3.26) *with times $t_k := k\eta$.*

If both $\uptheta$ and θ are started at the same point θ_0, then they are close in the following sense. Let $\varphi : \mathbb{R} \to \mathbb{R}$ have bounded derivatives up to order 6*. Then there exist constants $C > 0$ and $\eta_0 > 0$ such that*

$$\big|\mathbb{E}\,\varphi(\theta(t_k)) - \mathbb{E}\,\varphi(\uptheta(t_k))\big| \le C\eta^2 \qquad \text{for all } 0 < \eta < \eta_0 \text{ and all } 0 \le t_k \le T.$$

See also [LTE17, Th. 1] and [CYBJ20]; older work includes [BM99].

If the step sizes can be chosen to decrease in the same way as in the Robbins-Munro result (see (3.19)) then one can use the closeness of SGD and SDE to prove properties of the long-term behaviour of the limit, similar to but more explicit than the Robbins-Munro convergence Theorem 3.4.2. Wojtowiytsch [Woj21, Th. 2.2] gives a nice informal summary of the results of [MHKC20], which we reproduce here:

Informal theorem 3.5.3. *Let the sequence $\theta(t_k)$ be given by* (3.26)*. Under suitable conditions on the objective function f and a condition of the type $\mathbb{E}|g - \nabla f|^2 \le \sigma^2$, the following holds: If $\eta_k = \eta_0/(t_k + t_0)^p$ for some $p \in (1/2, 1]$, then*

1. *$f(\theta(t_k))$ converges to a random variable Y almost surely and Y lies in the set of critical values of f almost surely;*
2. *$\sup_k |\theta(t_k)| < \infty$ almost surely;*
3. *For every trajectory of SGD, there exists a connected component $\mathcal{X}$ of the set of critical points of f such that $\lim_{k\to\infty} \operatorname{dist}(\theta(t_k), \mathcal{X}) = 0$.*
4. *If S is a connected component of the set of critical points of f such that $d^2 f(\theta)$ has a negative eigenvalue for all $\theta \in S$ (making S a 'ridge manifold'), then $\lim_{k\to\infty} \operatorname{dist}(\theta(t_k), S) = 0$ almost surely.*

In words: the iterates $\theta(t_k)$ converge almost surely to a connected component of the set of critical points of f, and this connected component can not be everywhere unstable.

Note the difference with Theorem 3.4.2: while Theorem 3.4.2 asserts convergence to some implicitly defined set H, Theorem 3.5.3 characterizes the limiting behaviour in terms of connected components of the set of critical points of f, which is a much more explicit description. Also note that the range $1/2 < p \leq 1$ corresponds exactly to the range in which the conditions (3.19) hold.

On the other hand, in both theorems it can not be excluded that the iterates continue to 'wander' along the critical set. See [LWA21] for an explicit characterization of such 'wandering' in the case of added label noise.

3.6 Convergence of gradients to zero

One can also focus solely on the gradients of the loss function themselves. In this section we explore two results of this type.

3.6.1 Fixed stepsize sequence

Bertsekas and Tsitsiklis [BT00] consider the following setting: given a function $L : \mathbb{R}^n \to \mathbb{R}$ find $\min_{\theta \in \mathbb{R}^n} L(\theta)$ using

$$\theta_{k+1} = \theta_k + \eta_k(s_k + w_k), \tag{3.31}$$

where η_k is a stepsize, s_k a direction and w_k either a deterministic or random error. They assume the following,

1. $L \in C^1(\mathbb{R}^n)$
2. ∇L is Lipschitz continuous, i.e. $\exists M > 0 : \|\nabla L(x) - \nabla L(y)\| \leq M\|x - y\| \; \forall \; x, y \in \mathbb{R}^n$,
3. the stepsize sequence satisfies $\sum_{k=0}^{\infty} \eta_k = \infty$ and $\sum_{k=0}^{\infty} \eta_k^2 < \infty$,
4. the direction s_k satisfies $\exists c_1, c_2 > 0 : \quad \forall k : \quad c_1\|\nabla L(\theta_k)\|^2 \leq -\nabla L(\theta_k)^T s_k$ and $\|s_k\| \leq c_2\|\nabla L(\theta_k)\|$,
5. the error w_k satisfies either
   - $\exists p, q > 0 : \; \forall k : \; \|w_k\| \leq \eta_k(q + p\|\nabla L(\theta_k)\|)$ if w_k is deterministic
   - $\mathbb{E}[w_k|\mathcal{F}_k] = 0, \quad \exists A > 0 : \; \forall k : \mathbb{E}[\|w_k\|^2|\mathcal{F}_k] \leq A(1 + \|\nabla L(\theta_k)\|^2)$ a.s. if w_k is a random.

Their main result is

Theorem 3.6.1 ([BT00, Prop. 1&3])**.** *Let θ_k be a sequence generated by* (3.31)*. Under the aforementioned assumptions, we have that either*

- *$L(\theta_k) \to -\infty$ a.s. or*
- *$L(\theta_k)$ converges to a finite value a.s. and $\nabla L(\theta_k) \to 0$ a.s.*

Furthermore, every limit point of θ_k is a stationary point of L.

The proof makes use of the following lemma.

Lemma 3.6.2 ([BT00, Lemma 1])**.** *Let Y_t, W_t, Z_t be three sequences satisfying $W_k \geq 0$, $\lim_{T\to\infty} \sum_{i=0}^{T} Z_i < \infty$ and*

$$Y_{k+1} \leq Y_k - W_k + Z_k \quad \forall \; k = 0, 1, \ldots.$$

Then either

- *$Y_k \to -\infty$ or,*
- *Y_k converges to a finite value and $\sum_{k=0}^{\infty} W_k < \infty$.*

Proof of Theorem 3.6.1. Let $x, z \in \mathbb{R}^n$. Define the function $g : \mathbb{R}^n \to \mathbb{R}$ as

$$g(\alpha) = L(x + \alpha z).$$

Then

$$\begin{aligned}
L(x+z) - L(x) &= g(1) - g(0) \\
&= \int_0^1 \frac{dg}{d\alpha}(\alpha) d\alpha \quad \text{(Fundamental Thm of Calculus)} \\
&= \int_0^1 z^T \nabla L(x + \alpha z) d\alpha \quad \text{(chain rule)} \\
&\leq \int_0^1 z^T \nabla L(x) + \left| \int_0^1 z^T \left(\nabla L(x + \alpha z) - \nabla L(x) \right) d\alpha \right| \\
&\leq z^T \nabla L(x) + \int_0^1 \|z\| \|\nabla L(x + \alpha z) - \nabla L(x)\| d\alpha \quad \text{(triangle + Cauchy-Schwartz ineq.)} \\
&\leq z^T \nabla L(x) + \frac{M\|z\|^2}{2} \quad \text{(by } M\text{-Lipschitz continuity of } \nabla L\text{).}
\end{aligned}$$

Rearranging and setting $x = \theta_k$ and $z = \eta_k(s_k + w_k)$ gives

$$L(\theta_{k+1}) \leq L(\theta_k) + \eta_k (s_k + w_k)^T \nabla L(\theta_k) + \eta_k^2 \frac{M}{2} \|s_k + w_k\|^2.$$

By careful estimation using the bounds we made in the assumptions, we obtain

$$L(\theta_{k+1}) \leq L(\theta_k) - \beta_1 \eta_k \|\nabla L(\theta_k)\|^2 + \beta_2 \eta_k^2,$$

for some constants $\beta_1, \beta_2 > 0$. We can now apply the Lemma with

$$Y_k = L(\theta_k), \quad W_k = \eta_k \|\nabla L(\theta_k)\|^2, \quad Z_k = \eta_k^2.$$

By the Lemma we can conclude that either $Y_k \to -\infty$ or Y_k converges to a finite value. In the last case, the Lemma states that

$$\sum_{k=0}^{\infty} \eta_k \|\nabla L(\theta_k)\|^2 < \infty.$$

By assumption, $\sum \eta_k$ diverges, so we must have $\liminf_{k\to\infty} \|\nabla L(\theta_k)\| = 0$. To show that the limsup also converges to 0, one can argue by contradiction. $\square$

3.6.2 Adaptive step sizes

Li and Orabona [LO19] consider convergence of the gradient of the function using *adaptive* step sizes: the step size depends on the history of the gradient estimate. In SGD the performance of adaptive step sizes turns out to be much better than that of fixed ones.

Given a function $L : \mathbb{R}^n \to \mathbb{R}$ we aim to find $\min_{\theta \in \mathbb{R}^n} L(\theta)$ by

$$\begin{aligned} \theta_{k+1} &= \theta_k - \eta_k g(\theta_k, Z_k), \text{ where} \\ \eta_k &= \frac{\alpha}{\left(\beta + \sum_{i=1}^{k-1} \|g(\theta_i, Z_i)\|^2\right)^{1/2+\varepsilon}} \qquad \left(\alpha, \beta > 0,\ \varepsilon \in (0, \frac{1}{2}]\right) \end{aligned}$$

The assumptions for their result are

1. $L \in C^1(\mathbb{R}^n)$ is bounded from below and Lipschitz continuous,
2. ∇L is Lipschitz continuous,
3. $g(\theta, Z)$ is an unbiased estimator of $\nabla L(\theta)$,
4. $\exists S > 0 : \forall \theta : \|g(\theta, Z) - \nabla L(\theta)\| \leq S$.

Theorem 3.6.3 ([LO19, Theorem 1]). *Under these assumptions,* $\lim_{k\to\infty} \nabla L(\theta_k) = 0$ *a.s. Moreover,*

$$\lim_{k\to\infty} \|\nabla L(\theta_k)\|^2 k^{1/2-\varepsilon} = 0 \text{ a.s.}$$

See e.g. [RSS12] for convergence proofs for stochastic gradient descent for convex functions.

Chapter 4

Generalisation

4.1 Local minima and metastability

4.1.1 "Bad local minima slow down convergence to good local minima"

Optimization of non-convex functions suffers from the problem of *metastability*, the phenomenon that stochastic processes, such as the optimization algorithms that we have seen, can show different behaviour at small time scales and at longer time scales. A canonical example is an of the type SDE (3.27), with a drift b that is generated by a double-well potential: $b(\theta) = -\Phi'(\theta)$, where the double-well potential $\Phi(\theta) = \frac{1}{4}(\theta^2 - 1)^2$ is shown in Figure 4.1(a). In the small-noise limit $\sigma \to 0$ the SDE becomes the ordinary differential equation

$$\frac{d\theta}{dt} = -\Phi'(\theta),$$

which is the 'gradient-flow' evolution driven by Φ that we also have seen above in (3.6).

As we have seen, solutions of this equation converge towards local minima of Φ and stay there forever. However, upon turning on the noise, $\sigma > 0$, the behaviour changes in two ways. On the shorter time scale, in Figure 4.1(c), the noise causes θ_t to fluctuate around the well at $\theta = -1$. This fluctuation is a form of equilibrium between the noise that tends to move θ_t away from its current position and the restoring force $-\Phi'(\theta_t)$ that—at least locally—pushes θ_t back to the well $\theta = -1$. (The precise behaviour can be described in terms of a *quasi-stationary distribution* [CMSM12].) On longer time scales, however, as shown in Figure 4.1(d), the noise causes θ_t to jump out of the left well and into the right one, and after that to continue to jump to and fro between the two wells. At random times the process jumps between the wells, and between two jumps, the process appears to equilibrate in its current well.

This *multi-scale* behaviour is called *metastable*: the fluctuations inside a single well, as in Figure 4.1(c), are stable on intermediate time scales, but unstable on long time scales.

The same metastability can be recognized in the time-dependent distribution of for this system, shown in Figure 4.2. At short time scales (Figure 4.2(a)), the initial delta-function relaxes into a single hill in the left well, representing a form of local-in-time equilibrium in the left well. At longer time scales, in Figure 4.2(b), the distribution slowly 'seeps' into the second well, ultimately dividing itself equally over the two wells (Figure 4.2(c)).

The theory of metastability is particularly well developed for SDEs. The core result is that the

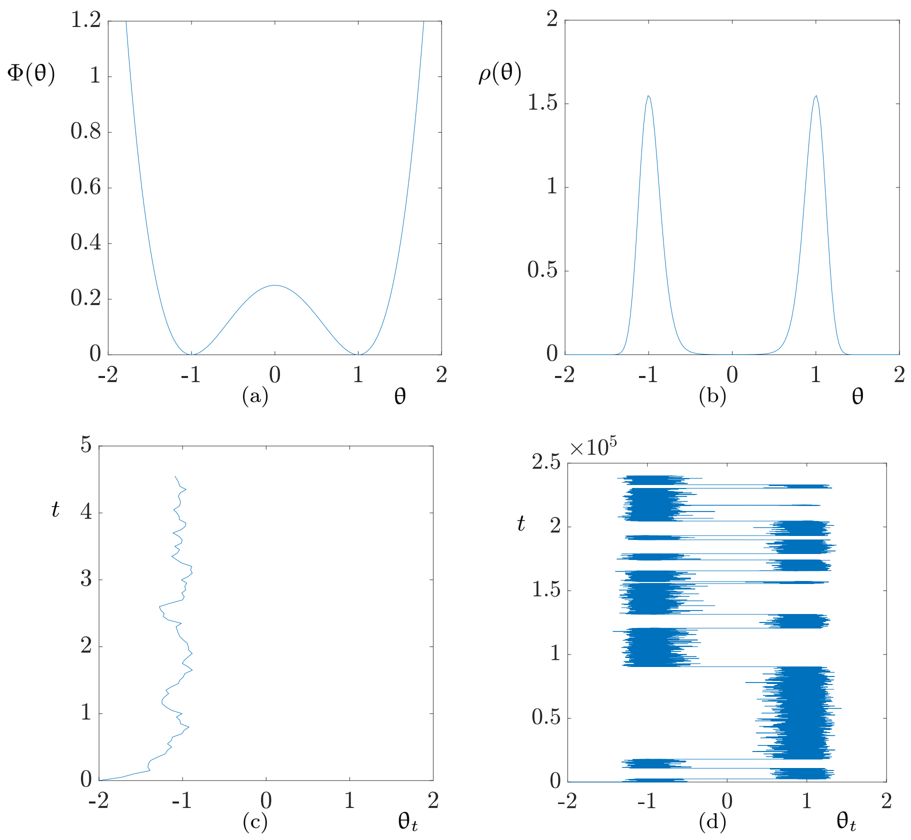


Figure 4.1: (a) The potential $\Phi(\theta) = \frac{1}{4}(\theta^2 - 1)^2$; (b) the invariant measure $\rho(\theta) = Z^{-1}\exp(-2\Phi(\theta)/\sigma^2)$, with $\sigma = 0.25$. Figures (c) and (d) show the distribution of θ_t with $b(\theta) = -\Phi'(\theta)$, plotted on two different time scales. .

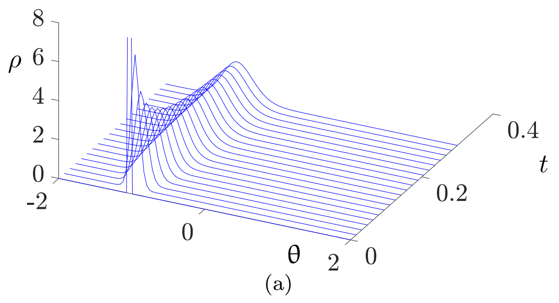
expected time for θ_t to make a transition from the left well to the right well scales as

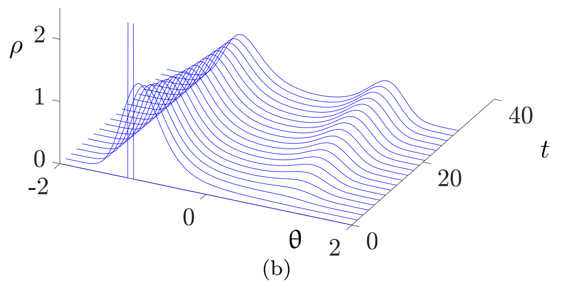
$$\exp\Big(-\frac{2\delta\Phi}{\sigma^2}\Big), \tag{4.1}$$

where $\delta\Phi$ is the energy barrier separating the two wells (in this case $\delta\Phi = \Phi(0) - \Phi(-1)$).

This is relevant for the discussion about training neural networks, because we have seen in Section 3.5 that one can approximate the SGD algorithm by an SDE, in which the noise scales with the step size (or learning rate). In many algorithms the step size tends to zero, either chosen by hand to satisfy (3.19) or by an adaptive-stepsize algorithm such as AdaGrad. By (3.29) this implies that the noise σ in an approximating SDE vanishes, and this in turn makes the expected time to transition (4.1) extremely long.

Summarizing, if one performs an SGD algorithm in which one eventually reduces the step size to zero (as suggested by all the theoretical results above) then at some point the algorithm will stop wandering around the parameter space, and get 'stuck' instead in a local minimum—with no guarantee that this local minimum may be a global minimum. This is the argument why metastability is a problem.





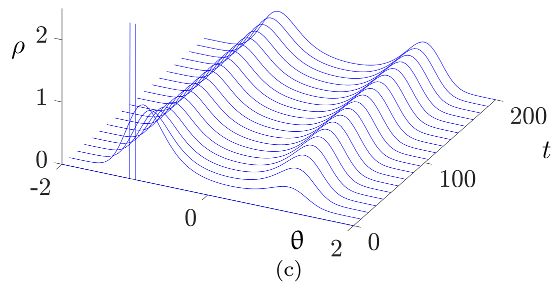


Figure 4.2: Solutions of the Fokker-Planck equation (**??**) with $\sigma = 0.25$ and $b(x) = -\Phi'(x)$ with $\Phi(x) = \frac{1}{4}(x^2-1)^2$. The same solution is plotted at three different time scales.

4.1.2 "There are no bad local minima"

There are separate arguments that suggest that yes, the algorithm may get stuck in some local minimum—of which there may be many—but it is reasonable to expect that this local minimum *also is a global minimum.*

The first result of this type appears to be by Dauphin et al. [DPG$^+$14], who appeal to results in the theory of Gaussian measures on spaces of functions (say ϕ) to claim that local minima are concentrated at ϕ-levels very close to the global minimum $\min \phi$. The basis of this argument for Gaussian random functions is a large-deviation principle in the limit of infinite dimensions. Specifically,

- A uniformly randomly selected local minimum has a value that concentrates exponentially, as dimension tends to infinity, around $\min \phi$ (where the range of ϕ is normalized to be independent in expectation of the dimension).
- Vice versa, at relative ϕ-levels (relative to the range of ϕ) different from minimal, the distribution of the index of the Hessian of a critical point (the relative number of negative eigenvalues) concentrates onto a value larger than zero.

There is no reason why the functions generated by a neural network would resemble 'typical' Gaussian random functions, but the authors investigated the neural-network case experimentally and found a similar concentration effect. This idea also is supported by the experimental observation that for sufficiently overparametrized networks the empirical loss can be trained to zero.

This led to the axiom 'there are no bad local minima', which should be interpreted as 'it's easy to find local minima of the empirical loss with a loss value close to zero'.

4.1.3 "Flat local minima generalize better"

This 'axiom' made the attention shift to the following question: How well do local minima of the empirical loss generalize? That is, how large or small is the *expected* loss of any given minimum of the empirical loss?

There is an old idea that 'flat' or 'broad' minima might generalize better than 'sharp' or 'narrow' ones:

1. Hochreiter and Schmidhuber [HS97] give heuristic arguments based on Bayesian statistics (flat minima correspond to 'fat' posterior distributions) and complexity arguments (low-complexity functions can only lead to flat minima).

This idea really took off after 2016:

2. Keskar et al. [KMN+16] use this idea to explain the observation that increasing mini-batch size can lead to reduced generalization: they claim that large-batch SGD prefers sharper minima and small-batch SGD wider minima, and back this up with numerical evidence.
3. Wu, Zhu, and Weinan E [WZE17] constructed global empirical-loss minima that performed no better than random on the test set, and found that they typically had much larger Hessian eigenvalues.

There does seem to be some confusion of concepts:

- One class of measures of 'sharpness' makes use of the Hessian matrix, such as the determinant [JKA+17] or the maximal eigenvalue [KMN+16]. This class of measures is completely local.
- Wu-Zhu-E [WZE17] instead argue that the size of the basin of attraction (a global property) is more important than the Hessian of the minimizer (a local property).

On the other hand, in many cases the two seem to be correlated.

In addition, the high dimensionality seems to make comparison difficult; Li et al. [LXT+17] argue that visualising loss landscapes in the neighbourhood of a critical point requires careful scaling in order to be meaningful.

4.1.4 "SGD prefers broader minima"

People started to build on this:

1. SGD seems to already prefer wider minima, but to further help the algorithm, Chaudhari et al. [CCS+19] proposed to have SGD minimize not f but the 'exponentially convoluted version'
$$F(\theta) := -\log \int_{\mathbb{R}^d} e^{-f(\theta')-\gamma|\theta'-\theta|^2}\, d\theta'$$
2. A related idea is Stochastic Langevin Gradient Descent, which adds a noise term to the SGD update; this allows to decouple mini-batch size from noise level [RRT17].

Chapter 5

Exercises

There are a couple of things that you can do.

Option 1: Experiment with a simple function-approximation neural network. This is what I would advise if you have no experience with neural networks; there is nothing better than 'playing around' with an example.

Google offers an online Python environment, *Colab*, in which many software dependencies have been taken care of. In addition, they provide a number of example 'notebooks' that you can use. A good one to start with is

`https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/classification.ipynb`

You use a colab notebook in a similar way to a jupyter notebook: you can edit any cell simply by typing, you use Command-Enter or Ctrl-Enter to execute a cell, and with +CODE you create a new cell for code.

This notebook walks you through the process of training a network to classify images of flowers. The database for this notebook is a selection of 3670 images of flowers, which are labelled into five classes (daisies, dandelions, roses, sunflowers, and tulips). The notebook documents the whole process, from importing appropriate libraries, through loading the data, inspecting the data, normalizing the images, constructing a network in Tensorflow, training the network, and inspecting the result. In addition, the notebook shows how incorporating data augmentation and dropout in the training can improve the quality of the trained network.

Here are some questions to think about and experiment with. Don't hesitate to use Google a lot to help you understand; it's the understanding in the end that counts.

1. What is the difference between loss and accuracy? Where are they specified in the code? What do you see in the plots of both against time?
2. Experiment with the trained network after the first training. Can you find examples of images that it misclassifies?
3. The first training session trains for '10 epochs'. What are epochs? Is it useful to train for shorter, or for longer? How do you know?

4. The first trained network ‘overfits’. What does that mean, and how can you recognize this?
5. The second training session incorporates both *data augmentation* and *dropout*. Work out what these are, and draw some pictures that would allow you to explain it to someone else. Experiment to answer the question: which of the two has the biggest effect on the training?
6. Study the structure of the first network that has been constructed. Make a picture of it. (You’ll need to research a few terms for this, such as a ‘convolution layer’ and a ‘max-pool layer’). Experiment with changing the setup; if you increase or decrease the ‘width’ of layers, or the depth of the network, how does this influence the training and the result?
7. Same questions for the second network. How is it different?

Option 2: Experiment with more specialized neural networks. I recommend this option if you have some experience with neural networks in general.

A nice example of a more specialized AI tool is the *Variational Autoencoder* (VAE). You can find a basic Colab notebook at

`https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/generative/cvae.ipynb`

Some online resources on VAEs are

`https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73`
`https://becominghuman.ai/variational-autoencoders-simply-explained-46e6f97947ed`
`https://en.wikipedia.org/wiki/Variational_autoencoder`

Here are some questions you might want to think about:

1. Describe, mathematically and in your own terms, what a VAE is and does.
2. Based on the Colab notebook above, explain the structure of the encoder and the decoder; how exactly do they deal with parametrized probability distributions? Which parameters are fixed and which are optimized during training?
3. Why does the encoder output the logarithm of the variance instead of the variance?
4. How could you improve the quality of the generated images?

Appendix A

Landau's O symbols and related notation

Consider some limit, e.g. $n \to \infty$, and consider sequences a_n, b_n in $\mathbb{R}$. We define the following:

$a_n = o(b_n)$	by	$\limsup_{n\to\infty} \left\lvert \frac{a_n}{b_n} \right\rvert = 0$	'asymptotically smaller than'
$a_n = O(b_n)$	by	$\limsup_{n\to\infty} \left\lvert \frac{a_n}{b_n} \right\rvert < \infty$	'no bigger than'
$a_n = \Theta(b_n)$	by	$0 < \liminf_{n\to\infty} \left\lvert \frac{a_n}{b_n} \right\rvert \leq \limsup_{n\to\infty} \left\lvert \frac{a_n}{b_n} \right\rvert < \infty$	'of the same order as'
$a_n = \Omega(b_n)$	by	$\liminf_{n\to\infty} \left\lvert \frac{a_n}{b_n} \right\rvert > 0$	'at least as large as'
$a_n = \omega(b_n)$	by	$\liminf_{n\to\infty} \left\lvert \frac{a_n}{b_n} \right\rvert = \infty$	'asymptotically larger than'

Bibliography

[BM99] V. S. Borkar and S. K. Mitter. A strong approximation theorem for stochastic recursive algorithms. *Journal of optimization theory and applications*, 100(3):499–513, 1999.

[BPP13] S. Bhatnagar, H. Prasad, and L. Prashanth. Stochastic approximation algorithms. In *Stochastic Recursive Algorithms for Optimization*, pages 17–28. Springer, 2013.

[Bre11] H. Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations.* Springer, New York, 2011.

[BT00] D. P. Bertsekas and J. N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM Journal on Optimization*, 10(3):627–642, 2000.

[BV04] S. P. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[CC18] A. L. Caterini and D. E. Chang. *Deep neural networks in a mathematical framework.* Springer, 2018.

[CCS$^+$19] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

[CMSM12] P. Collet, S. Martínez, and J. San Martín. *Quasi-Stationary Distributions: Markov Chains, Diffusions and Dynamical Systems.* Springer Science & Business Media, 2012.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[CYBJ20] X. Cheng, D. Yin, P. Bartlett, and M. Jordan. Stochastic gradient and Langevin processes. In *International Conference on Machine Learning*, pages 1810–1819. PMLR, 2020.

[DHM89] R. A. DeVore, R. Howard, and C. Micchelli. Optimal nonlinear approximation. *Manuscripta mathematica*, 63(4):469–478, 1989.

[DHP21] R. DeVore, B. Hanin, and G. Petrova. Neural network approximation. *Acta Numerica*, 30:327–444, 2021.

[DPG$^+$14] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv preprint arXiv:1406.2572*, 2014.

[Dra03] S. S. Dragomir. *Some Gronwall type inequalities and applications.* Nova Science Publishers New York, 2003. Downloaded from `https://rgmia.org/papers/monographs/standard.pdf`.

[Dud02] R. M. Dudley. *Real analysis and probability*, volume 74. Cambridge University Press, 2002.

[EW18] W. E and Q. Wang. Exponential convergence of the deep neural network approximation for analytic functions. *arXiv preprint arXiv:1807.00297*, 2018.

[FSB10] C. L. Frenzen, T. Sasao, and J. T. Butler. On the number of segments needed in a piecewise linear approximation. *Journal of Computational and Applied mathematics*, 234(2):437–446, 2010.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

[GP89] F. Girosi and T. A. Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Comput.*, 1(4):465–469, 1989.

[HLLL19] W. Hu, C. J. Li, L. Li, and J.-G. Liu. On the diffusion approximation of nonconvex stochastic gradient descent. *Annals of Mathematical Sciences and Applications*, 4(1), 2019.

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics.* Springer-Verlag, Berlin, 1993.

[Hor91] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[HS97] S. Hochreiter and J. Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[IP95] B. Igelnik and Y.-H. Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Transactions on Neural Networks*, 6(6):1320–1329, 1995.

[Isk18] A. Iske. *Approximation theory and algorithms for data analysis.* Springer, 2018.

[JKA+17] S. Jastrzębski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Three factors influencing minima in SGD. *arXiv preprint arXiv:1711.04623*, 2017.

[KMN+16] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[Kol57] A. N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114, pages 953–956. Russian Academy of Sciences, 1957.

[LLPS93] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[LO19] X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 983–992. PMLR, 2019.

[LP93] Y. V. Lin and A. Pinkus. Fundamentality of ridge functions. *Journal of Approximation Theory*, pages 295–311, 1993.

[LTE17] Q. Li, C. Tai, and W. E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pages 2101–2110. PMLR, 2017.

[LWA21] Z. Li, T. Wang, and S. Arora. What happens after SGD reaches zero loss?–a mathematical framework. *arXiv preprint arXiv:2110.06914*, 2021.

[LXT$^+$17] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[MHKC20] P. Mertikopoulos, N. Hallak, A. Kavis, and V. Cevher. On the almost sure convergence of stochastic gradient descent in non-convex problems. *Advances in Neural Information Processing Systems*, 33:1117–1128, 2020.

[MYD19] H. Montanelli, H. Yang, and Q. Du. Deep ReLU networks overcome the curse of dimensionality for bandlimited functions. *arXiv preprint arXiv:1903.00735*, 2019.

[PGEB18] D. Perekrestenko, P. Grohs, D. Elbrächter, and H. Bölcskei. The universal approximation power of finite-width deep ReLU networks. *arXiv preprint arXiv:1806.01528*, 2018.

[RM51] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[RR08] A. Rahimi and B. Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008.

[RRT17] M. Raginsky, A. Rakhlin, and M. Telgarsky. Non-convex learning via stochastic gradient Langevin dynamics: A nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703. PMLR, 2017.

[RSS12] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2012.

[Rud87] W. Rudin. *Real and Complex Analysis.* McGraw-Hill International Editions, Mathematics Series. McGraw-Hill, Inc., third edition, 1987.

[SH21] J. Schmidt-Hieber. The Kolmogorov–Arnold representation theorem revisited. *Neural Networks*, 2021.

[SSBD14] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[Sto82] C. J. Stone. Optimal global rates of convergence for nonparametric regression. *The annals of statistics*, pages 1040–1053, 1982.

[Tel15] M. Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

[Woj21] S. Wojtowytsch. Stochastic gradient descent with noise of machine learning type. Part I: Discrete time analysis. *arXiv preprint arXiv:2105.01650*, 2021.

[WZE17] L. Wu, Z. Zhu, and W. E. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

[Yar17] D. Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.