Mathematical theory of deep learning

Philipp Petersen[1] and Jakob Zech[2]

[1]Universität Wien, Fakultät für Mathematik, 1090 Wien, Austria,
philipp.petersen@univie.ac.at
[2]Universität Heidelberg, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, 69120 Heidelberg, Germany, jakob.zech@uni-heidelberg.de

October 14, 2024

Contents

Preface

This book serves as an introduction to the key ideas in the mathematical analysis of deep learning. It is designed to help students and researchers to quickly familiarize themselves with the area and to provide a foundation for the development of university courses on the mathematics of deep learning. Our main goal in the composition of this book was to present various rigorous, but easy to grasp, results that help to build an understanding of fundamental mathematical concepts in deep learning. To achieve this, we prioritize simplicity over generality.

As a mathematical introduction to deep learning, this book does not aim to give an exhaustive survey of the entire (and rapidly growing) field, and some important research directions are missing. In particular, we have favored mathematical results over empirical research, even though an accurate account of the theory of deep learning requires both.

The book is intended for students and researchers in mathematics and related areas. While we believe that every diligent researcher or student will be able to work through this manuscript, it should be emphasized that a familiarity with analysis, linear algebra, probability theory, and basic functional analysis is recommended for an optimal reading experience. To assist readers, a review of key concepts in probability theory and functional analysis is provided in the appendix.

The material is structured around the three main pillars of deep learning theory: Approximation theory, Optimization theory, and Statistical Learning theory. Chapter 1 provides an overview and outlines key questions for understand deep learning. Chapters 2 - 9 explore results in approximation theory, Chapters 10 - 13 discuss optimization theory for deep learning, and the remaining Chapters 14 - 16 address the statistical aspects of deep learning.

This book is the result of a series of lectures given by the authors. Parts of the material were presented by P.P. in a lecture titled “Neural Network Theory” at the University of Vienna, and by J.Z. in a lecture titled “Theory of Deep Learning” at Heidelberg University. The lecture notes of these courses formed the basis of this book. We are grateful to the many colleagues and students who contributed to this book through insightful discussions and valuable suggestions. We would like to offer special thanks to the following individuals:

Jonathan Garcia Rebellon, Jakob Lanser, Andrés Felipe Lerma Pineda, Martin Mauser, Davide Modesto, Martina Neuman, Bruno Perreaux, Johannes Asmus Petersen, Milutin Popovic, Tuan Quach, Lorenz Riess, Jakob Fabian Rohner, Jonas Schuhmann, Peter Školník, Matej Vedak, Simon Weissmann, Ashia Wilson.

Notation

In this section, we provide a summary of the notations used throughout the manuscript for the reader's convenience.

Symbol	Description	Reference
$\mathcal{A}$	vector of layer widths	Definition 12.1
$\mathfrak{A}$	a sigma-algebra	Definition A.1
$\operatorname{aff}(S)$	affine hull of S	(5.3.7)
$\mathfrak{B}_d$	the Borel sigma-algebra on $\mathbb{R}^d$	Section A.1
$\mathcal{B}^n$	B-Splines of order n	Definition 4.2
$B_r(x)$	ball of radius $r \geq 0$ around x in a metric space X	(B.1.1)
B_r^d	ball of radius $r \geq 0$ around $\mathbf{0}$ in $\mathbb{R}^d$	
$C^k(\Omega)$	k-times continuously differentiable functions from $\Omega \to \mathbb{R}$	
$C_c^\infty(\Omega)$	infinitely differentiable functions from $\Omega \to \mathbb{R}$ with compact support in Ω	
$C^{0,s}(\Omega)$	s-Hölder continuous functions from $\Omega \to \mathbb{R}$	
$C^{k,s}(\Omega)$	$C^k(\Omega)$ functions f for which $f^{(k)} \in C^{0,s}(\Omega)$	Definition 7.5
$f_n \xrightarrow{\text{cc}} f$	compact convergence of f_n to f	Definition 3.1
$\operatorname{co}(S)$	convex hull of a set S	(5.3.1)
$f * g$	convolution of f and g	
$\mathcal{D}$	data distribution	(1.2.4)/Section 14.1
$D^{\boldsymbol{\alpha}}$	partial derivative	
$\operatorname{depth}(\Phi)$	depth of Φ	Definition 2.1
$\varepsilon_{\text{approx}}$	approximation error	(14.2.3)
ε_{gen}	generalization error	(14.2.3)
ε_{int}	interpolation error	(14.2.4)
$\mathbb{E}[X]$	expectation of random variable X	(A.2.1)
$\mathbb{E}[X\|Y]$	conditional expectation of random variable X	Subsection A.3.3
$\mathcal{F}(f)$ or $\hat{f}$	Fourier transform of f	Definition B.15
$\mathcal{G}(S, \varepsilon, X)$	ε-covering number of a set $S \subseteq X$	Definition 14.10
Γ_C	Barron space with constant C	Section 8.1
$\nabla_x f$	gradient of f w.r.t. x	
$\oslash$	componentwise (Hadamard) division	

Continued on next page

Symbol	Description	Reference
$\otimes$	componentwise (Hadamard) product	
h_S	empirical risk minimizer for a sample S	Definition 14.5
Φ_L^{id}	identity ReLU neural network	Lemma 5.1
$\mathbb{1}_S$	indicator function of the set S	
$\langle \cdot, \cdot \rangle$	Euclidean inner product on $\mathbb{R}^d$	
$\langle \cdot, \cdot \rangle_H$	inner product on a vector space H	Definition B.9
$k_{\mathcal{T}}$	maximal number of elements shared by a single node of a triangulation	(5.3.2)
K^{LC}	neural tangent kernel for the LeCun initialization	Theorem 11.16
$\hat{K}_n(\boldsymbol{x}, \boldsymbol{x}')$	empirical tangent kernel	(11.3.4)
K^{NTK}	neural tangent kernel for the NTK initialization	Theorem 11.30
$\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$	loss landscape defining function	Definition 12.2
$\mathrm{Lip}(f)$	Lipschitz constant of a function f	(9.2.1)
$\mathrm{Lip}_M(\Omega)$	M-Lipschitz continuous functions on Ω	(9.2.4)
$\mathcal{L}$	general loss function	Section 14.1
$\mathcal{L}_{0-1}$	0-1 loss	Section 14.1
$\mathcal{L}_{\mathrm{ce}}$	binary cross entropy loss	Section 14.1
$\mathcal{L}_2$	square loss	Section 14.1
$L^p(\Omega)$	Lebesgue space over Ω	Section B.1.3
$\mathcal{M}$	piecewise continuous and locally bounded functions	Definition 3.1.1
$\mathcal{N}_d^m(\sigma; L, n)$	set of multilayer perceptrons with d-dim input, m-dim output, activation function σ, depth L, and width n	Definition 3.6
$\mathcal{N}_d^m(\sigma; L)$	union of $\mathcal{N}_d^m(\sigma; L, n)$ for all $n \in \mathbb{N}$	Definition 3.6
$\mathcal{N}(\sigma; \mathcal{A}, B)$	set of neural networks with architecture $\mathcal{A}$, activation function σ and all weights bounded in modulus by B	Definition 12.1
$\mathcal{N}^*(\sigma, \mathcal{A}, B)$	neural networks in $\mathcal{N}(\sigma; \mathcal{A}, B)$ with range in $[-1, 1]$	(14.5.1)
$\mathbb{N}$	positive natural numbers	
$\mathbb{N}_0$	natural numbers including 0	
$\mathrm{N}(\boldsymbol{m}, \boldsymbol{C})$	multivariate normal distribution with mean $\boldsymbol{m} \in \mathbb{R}^d$ and covariance $\boldsymbol{C} \in \mathbb{R}^{d \times d}$	

Continued on next page

Symbol	Description	Reference
$n_{\mathcal{A}}$	number of parameters of a neural network with layer widths described by $\mathcal{A}$	Definition 12.1
$\|\cdot\|$	Euclidean norm for vectors in $\mathbb{R}^d$ and spectral norm for matrices in $\mathbb{R}^{n\times d}$	
$\|\cdot\|_F$	Frobenius norm for matrices	
$\|\cdot\|_\infty$	∞-norm on $\mathbb{R}^d$ or supremum norm for functions	
$\|\cdot\|_p$	p-norm on $\mathbb{R}^d$	
$\|\cdot\|_X$	norm on a vector space X	
$\mathbf{0}$	zero vector in $\mathbb{R}^d$	
$O(\cdot)$	Landau notation	
$\omega(\eta)$	patch of the node η	(5.3.5)
$\Omega_\Lambda(c)$	sublevel set of loss landscape	Definition 12.3
$\mathcal{P}_n$	short for $\mathcal{P}_n(\mathbb{R}^d)$	
$\mathcal{P}_n(\mathbb{R}^d)$	space of multivariate polynomials of degree n in $\mathbb{R}^d$	Example 3.5
$\mathcal{P}$	short for $\mathcal{P}(\mathbb{R}^d)$	
$\mathbb{P}[A]$	probability of event A	Definition A.5
$\mathbb{P}[A\|B]$	conditional probability of event A given B	Definition A.3.2
$\mathbb{P}_X$	distribution of random variable X	Definition A.10
$\mathcal{P}(\mathbb{R}^d)$	space of multivariate polynomials in $\mathbb{R}^d$	Example 3.5
Φ^{lin}	linearization of a model around initialization	(11.3.1)
$\Phi_n^{\min}$	minimum neural network	Lemma 5.11
$\Phi_\varepsilon^\times$	multiplication neural network	Lemma 7.3
$\Phi_{n,\varepsilon}^\times$	multiplication of n numbers neural network	Proposition 7.4
$\Phi_2 \circ \Phi_1$	composition of neural networks	Lemma 5.2
$\Phi_2 \bullet \Phi_1$	sparse composition of neural networks	Lemma 5.2
$(\Phi_1, \ldots, \Phi_m)$	parallelization of neural networks	(5.1.1)
$\text{Pieces}(f, \Omega)$	number of pieces of f on Ω	Definition 6.1
$\mathcal{PN}(\mathcal{A}, B)$	parameter set of neural networks with architecture $\mathcal{A}$ and all weights bounded in modulus by B	Definition 12.1
$\mathbb{Q}$	rational numbers	
$\mathbb{R}$	real numbers	

Continued on next page

Symbol	Description	Reference
$\mathbb{R}_-$	non-positive real numbers	
$\mathbb{R}_+$	non-negative real numbers	
R_σ	Realization map	Definition 12.1
R^*	Bayes risk	(14.1.1)
$\mathcal{R}(h)$	risk of hypothesis h	Definition 14.2
$\widehat{\mathcal{R}}_S(h)$	empirical risk of h for sample S	(1.2.3), Definition 14.4
$\mathcal{S}_n$	cardinal B-spline	Definition 4.1
$\mathcal{S}^d_{\ell,\boldsymbol{t},n}$	multivariate cardinal B-spline	Definition 4.2
$\lvert S\rvert$	cardinality of an arbitrary set S, or Lebesgue measure of $S \subseteq \mathbb{R}^d$	
$\mathring{S}$	interior of a set S	
$\overline{S}$	closure of a set S	
∂S	boundary of a set S	
S^c	complement of a set S	
σ	general activation function	
σ_a	parametric ReLU activation function	Section 2.3
σ_{ReLU}	ReLU activation function	Section 2.3
sign	sign function	
size(Φ)	number of free network parameters in Φ	Definition 2.4
span(S)	linear hull or span of S	
$\mathcal{T}$	triangulation	Definition 5.13
$\mathbb{V}[X]$	variance of random variable X	Section A.2.2
VCdim($\mathcal{H}$)	VC dimension of a set of functions $\mathcal{H}$	Definition 14.16
$\mathcal{W}$	distribution of weight intialization	Section 11.5.1
$\boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(\ell)}$	weights and biases in layer ℓ of a neural network	Definition 2.1
width(Φ)	width of Φ	Definition 2.1
$\boldsymbol{x}^{(\ell)}$	output of ℓ-th layer of a neural network	Definition 2.1
$\bar{\boldsymbol{x}}^{(\ell)}$	preactivations	(10.3.3)
X'	dual space to a normed space X	Definition B.7

Chapter 1

Introduction

1.1 Mathematics of deep learning

In 2012, a deep learning architecture revolutionized the field of computer vision by achieving unprecedented performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [121]. The deep learning architecture, known as AlexNet, significantly outperformed all competing technologies. A few years later, in March 2016, a deep learning-based architecture called AlphaGo defeated the best Go player at the time, Lee Sedol, in a five-game match [214]. Go is a highly complex board game with a vast number of possible moves, making it a challenging problem for artificial intelligence. Because of this complexity, many researchers believed that defeating a top human Go player was a feat that would only be achieved decades later.

These breakthroughs, along with many others including DeepMind's AlphaFold [110], which revolutionized protein structure prediction in 2020, the unprecedented language capabilities of large language models like GPT-3 (and later versions) [234, 28], and the emergence of generative AI models like Stable Diffusion, Midjourney, and DALL-E, have sparked interest among scientists across (almost) all disciplines. Likewise, while mathematical research on neural networks has a long history, these groundbreaking developments revived interest in the theoretical underpinnings of deep learning among mathematicians. However, initially, there was a clear consensus in the mathematics community: *We do not understand why this technology works so well! In fact, there are many mathematical reasons that, at least superficially, should prevent the observed success.*

Over the past decade the field has matured, and mathematicians have gained a more profound understanding of deep learning, although many open questions remain. Recent years have brought various new explanations and insights into the inner workings of deep learning models. Before discussing these in detail in the following chapters, we first give a high-level introduction to deep learning, with a focus on the supervised learning framework, which is the central theme of this book.

1.2 High-level overview of deep learning

Deep learning refers to the application of deep neural networks trained by gradient-based methods, to identify unknown input-output relationships. This approach has three key ingredients: *deep neural networks, gradient-based training, and prediction.* We now explain each of these ingredients separately.

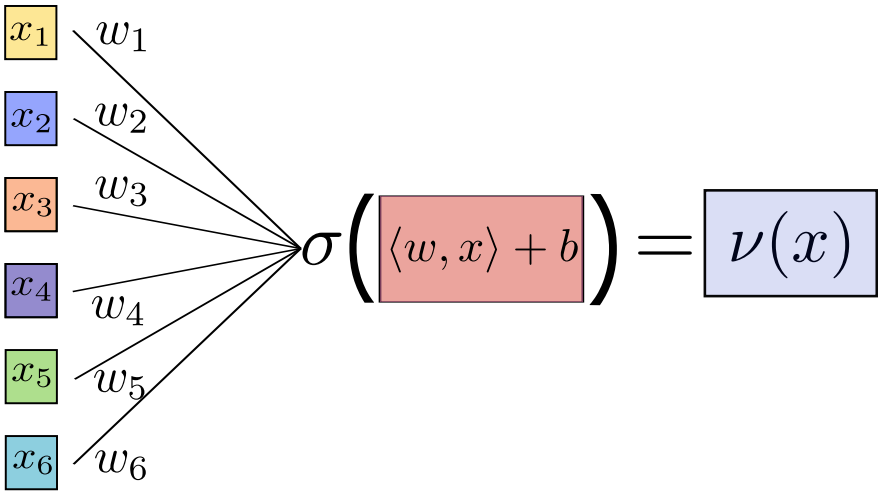


Figure 1.1: Illustration of a single neuron ν. The neuron receives six inputs $(x_1, \dots, x_6) = \boldsymbol{x}$, computes their weighted sum $\sum_{j=1}^{6} x_j w_j$, adds a bias b, and finally applies the activation function σ to produce the output $\nu(x)$.

Deep Neural Networks Deep neural networks are formed by a combination of neurons. A **neuron** is a function of the form

$$\mathbb{R}^d \ni \boldsymbol{x} \mapsto \nu(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b), \tag{1.2.1}$$

where $\boldsymbol{w} \in \mathbb{R}^d$ is a **weight vector**, $b \in \mathbb{R}$ is called **bias**, and the function σ is referred to as an **activation function**. This concept is due to McCulloch and Pitts [142] and is a mathematical model for biological neurons. If we consider σ to be the Heaviside function, i.e., $\sigma = \mathbb{1}_{\mathbb{R}_+}$ with $\mathbb{R}_+ := [0, \infty)$, then the neuron "fires" if the weighted sum of the inputs $\boldsymbol{x}$ surpasses the threshold $-b$. We depict a neuron in Figure 1.1. Note that if we fix d and σ, then the set of neurons can be naturally parameterized by the $d + 1$ real values $w_1, \dots, w_d, b \in \mathbb{R}$.

Neural networks are functions formed by connecting neurons, where the output of one neuron becomes the input to another. One simple but very common type of neural network is the so-called feedforward neural network. This structure distinguishes itself by having the neurons grouped in layers, and the inputs to neurons in the $(\ell + 1)$-st layer are exclusively neurons from the ℓ-th layer.

We start by defining a **shallow feedforward neural network** as an affine transformation applied to the output of a set of neurons that share the same input and the same activation function. Here, an **affine transformation** is a map $T : \mathbb{R}^p \to \mathbb{R}^q$ such that $T(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ for some $\boldsymbol{W} \in \mathbb{R}^{q \times p}$, $\boldsymbol{b} \in \mathbb{R}^q$ where $p, q \in \mathbb{N}$.

Formally, a shallow feedforward neural network is, therefore, a map Φ of the form

$$\mathbb{R}^d \ni \boldsymbol{x} \mapsto \Phi(\boldsymbol{x}) = T_1 \circ \sigma \circ T_0(\boldsymbol{x})$$

where T_0, T_1 are affine transformations and the application of σ is understood to be in each component of $T_1(\boldsymbol{x})$. A visualization of a shallow neural network is given in Figure 1.2.

A **deep feedforward neural network** is constructed by compositions of shallow neural networks. This yields a map of the type

$$\mathbb{R}^d \ni \boldsymbol{x} \mapsto \Phi(\boldsymbol{x}) = T_{L+1} \circ \sigma \circ \cdots \circ T_1 \circ \sigma \circ T_0(\boldsymbol{x}),$$

where $L \in \mathbb{N}$ and $(T_j)_{j=0}^{L+1}$ are affine transformations. The number of compositions L is referred to as the **number of layers** of the deep neural network. Similar to a single neuron, (deep) neural networks can be viewed as a parameterized function class, with the **parameters** being the entries of the matrices and vectors determining the affine transformations $(T_j)_{j=0}^{L+1}$.

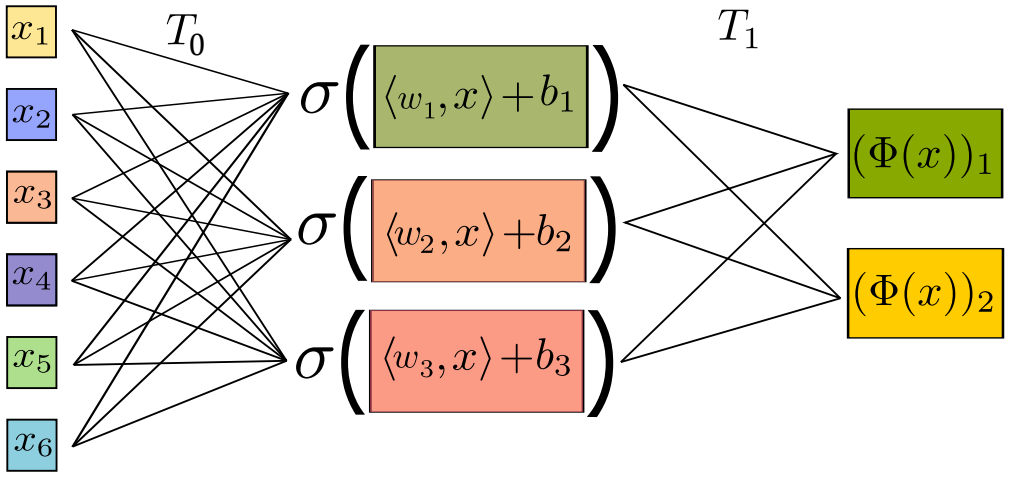


Figure 1.2: Illustration of a shallow neural network. The affine transformation T_0 is of the form $(x_1, \dots, x_6) = \boldsymbol{x} \mapsto \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$, where the rows of $\boldsymbol{W}$ are the weight vectors $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{w}_3$ for each respective neuron.

Gradient-based training After defining the structure or **architecture** of the neural network, e.g., the activation function and the number of layers, the second step of deep learning consists of determining optimal values for its parameters. This optimization is carried out by minimizing an objective function. In **supervised learning**, which will be our focus, this objective depends on a collection of input-output pairs known as a **sample**. Concretely, let $S = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^m$ be a sample, where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the inputs and $\boldsymbol{y}_i \in \mathbb{R}^k$ the corresponding outputs with d, $k \in \mathbb{N}$. Our goal is to find a deep neural network Φ such that

$$\Phi(\boldsymbol{x}_i) \approx \boldsymbol{y}_i \qquad \text{for all } i = 1, \dots, m \tag{1.2.2}$$

in a meaningful sense. For example, we could interpret "$\approx$" to mean closeness with respect to the Euclidean norm, or more generally, that $\mathcal{L}(\Phi(\boldsymbol{x}_i), \boldsymbol{y}_i)$ is small for a function $\mathcal{L}$ measuring the dissimilarity between its inputs. Such a function $\mathcal{L}$ is called a **loss function**. A standard way of achieving (1.2.2) is by minimizing the so-called **empirical risk of Φ with respect to the sample** S defined as

$$\widehat{\mathcal{R}}_S(\Phi) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\Phi(\boldsymbol{x}_i), \boldsymbol{y}_i). \tag{1.2.3}$$

If $\mathcal{L}$ is differentiable, and for all $\boldsymbol{x}_i$ the output $\Phi(\boldsymbol{x}_i)$ depends differentiably on the parameters of the neural network, then the gradient of the empirical risk $\widehat{\mathcal{R}}_S(\Phi)$ with respect to the parameters is well-defined. This gradient can be efficiently computed using a technique called **backpropagation**. This allows to minimize (1.2.3) by optimization algorithms such as (stochastic) gradient

descent. They produce a sequence of neural networks parameters, and corresponding neural network functions $\Phi_1, \Phi_2, \ldots$, for which the empirical risk is expected to decrease. Figure 1.3 illustrates a possible behavior of this sequence.

Prediction The final part of deep learning concerns the question of whether we have actually learned something by the procedure above. Suppose that our optimization routine has either converged or has been terminated, yielding a neural network Φ_*. While the optimization aimed to minimize the empirical risk on the training sample S, our ultimate interest is not in how well Φ_* performs on S. Rather, we are interested in its performance on new, unseen data points $(\boldsymbol{x}_{\text{new}}, \boldsymbol{y}_{\text{new}})$. To make meaningful statements about this performance, we need to assume a relationship between the training sample S and other data points.

The standard approach is to assume existence of a **data distribution** $\mathcal{D}$ on the input-output space—in our case, this is $\mathbb{R}^d \times \mathbb{R}^k$—such that both the elements of S and all other considered data points are drawn from this distribution. In other words, we treat S as an i.i.d. draw from $\mathcal{D}$, and $(\boldsymbol{x}_{\text{new}}, \boldsymbol{y}_{\text{new}})$ also sampled independently from $\mathcal{D}$. If we want Φ_* to perform well on average, then this amounts to controlling the following expression

$$\mathcal{R}(\Phi_*) = \mathbb{E}_{(\boldsymbol{x}_{\text{new}}, \boldsymbol{y}_{\text{new}}) \sim \mathcal{D}}[\mathcal{L}(\Phi_*(\boldsymbol{x}_{\text{new}}), \boldsymbol{y}_{\text{new}})], \tag{1.2.4}$$

which is called the **risk** of Φ_*. If the risk is not much larger than the empirical risk, then we say that the neural network Φ_* has a small **generalization error**. On the other hand, if the risk is much larger than the empirical risk, then we say that Φ_* **overfits** the training data, meaning that Φ_* has memorized the training samples, but does not generalize well to new data.

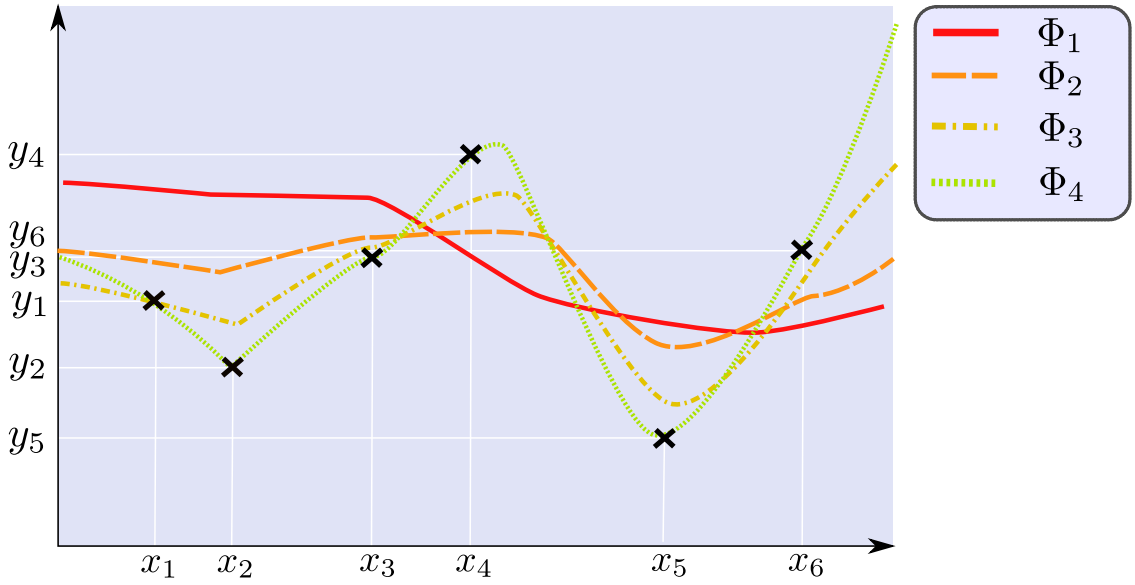


Figure 1.3: A sequence of one dimensional neural networks $\Phi_1, \ldots, \Phi_4$ that successively minimizes the empirical risk for the sample $S = (x_i, y_i)_{i=1}^6$.

1.3 Why does it work?

It is natural to wonder why the deep learning pipeline, as outlined in the previous subsection, ultimately succeeds in learning, i.e., achieving a small risk. Is it true that for a given sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^m$ there exist a neural network such that $\Phi(\boldsymbol{x}_i) \approx \boldsymbol{y}_i$ for all $i = 1, \dots m$? Does the optimization routine produce a meaningful result? Can we control the risk, knowing only that the empirical risk is small?

While most of these questions can be answered affirmatively under certain assumptions, these assumptions often do not apply to deep learning in practice. We next explore some potential explanations and show that they lead to even more questions.

Approximation A fundamental result in the study of neural networks is the so-called universal approximation theorem, which will be discussed in Chapter 3. This result states that every continuous function on a compact domain can be approximated arbitrarily well (in a uniform sense) by a shallow neural network.

This result, however, does not answer questions that are more specific to deep learning, such as the question of efficiency. For example, if we aim for computational efficiency, then we might be interested in the smallest neural network that fits the data. This raises the question: *What is the role of the architecture for the expressive capabilities of neural networks?* Furthermore, if we consider reducing the empirical risk an approximation problem, we are confronted with one of the main issues of approximation theory, which is the *curse of dimensionality*. Function approximation in high dimensions is notoriously difficult and gets exponentially harder with increasing dimension. In practice, many successful deep learning architectures operate in this high-dimensional regime. *Why do these neural networks not seem to suffer from the curse of dimensionality*?

Optimization While gradient descent can sometimes be proven to converge to a global minimum as we will discuss in Chapter 10, this typically requires the objective function to be at least convex. However, there is no reason to believe that for example the empirical risk is a convex function of the network parameters. In fact, due to the repeatedly occurring compositions with the nonlinear activation function in the network, the empirical risk is typically *highly non-linear and not convex*. Therefore, there is generally no guarantee that the optimization routine will converge to a global minimum, and it may get stuck in a local (and non-global) minimum or a saddle point. *Why is the output of the optimization nonetheless often meaningful in practice?*

Generalization In traditional statistical learning theory, which we will review in Chapter 14, the extent to which the risk exceeds the empirical risk, can be bounded a priori; such bounds are often expressed in terms of a notion of complexity of the set of admissible functions (the class of neural networks) divided by the number of training samples. For the class of neural networks of a fixed architecture, the complexity roughly amounts to the number of neural network parameters. In practice, typically neural networks with *more* parameters than training samples are used. This is dubbed the *overparameterized regime*. In this regime, the classical estimates described above are void.

Why is it that, nonetheless, *deep overparameterized architectures are capable of making accurate predictions* on unseen data? Furthermore, while deep architectures often generalize well, they sometimes fail spectacularly on specific, carefully crafted examples. In image classification tasks,

these examples may differ only slightly from correctly classified images in a way that is not perceptible to the human eye. Such examples are known as *adversarial examples*, and their existence poses a great challenge for applications of deep learning.

1.4 Outline and philosophy

This book addresses the questions raised in the previous section, providing answers that are mathematically rigorous and accessible. Our focus will be on provable statements, presented in a manner that prioritizes simplicity and clarity over generality. We will sometimes illustrate key ideas only in special cases, or under strong assumptions, both to avoid an overly technical exposition, and because definitive answers are often not yet available. In the following, we summarize the content of each chapter and highlight parts pertaining to the questions stated in the previous section.

Chapter 2: Feedforward neural networks. In this chapter, we introduce the main object of study of this book—the feedforward neural network.

Chapter 3: Universal approximation. We present the classical view of function approximation by neural networks, and give two instances of so-called universal approximation results. Such statements describe the ability of neural networks to approximate every function of a given class to arbitrary accuracy, given that the network size is sufficiently large. The first result, which holds under very broad assumptions on the activation function, is on uniform approximation of continuous functions on compact domains. The second result shows that for a very specific activation function, the network size can be chosen independent of the desired accuracy, highlighting that universal approximation needs to be interpreted with caution.

Chapter 4: Splines. Going beyond universal approximation, this chapter starts to explore approximation rates of neural networks. Specifically, we examine how well certain functions can be approximated relative to the number of parameters in the network. For so-called sigmoidal activation functions, we establish a link between neural-network- and spline-approximation. This reveals, that smoother functions require fewer network parameters. However, achieving this increased efficiency necessitates the use of deeper neural networks. This observation offers a first glimpse into the *importance of depth in deep learning*.

Chapter 5: ReLU neural networks. This chapter focuses on one of the most popular activation functions in practice—the ReLU. We prove that the class of ReLU networks is equal to the set of continuous piecewise linear functions, thus providing a theoretical foundation for their expressive power. Furthermore, given a continuous piecewise linear function, we investigate the necessary width and depth of a ReLU network to represent it. Finally, we leverage approximation theory for piecewise linear functions to derive convergence rates for approximating Hölder continuous functions.

Chapter 6: Affine pieces for ReLU neural networks. Having gained some intuition about ReLU neural networks, in this chapter, we adress some potential limitations. We analyze ReLU neural networks by counting the number of affine regions that they generate. The key insight of this chapter is that deep neural networks can generate exponentially more regions than shallow ones. This observation provides *further evidence for the potential advantages of depth* in neural network architectures.

Chapter 7: Deep ReLU neural networks. Having identified the ability of deep ReLU neural networks to generate a large number of affine regions, we investigate whether this translates into an actual advantage in function approximation. Indeed, for approximating smooth functions,

we prove substantially better approximation rates than we obtained for shallow neural networks. This adds again to our *understanding of depth and its connections to expressive power* of neural network architectures.

Chapter 8: High-dimensional approximation. The convergence rates established in the previous chapters deteriorate significantly in high-dimensional settings. This chapter examines three scenarios under which neural networks can provably *overcome the curse of dimensionality*.

Chapter 9: Interpolation. In this chapter we shift our perspective from approximation to exact interpolation of the training data. We analyze conditions under which exact interpolation is possible, and discuss the implications for empirical risk minimization. Furthermore, we present a constructive proof showing that ReLU networks can express an optimal interpolant of the data (in a specific sense).

Chapter 10: Training of neural networks. We start to examine the training process of deep learning. First, we study the fundamentals of (stochastic) gradient descent and convex optimization. Then, we discuss how the backpropagation algorithm can be used to implement these optimization algorithms for training neural networks. Finally, we examine accelerated methods and highlight the key principles behind popular and more advanced training algorithms such as Adam.

Chapter 11: Wide neural networks and the neural tangent kernel. This chapter introduces the neural tangent kernel as a tool for analyzing the training behavior of neural networks. We begin by revisiting linear and kernel regression for the approximation of functions based on data. Afterwards, we demonstrate in an abstract setting that under certain assumptions, the training dynamics of gradient descent for neural networks resemble those of kernel regression, converging to a global minimum. Using standard intialization schemes, we then show that the assumptions for such a statement to hold are satisfied with high probability, if the network is sufficiently wide (overparameterized). This analysis provides insights into why, under certain conditions, we can train neural networks *without getting stuck in (bad) local minima*, despite the non-convexity of the objective function. Additionally, we discuss a well-known link between neural networks and Gaussian processes, giving some indication why overparameterized networks *do not necessarily overfit* in practice.

Chapter 12: Loss landscape analysis. In this chapter, we present an alternative view on the optimization problem, by analyzing the loss landscape—the empirical risk as a function of the neural network parameters. We give theoretical arguments showing that increasing overparameterization leads to greater connectivity between the valleys and basins of the loss landscape. Consequently, overparameterized architectures make it easier to reach a region where all minima are global minima. Additionally, we observe that most stationary points associated with non-global minima are saddle points. This sheds further light on the empirically observed fact that deep architectures can often be optimized *without getting stuck in non-global minima*.

Chapter 13: Shape of neural network spaces. While Chapters 11 and 12 highlight potential reasons for the success of neural network training, in this chapter, we show that the set of neural networks of a fixed architecture has some undesirable properties from an optimization perspective. Specifically, we show that this set is typically non-convex. Moreover, in general it does not possess the best-approximation property, meaning that there might not exist a neural network within the set yielding the best approximation for a given function.

Chapter 14 : Generalization properties of deep neural networks. To understand why deep neural networks successfully generalize to unseen data points (outside of the training set), we study classical statistical learning theory, with a focus on neural network functions as the

hypothesis class. We then show how to establish generalization bounds for deep learning, providing theoretical insights into the *performance on unseen data*.

Chapter 15: Generalization in the overparameterized regime. The generalization bounds of the previous chapter are not meaningful when the number of parameters of a neural network surpasses the number of training samples. However, this overparameterized regime is where many successful network architectures operate. To gain a deeper understanding of generalization in this regime, we describe the phenomenon of double descent and present a potential explanation. This addresses the question of why deep neural networks *perform well despite being highly overparameterized*.

Chapter 16: Robustness and adversarial examples. In the final chapter, we explore the existence of adversarial examples—inputs designed to deceive neural networks. We provide some *theoretical explanations of why adversarial examples arise*, and discuss potential strategies to prevent them.

1.5 Material not covered in this book

This book studies some central topics of deep learning but leaves out even more. Interesting questions associated with the field that were omitted, as well as some pointers to related works are listed below:

Advanced architectures: The (deep) feedforward neural network is far from the only type of neural network. In practice, architectures must be adapted to the type of data. For example, images exhibit strong spatial dependencies in the sense that adjacent pixels often have similar values. Convolutional neural networks [128] are particularly well suited for this type of input, as they employ convolutional filters that aggregate information from neighboring pixels, thus capturing the data structure better than a fully connected feedforward network. Similarly, graph neural networks [27] are a natural choice for graph-based data. For sequential data, such as natural language, architectures with some form of memory component are used, including Long Short-Term Memory (LSTM) networks [93] and attention-based architectures like transformers [234].

Interpretability/Explainability and Fairness: The use of deep neural networks in critical decision-making processes, such as allocating scarce resources (e.g., organ transplants in medicine, financial credit approval, hiring decisions) or engineering (e.g., optimizing bridge structures, autonomous vehicle navigation, predictive maintenance), necessitates an understanding of their decision-making process. This is crucial for both practical and ethical reasons.

Practically, understanding how a model arrives at a decision can help us improve its performance and mitigate problems. It allows us to ensure that the model performs according to our intentions and does not produce undesirable outcomes. For example, in bridge design, understanding why a model suggests or rejects a particular configuration can help engineers identify potential vulnerabilities, ultimately leading to safer and more efficient designs. Ethically, transparent decision-making is crucial, especially when the outcomes have significant consequences for individuals or society; biases present in the data or model design can lead to discriminatory outcomes, making explainability essential.

However, explaining the predictions of deep neural networks is not straightforward. Despite knowledge of the network weights and biases, the repeated and complex interplay of linear transformations and non-linear activation functions often renders these models black boxes. A comprehensive overview of various techniques for interpretability, not only for deep neural networks, can

be found in [149]. Regarding the topic of fairness, we refer for instance to [55, 8].

Unsupervised and Reinforcement Learning: While this book focuses on supervised learning, where each data point x_i has a label y_i, there is a vast field of machine learning called unsupervised learning, where labels are absent. Classical unsupervised learning problems include clustering and dimensionality reduction [212, Chapters 22/23].

A popular area in deep learning, where no labels are used, is physics-informed neural networks [187]. Here, a neural network is trained to satisfy a partial differential equation (PDE), with the loss function quantifying the deviation from this PDE.

Finally, reinforcement learning is a technique where an agent can interact with an environment and receives feedback based on its actions. The actions are guided by a so-called policy, which is to be learned, [148, Chapter 17]. In deep reinforcement learning, this policy is modeled by a deep neural network. Reinforcement learning is the basis of the aforementioned AlphaGo.

Implementation: While this book focuses on provable theoretical results, the field of deep learning is strongly driven by applications, and a thorough understanding of deep learning cannot be achieved without practical experience. For this, there exist numerous resources with excellent explanations. We recommend [67, 38, 182] as well as the countless online tutorials that are just a Google (or alternative) search away.

Many more: The field is evolving rapidly, and new ideas are constantly being generated and tested. This book cannot give a complete overview. However, we hope that it provides the reader with a solid foundation in the fundamental knowledge and principles to quickly grasp and understand new developments in the field.

Bibliography and further reading

Throughout this book, we will end each chapter with a short overview of related work and the references used in the chapter.

In this introductory chapter, we highlight several other recent textbooks and works on deep learning. For a historical survey on neural networks see [202] and also [127]. For general textbooks on neural networks and deep learning, we refer to [84, 72, 182] for more recent monographs. A more mathematical introduction to the topic is given, for example, in [3, 107, 29]. For the implementation of neural networks we refer for example to [67, 38].

Chapter 2

Feedforward neural networks

Feedforward neural networks, henceforth simply referred to as neural networks (NNs), constitute the central object of study of this book. In this chapter, we provide a formal definition of neural networks, discuss the *size* of a neural network, and give a brief overview of common activation functions.

2.1 Formal definition

We previously defined a single neuron ν in (1.2.1) and Figure 1.1. A neural network is constructed by connecting multiple neurons. Let us now make precise this connection procedure.

Definition 2.1. Let $L \in \mathbb{N}$, $d_0, \dots, d_{L+1} \in \mathbb{N}$, and let $\sigma \colon \mathbb{R} \to \mathbb{R}$. A function $\Phi \colon \mathbb{R}^{d_0} \to \mathbb{R}^{d_{L+1}}$ is called a **neural network** if there exist matrices $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ and vectors $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$, $\ell = 0, \dots, L$, such that with

$$\boldsymbol{x}^{(0)} := \boldsymbol{x} \tag{2.1.1a}$$

$$\boldsymbol{x}^{(\ell)} := \sigma(\boldsymbol{W}^{(\ell-1)}\boldsymbol{x}^{(\ell-1)} + \boldsymbol{b}^{(\ell-1)}) \qquad \text{for } \ell \in \{1, \dots, L\} \tag{2.1.1b}$$

$$\boldsymbol{x}^{(L+1)} := \boldsymbol{W}^{(L)}\boldsymbol{x}^{(L)} + \boldsymbol{b}^{(L)} \tag{2.1.1c}$$

holds

$$\Phi(\boldsymbol{x}) = \boldsymbol{x}^{(L+1)} \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^{d_0}.$$

We call L the **depth**, $d_{\max} = \max_{\ell=1,\dots,L} d_\ell$ the **width**, σ the **activation function**, and $(\sigma; d_0, \dots, d_{L+1})$ the **architecture** of the neural network Φ. Moreover, $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ are the **weight matrices** and $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ the **bias vectors** of Φ for $\ell = 0, \dots L$.

Remark 2.2. Typically, there exist different choices of architectures, weights, and biases yielding the same function $\Phi : \mathbb{R}^{d_0} \to \mathbb{R}^{d_{L+1}}$. For this reason we cannot associate a unique meaning to these notions solely based on the *function* realized by Φ. In the following, when we refer to the properties

of a neural network Φ, it is always understood to mean that there exists at least one construction as in Definition 2.1, which realizes the function Φ and uses parameters that satisfy those properties.

The architecture of a neural network is often depicted as a connected graph, as illustrated in Figure 2.1. The **nodes** in such graphs represent (the output of) the neurons. They are arranged in **layers**, with $\boldsymbol{x}^{(\ell)}$ in Definition 2.1 corresponding to the neurons in layer ℓ. We also refer to $\boldsymbol{x}^{(0)}$ in (2.1.1a) as the **input layer** and to $\boldsymbol{x}^{(L+1)}$ in (2.1.1c) as the **output layer**. All layers in between are referred to as the **hidden layers** and their output is given by (2.1.1b). The number of hidden layers corresponds to the depth. For the correct interpretation of such graphs, we note that by our conventions in Definition 2.1, the activation function is applied after each affine transformation, except in the final layer.

Neural networks of depth one are called **shallow**, if the depth is larger than one they are called **deep**. The notion of deep neural networks is not used entirely consistently in the literature, and some authors use the word deep only in case the depth is much larger than one, where the precise meaning of "much larger" depends on the application.

Throughout, we only consider neural networks in the sense of Definition 2.1. We emphasize however, that this is just one (simple but very common) type of neural network. Many adjustments to this construction are possible and also widely used. For example:

- We may use **different activation functions** σ_ℓ in each layer ℓ or we may even use a different activation function for each node.

- **Residual** neural networks allow "skip connections". This means that information is allowed to skip layers in the sense that the nodes in layer ℓ may have $\boldsymbol{x}^{(0)}, \dots, \boldsymbol{x}^{(\ell-1)}$ as their input (and not just $\boldsymbol{x}^{(\ell-1)}$), cf. (2.1.1).

- In contrast to feedforward neural networks, **recurrent** neural networks allow information to flow backward, in the sense that $\boldsymbol{x}^{(\ell-1)}, \dots, \boldsymbol{x}^{(L+1)}$ may serve as input for the nodes in layer ℓ (and not just $\boldsymbol{x}^{(\ell-1)}$). This creates loops in the flow of information, and one has to introduce a time index $t \in \mathbb{N}$, as the output of a node in time step t might be different from the output in time step $t+1$.

Let us clarify some further common terminology used in the context of neural networks:

- **parameters**: The parameters of a neural network refer to the set of all entries of the weight matrices and bias vectors. These are often collected in a single vector

$$\boldsymbol{w} = ((\boldsymbol{W}^{(0)}, \boldsymbol{b}^{(0)}), \dots, (\boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)})). \tag{2.1.2}$$

  These parameters are adjustable and are learned during the training process, determining the specific function realized by the network.

- **hyperparameters**: Hyperparameters are settings that define the network's architecture (and training process), but are not directly learned during training. Examples include the depth, the number of neurons in each layer, and the choice of activation function. They are typically set before training begins.

- **weights**: The term "weights" is often used broadly to refer to *all* parameters of a neural network, including both the weight matrices and bias vectors.

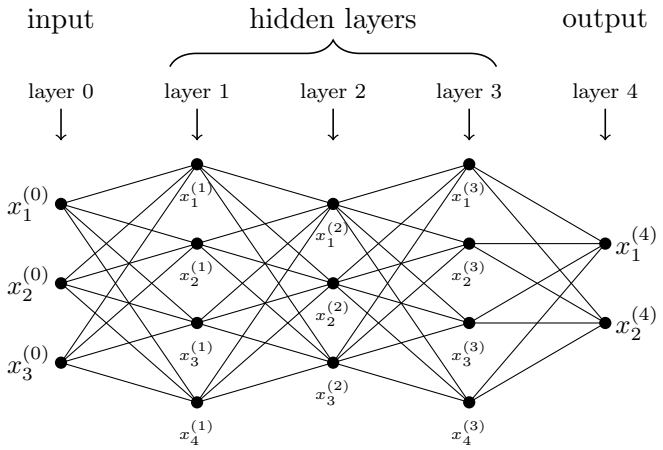


Figure 2.1: Sketch of a neural network with three hidden layers, and $d_0 = 3$, $d_1 = 4$, $d_2 = 3$, $d_3 = 4$, $d_4 = 2$. The neural network has depth three and width four.

- **model**: For a fixed architecture, every choice of network parameters $\boldsymbol{w}$ as in (2.1.2) defines a specific function $\boldsymbol{x} \mapsto \Phi_{\boldsymbol{w}}(\boldsymbol{x})$. In deep learning this function is often referred to as a model. More generally, "model" can be used to describe any function parameterization by a set of parameters $\boldsymbol{w} \in \mathbb{R}^n$, $n \in \mathbb{N}$.

2.1.1 Basic operations on neural networks

There are various ways how neural networks can be combined with one another. The next proposition addresses this for linear combinations, compositions, and parallelization. The formal proof, which is a good exercise to familiarize oneself with neural networks, is left as Exercise 2.5.

Proposition 2.3. *For two neural networks Φ_1, Φ_2, with architectures*

$$(\sigma; d_0^1, d_1^1, \dots, d_{L_1+1}^1) \quad \text{and} \quad (\sigma; d_0^2, d_1^2, \dots, d_{L_2+1}^2)$$

respectively, it holds that

(i) for all $\alpha \in \mathbb{R}$ exists a neural network Φ_α with architecture $(\sigma; d_0^1, d_1^1, \dots, d_{L_1+1}^1)$ such that

$$\Phi_\alpha(\boldsymbol{x}) = \alpha \Phi_1(\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^{d_0^1},$$

(ii) if $d_0^1 = d_0^2 =: d_0$ and $L_1 = L_2 =: L$, then there exists a neural network Φ_{parallel} with architecture $(\sigma; d_0, d_1^1 + d_1^2, \dots, d_{L+1}^1 + d_{L+1}^2)$ such that

$$\Phi_{\text{parallel}}(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}), \Phi_2(\boldsymbol{x})) \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^{d_0},$$

(iii) if $d_0^1 = d_0^2 =: d_0$, $L_1 = L_2 =: L$, and $d_{L+1}^1 = d_{L+1}^2 =: d_{L+1}$, then there exists a neural network Φ_{sum} with architecture $(\sigma; d_0, d_1^1 + d_1^2, \dots, d_L^1 + d_L^2, d_{L+1})$ such that

$$\Phi_{\text{sum}}(\boldsymbol{x}) = \Phi_1(\boldsymbol{x}) + \Phi_2(\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^{d_0},$$

(iv) if $d^1_{L_1+1} = d^2_0$, *then there exists a neural network* Φ_{comp} *with architecture* $(\sigma; d^1_0, d^1_1, \dots, d^1_{L_1}, d^2_1, \dots, d^2_{L_2+1})$ *such that*

$$\Phi_{\text{comp}}(\boldsymbol{x}) = \Phi_2 \circ \Phi_1(\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^{d^1_0}.$$

2.2 Notion of size

Neural networks provide a framework to parametrize functions. Ultimately, our goal is to find a neural network that fits some underlying input-output relation. As mentioned above, the architecture (depth, width and activation function) is typically chosen apriori and considered fixed. During training of the neural network, its parameters (weights and biases) are suitably adapted by some algorithm. Depending on the application, on top of the stated architecture choices, further restrictions on the weights and biases can be desirable. For example, the following two appear frequently:

- **weight sharing**: This is a technique where specific entries of the weight matrices (or bias vectors) are constrained to be equal. Formally, this means imposing conditions of the form $W^{(i)}_{k,l} = W^{(j)}_{s,t}$, i.e. the entry (k,l) of the ith weight matrix is equal to the entry at position (s,t) of weight matrix j. We denote this assumption by $(i,k,l) \sim (j,s,t)$, paying tribute to the trivial fact that "$\sim$" is an equivalence relation. During training, shared weights are updated jointly, meaning that any change to one weight is simultaneously applied to all other weights of this class. Weight sharing can also be applied to the entries of bias vectors.
- **sparsity**: This refers to imposing a sparsity structure on the weight matrices (or bias vectors). Specifically, we apriorily set $W^{(i)}_{k,l} = 0$ for certain (k,l,i), i.e. we impose entry (k,l) of the ith weight matrix to be 0. These zero-valued entries are considered fixed, and are not adjusted during training. The condition $W^{(i)}_{k,l} = 0$ corresponds to node l of layer $i-1$ *not* serving as an input to node k in layer i. If we represent the neural network as a graph, this is indicated by not connecting the corresponding nodes. Sparsity can also be imposed on the bias vectors.

Both of these restrictions decrease the number of learnable parameters in the neural network. The number of parameters can be seen as a measure of the complexity of the represented function class. For this reason, we introduce size(Φ) as a notion for the number of learnable parameters. Formally (with $|S|$ denoting the cardinality of a set S):

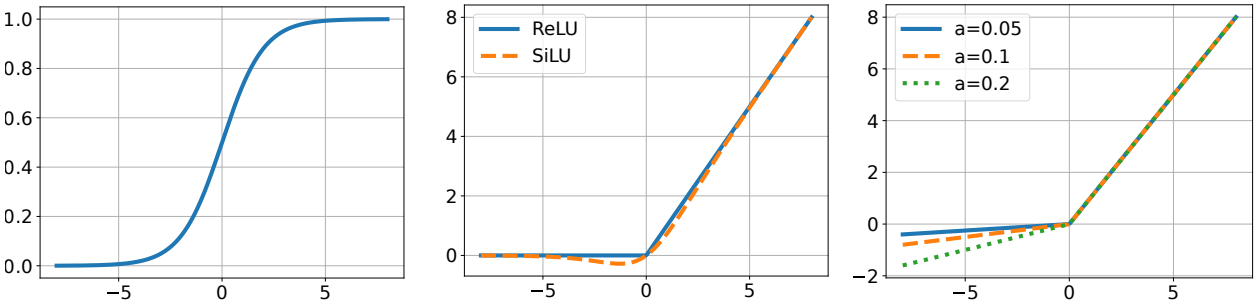
Definition 2.4. Let Φ be as in Definition 2.1. Then the **size** of Φ is

$$\text{size}(\Phi) := \Big| \Big(\{(i,k,l) \,|\, W^{(i)}_{k,l} \neq 0\} \cup \{(i,k) \,|\, b^{(i)}_k \neq 0\} \Big) \,/\, \sim \Big| . \tag{2.2.1}$$

2.3 Activation functions

Activation functions are a crucial part of neural networks, as they introduce nonlinearity into the model. If an affine activation function were used, the resulting neural network function would also be affine and hence very restricted in what it can represent.

The choice of activation function can have a significant impact on the performance, but there does not seem to be a universally optimal one. We next discuss a few important activation functions and highlight some common issues associated with them.



(a) Sigmoid (b) ReLU and SiLU (c) Leaky ReLU

Figure 2.2: Different activation functions.

Sigmoid: The sigmoid activation function is given by

$$\sigma_{\text{sig}}(x) = \frac{1}{1+e^{-x}} \qquad \text{for } x \in \mathbb{R},$$

and depicted in Figure 2.2 (a). Its output ranges between zero and one, making it interpretable as a probability. The sigmoid is a smooth function, which allows the application of gradient-based training.

It has the disadvantage that its derivative becomes very small if $|x| \to \infty$. This can affect learning due to the so-called vanishing gradient problem. Consider the simple neural network $\Phi_n(x) = \sigma \circ \cdots \circ \sigma(x+b)$ defined with $n \in \mathbb{N}$ compositions of σ, and where $b \in \mathbb{R}$ is a bias. Its derivative with respect to b is

$$\frac{\mathrm{d}}{\mathrm{d}b}\Phi_n(x) = \sigma'(\Phi_{n-1}(x))\frac{\mathrm{d}}{\mathrm{d}b}\Phi_{n-1}(x).$$

If $\sup_{x\in\mathbb{R}}|\sigma'(x)| \leq 1-\delta$, then by induction, $|\frac{\mathrm{d}}{\mathrm{d}b}\Phi_n(x)| \leq (1-\delta)^n$. The opposite effect happens for activation functions with derivatives uniformly larger than one. This argument shows that the derivative of $\Phi_n(x,b)$ with respect to b can become exponentially small or exponentially large when propagated through the layers. This effect, known as the *vanishing- or exploding gradient effect*, also occurs for activation functions which do not admit the uniform bounds assumed above. However, since the sigmoid activation function exhibits areas with extremely small gradients, the vanishing gradient effect can be strongly exacerbated.

ReLU (Rectified Linear Unit): The ReLU is defined as

$$\sigma_{\text{ReLU}}(x) = \max\{x, 0\} \qquad \text{for } x \in \mathbb{R},$$

and depicted in Figure 2.2 (b). It is piecewise linear, and due to its simplicity its evaluation is computationally very efficient. It is one of the most popular activation functions in practice. Since its derivative is always zero or one, it does not suffer from the vanishing gradient problem to the same extent as the sigmoid function. However, ReLU can suffer from the so-called *dead neurons* problem. Consider the neural network

$$\Phi(x) = \sigma_{\mathrm{ReLU}}(b - \sigma_{\mathrm{ReLU}}(x)) \qquad \text{for } x \in \mathbb{R}$$

depending on the bias $b \in \mathbb{R}$. If $b < 0$, then $\Phi(x) = 0$ for all $x \in \mathbb{R}$. The neuron corresponding to the second application of σ_{ReLU} thus produces a constant signal. Moreover, if $b < 0$, $\frac{\mathrm{d}}{\mathrm{d}b}\Phi(x) = 0$ for all $x \in \mathbb{R}$. As a result, every negative value of b yields a stationary point of the empirical risk. A gradient-based method will not be able to further train the parameter b. We thus refer to this neuron as a dead neuron.

SiLU (Sigmoid Linear Unit): An important difference between the ReLU and the Sigmoid is that the ReLU is not differentiable at 0. The SiLU activation function (also referred to as "swish") can be interpreted as a smooth approximation to the ReLU. It is defined as

$$\sigma_{\mathrm{SiLU}}(x) := x\sigma_{\mathrm{sig}}(x) = \frac{x}{1 + e^{-x}} \qquad \text{for } x \in \mathbb{R},$$

and is depicted in Figure 2.2 (b). There exist various other smooth activation functions that mimic the ReLU, including the Softplus $x \mapsto \log(1 + \exp(x))$, the GELU (Gaussian Error Linear Unit) $x \mapsto xF(x)$ where $F(x)$ denotes the cumulative distribution function of the standard normal distribution, and the Mish $x \mapsto x \tanh(\log(1 + \exp(x)))$.

Parametric ReLU or Leaky ReLU: This variant of the ReLU addresses the dead neuron problem. For some $a \in (0, 1)$, the parametric ReLU is defined as

$$\sigma_a(x) = \max\{x, ax\} \qquad \text{for } x \in \mathbb{R},$$

and is depicted in Figure 2.2 (c) for three different values of a. Since the output of σ does not have flat regions like the ReLU, the dying ReLU problem is mitigated. If a is not chosen too small, then there is less of a vanishing gradient problem than for the Sigmoid. In practice, the additional parameter a has to be fine-tuned depending on the application. Like the ReLU, the parametric ReLU is not differentiable at 0.

Bibliography and further reading

The concept of neural networks was first introduced by McCulloch and Pitts in [142]. Later Rosenblatt [192] introduced the perceptron, an artificial neuron with adjustable weights that forms the basis of the multilayer perceptron (a fully connected feedforward neural network). The vanishing gradient problem shortly addressed in Section 2.3 was discussed by Hochreiter in his diploma thesis [91] and later in [17, 93].

Exercises

Exercise 2.5. Prove Proposition 2.3.

Exercise 2.6. In this exercise, we show that ReLU and parametric ReLU create similar sets of neural network functions. Fix $a > 0$.

(i) Find a set of weight matrices and biases vectors, such that the associated neural network Φ_1, with the ReLU activation function σ_{ReLU} satisfies $\Phi_1(x) = \sigma_a(x)$ for all $x \in \mathbb{R}$.

(ii) Find a set of weight matrices and biases vectors, such that the associated neural network Φ_2, with the parametric ReLU activation function σ_a satisfies $\Phi_2(x) = \sigma_{\mathrm{ReLU}}(x)$ for all $x \in \mathbb{R}$.

(iii) Conclude that every ReLU neural network can be expressed as a leaky ReLU neural network and vice versa.

Exercise 2.7. Let $d \in \mathbb{N}$, and let Φ_1 be a neural network with the ReLU as activation function, input dimension d, and output dimension 1. Moreover, let Φ_2 be a neural network with the sigmoid activation function, input dimension d, and output dimension 1. Show that, if $\Phi_1 = \Phi_2$, then Φ_1 is a constant function.

Exercise 2.8. In this exercise, we show that for the sigmoid activation functions, dead-neuron-like behavior is very rare. Let Φ be a neural network with the sigmoid activation function. Assume that Φ is a constant function. Show that for every $\varepsilon > 0$ there is a non-constant neural network $\widetilde{\Phi}$ with the same architecture as Φ such that for all $\ell = 0, \dots L$,

$$\|\boldsymbol{W}^{(\ell)} - \widetilde{\boldsymbol{W}}^{(\ell)}\| \leq \varepsilon \text{ and } \|\boldsymbol{b}^{(\ell)} - \widetilde{\boldsymbol{b}}^{(\ell)}\| \leq \varepsilon$$

where $\boldsymbol{W}^{(\ell)}$, $\boldsymbol{b}^{(\ell)}$ are the weights and biases of Φ and $\widetilde{\boldsymbol{W}}^{(\ell)}$, $\widetilde{\boldsymbol{b}}^{(\ell)}$ are the biases of $\widetilde{\Phi}$.

Show that such a statement does not hold for ReLU neural networks. What about leaky ReLU?

Chapter 3

Universal approximation

After introducing neural networks in Chapter 2, it is natural to inquire about their capabilities. Specifically, we might wonder if there exist inherent limitations to the type of functions a neural network can represent. Could there be a class of functions that neural networks cannot approximate? If so, it would suggest that neural networks are specialized tools, similar to how linear regression is suited for linear relationships, but not for data with nonlinear relationships.

In this chapter, we will show that this is not the case, and neural networks are indeed a *universal* tool. More precisely, given sufficiently large and complex architectures, they can approximate almost every sensible input-output relationship. We will formalize and prove this claim in the subsequent sections.

3.1 A universal approximation theorem

To analyze what kind of functions can be approximated with neural networks, we start by considering the uniform approximation of continuous functions $f : \mathbb{R}^d \to \mathbb{R}$ on compact sets. To this end, we first introduce the notion of compact convergence.

Definition 3.1. Let $d \in \mathbb{N}$. A sequence of functions $f_n : \mathbb{R}^d \to \mathbb{R}$, $n \in \mathbb{N}$, is said to **converge compactly** to a function $f : \mathbb{R}^d \to \mathbb{R}$, if for every compact $K \subseteq \mathbb{R}^d$ it holds that $\lim_{n\to\infty} \sup_{\boldsymbol{x}\in K} |f_n(\boldsymbol{x}) - f(\boldsymbol{x})| = 0$. In this case we write $f_n \xrightarrow{\text{cc}} f$.

Throughout what follows, we always consider $C^0(\mathbb{R}^d)$ equipped with the topology of Definition 3.1 (also see Exercise 3.22), and every subset such as $C^0(D)$ with the subspace topology: for example, if $D \subseteq \mathbb{R}^d$ is bounded, then convergence in $C^0(D)$ refers to uniform convergence $\lim_{n\to\infty} \sup_{x\in D} |f_n(x) - f(x)| = 0$.

3.1.1 Universal approximators

As stated before, we want to show that deep neural networks can approximate every continuous function in the sense of Definition 3.1. We call sets of functions that satisfy this property *universal approximators*.

Definition 3.2. Let $d \in \mathbb{N}$. A set of functions $\mathcal{H}$ from $\mathbb{R}^d$ to $\mathbb{R}$ is a **universal approximator** (of $C^0(\mathbb{R}^d)$), if for every $\varepsilon > 0$, every compact $K \subseteq \mathbb{R}^d$, and every $f \in C^0(\mathbb{R}^d)$, there exists $g \in \mathcal{H}$ such that $\sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x}) - g(\boldsymbol{x})| < \varepsilon$.

For a set of (not necessarily continuous) functions $\mathcal{H}$ mapping between $\mathbb{R}^d$ and $\mathbb{R}$, we denote by $\overline{\mathcal{H}}^{\mathrm{cc}}$ its closure with respect to compact convergence.

The relationship between a universal approximator and the closure with respect to compact convergence is established in the proposition below.

Proposition 3.3. *Let $d \in \mathbb{N}$ and $\mathcal{H}$ be a set of functions from $\mathbb{R}^d$ to $\mathbb{R}$. Then, $\mathcal{H}$ is a universal approximator of $C^0(\mathbb{R}^d)$ if and only if $C^0(\mathbb{R}^d) \subseteq \overline{\mathcal{H}}^{\mathrm{cc}}$.*

Proof. Suppose that $\mathcal{H}$ is a universal approximator and fix $f \in C^0(\mathbb{R}^d)$. For $n \in \mathbb{N}$, define $K_n := [-n, n]^d \subseteq \mathbb{R}^d$. Then for every $n \in \mathbb{N}$ there exists $f_n \in \mathcal{H}$ such that $\sup_{\boldsymbol{x} \in K_n} |f_n(\boldsymbol{x}) - f(\boldsymbol{x})| < 1/n$. Since for every compact $K \subseteq \mathbb{R}^d$ there exists n_0 such that $K \subseteq K_n$ for all $n \geq n_0$, it holds $f_n \xrightarrow{\mathrm{cc}} f$. The "only if" part of the assertion is trivial. □

A key tool to show that a set is a universal approximator is the Stone-Weierstrass theorem, see for instance [196, Sec. 5.7].

Theorem 3.4 (Stone-Weierstrass)**.** *Let $d \in \mathbb{N}$, let $K \subseteq \mathbb{R}^d$ be compact, and let $\mathcal{H} \subseteq C^0(K, \mathbb{R})$ satisfy that*

(a) for all $\boldsymbol{x} \in K$ there exists $f \in \mathcal{H}$ such that $f(\boldsymbol{x}) \neq 0$,

(b) for all $\boldsymbol{x} \neq \boldsymbol{y} \in K$ there exists $f \in \mathcal{H}$ such that $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$,

(c) $\mathcal{H}$ is an algebra of functions, i.e., $\mathcal{H}$ is closed under addition, multiplication and scalar multiplication.

Then $\mathcal{H}$ is dense in $C^0(K)$.

Example 3.5 (Polynomials are dense in $C^0(\mathbb{R}^d)$)**.** For a multiindex $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ and a vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ denote $\boldsymbol{x}^{\boldsymbol{\alpha}} := \prod_{j=1}^d x_j^{\alpha_j}$. In the following, with $|\boldsymbol{\alpha}| := \sum_{j=1}^d \alpha_j$, we write

$$\mathcal{P}_n := \operatorname{span}\{\boldsymbol{x}^{\boldsymbol{\alpha}} \,|\, \boldsymbol{\alpha} \in \mathbb{N}_0^d,\ |\boldsymbol{\alpha}| \leq n\}$$

i.e., $\mathcal{P}_n$ is the space of polynomials of degree at most n (with real coefficients). It is easy to check that $\mathcal{P} := \bigcup_{n \in \mathbb{N}} \mathcal{P}_n(\mathbb{R}^d)$ satisfies the assumptions of Theorem 3.4 on every compact set $K \subseteq \mathbb{R}^d$. Thus the space of polynomials $\mathcal{P}$ is a universal approximator of $C^0(\mathbb{R}^d)$, and by Proposition 3.3, $\mathcal{P}$ is dense in $C^0(\mathbb{R}^d)$. In case we wish to emphasize the dimension of the underlying space, in the following we will also write $\mathcal{P}_n(\mathbb{R}^d)$ or $\mathcal{P}(\mathbb{R}^d)$ to denote $\mathcal{P}_n$, $\mathcal{P}$ respectively.

3.1.2 Shallow neural networks

With the necessary formalism established, we can now show that shallow neural networks of arbitrary width form a universal approximator under certain (mild) conditions on the activation function. The results in this section are based on [132], and for the proofs we follow the arguments in that paper.

We first introduce notation for the set of all functions realized by certain architectures.

Definition 3.6. Let d, m, L, $n \in \mathbb{N}$ and $\sigma\colon \mathbb{R} \to \mathbb{R}$. The set of all functions realized by neural networks with d-dimensional input, m-dimensional output, depth at most L, width at most n, and activation function σ is denoted by

$$\mathcal{N}_d^m(\sigma; L, n) := \{\Phi : \mathbb{R}^d \to \mathbb{R}^m \,|\, \Phi \text{ as in Def. 2.1, } \mathrm{depth}(\Phi) \leq L, \ \mathrm{width}(\Phi) \leq n\}.$$

Furthermore,

$$\mathcal{N}_d^m(\sigma; L) := \bigcup_{n \in \mathbb{N}} \mathcal{N}_d^m(\sigma; L, n).$$

In the sequel, we require the activation function σ to belong to the set of piecewise continuous and locally bounded functions

$$\begin{aligned} \mathcal{M} := \big\{ \sigma \in L^\infty_{\mathrm{loc}}(\mathbb{R}) \;\big|\; &\text{there exist intervals } I_1, \dots, I_M \text{ partitioning } \mathbb{R}, \\ &\text{s.t. } \sigma \in C^0(I_j) \text{ for all } j = 1, \dots, M \big\}. \end{aligned} \tag{3.1.1}$$

Here, $M \in \mathbb{N}$ is finite, and the intervals I_j are understood to have positive (possibly infinite) Lebesgue measure, i.e. I_j is e.g. not allowed to be empty or a single point. Hence, σ is a piecewise continuous function, and it has discontinuities at most finitely many points.

Example 3.7. Activation functions belonging to $\mathcal{M}$ include, in particular, all continuous non-polynomial functions, which in turn includes all practically relevant activation functions such as the ReLU, the SiLU, and the Sigmoid discussed in Section 2.3. In these cases, we can choose $M = 1$ and $I_1 = \mathbb{R}$. Discontinuous functions include for example the Heaviside function $x \mapsto \mathbb{1}_{x>0}$ (also called a "perceptron" in this context) but also $x \mapsto \mathbb{1}_{x>0} \sin(1/x)$: Both belong to $\mathcal{M}$ with $M = 2$, $I_1 = (-\infty, 0]$ and $I_2 = (0, \infty)$. We exclude for example the function $x \mapsto 1/x$, which is not locally bounded.

The rest of this subsection is dedicated to proving the following theorem that has now already been anounced repeatedly.

Theorem 3.8. *Let $d \in \mathbb{N}$ and $\sigma \in \mathcal{M}$. Then $\mathcal{N}_d^1(\sigma; 1)$ is a universal approximator of $C^0(\mathbb{R}^d)$ if and only if σ is not a polynomial.*

Remark 3.9. We will see in Exercise 3.26 and Corollary 3.18 that neural networks can also arbitrarily well approximate non-continuous functions with respect to suitable norms.

The universal approximation theorem by Leshno, Lin, Pinkus and Schocken [132]—of which Theorem 3.8 is a special case—is even formulated for a much larger set $\mathcal{M}$, which allows for activation functions that have discontinuities at a (possibly non-finite) set of Lebesgue measure zero. Instead of proving the theorem in this generality, we resort to the simpler case stated above. This allows to avoid some technicalities, but the main ideas remain the same. The proof strategy is to verify the following three claims:

(i) if $C^0(\mathbb{R}^1) \subseteq \overline{\mathcal{N}_1^1(\sigma;1)}^{\mathrm{cc}}$ then $C^0(\mathbb{R}^d) \subseteq \overline{\mathcal{N}_d^1(\sigma;1)}^{\mathrm{cc}}$,

(ii) if $\sigma \in C^\infty(\mathbb{R})$ is not a polynomial then $C^0(\mathbb{R}^1) \subseteq \overline{\mathcal{N}_1^1(\sigma;1)}^{\mathrm{cc}}$,

(iii) if $\sigma \in \mathcal{M}$ is not a polynomial then there exists $\tilde{\sigma} \in C^\infty(\mathbb{R}) \cap \overline{\mathcal{N}_1^1(\sigma;1)}^{\mathrm{cc}}$ which is not a polynomial.

Upon observing that $\tilde{\sigma} \in \overline{\mathcal{N}_1^1(\sigma;1)}^{\mathrm{cc}}$ implies $\overline{\mathcal{N}_1^1(\tilde{\sigma},1)}^{\mathrm{cc}} \subseteq \overline{\mathcal{N}_1^1(\sigma;1)}^{\mathrm{cc}}$, it is easy to see that these statements together with Proposition 3.3 establish the implication "$\Leftarrow$" asserted in Theorem 3.8. The reverse direction is straightforward to check and will be the content of Exercise 3.23.

We start with a more general version of (i) and reduce the problem to the one dimensional case.

Lemma 3.10. *Assume that $\mathcal{H}$ is a universal approximator of $C^0(\mathbb{R})$. Then for every $d \in \mathbb{N}$*

$$\operatorname{span}\{\boldsymbol{x} \mapsto g(\boldsymbol{w}\cdot\boldsymbol{x}) \,|\, \boldsymbol{w} \in \mathbb{R}^d,\ g \in \mathcal{H}\}$$

is a universal approximator of $C^0(\mathbb{R}^d)$.

Proof. For $k \in \mathbb{N}_0$, denote by $\mathbb{H}_k$ the space of all k-homogenous polynomials, that is

$$\mathbb{H}_k := \operatorname{span}\left\{\mathbb{R}^d \ni \boldsymbol{x} \mapsto \boldsymbol{x}^{\boldsymbol{\alpha}} \,\middle|\, \boldsymbol{\alpha} \in \mathbb{N}_0^d,\ |\boldsymbol{\alpha}| = k\right\}.$$

We claim that

$$\mathbb{H}_k \subseteq \overline{\operatorname{span}\{\mathbb{R}^d \ni \boldsymbol{x} \mapsto g(\boldsymbol{w}\cdot\boldsymbol{x}) \,|\, \boldsymbol{w} \in \mathbb{R}^d,\ g \in \mathcal{H}\}}^{\mathrm{cc}} =: X \tag{3.1.2}$$

for all $k \in \mathbb{N}_0$. This implies that all multivariate polynomials belong to X. An application of the Stone-Weierstrass theorem (cp. Example 3.5) and Proposition 3.3 then conclude the proof.

For every $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}_0^d$ with $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}| = k$, it holds $D^{\boldsymbol{\beta}}\boldsymbol{x}^{\boldsymbol{\alpha}} = \delta_{\boldsymbol{\beta},\boldsymbol{\alpha}}\boldsymbol{\alpha}!$, where $\boldsymbol{\alpha}! := \prod_{j=1}^d \alpha_j!$ and $\delta_{\boldsymbol{\beta},\boldsymbol{\alpha}} = 1$ if $\boldsymbol{\beta} = \boldsymbol{\alpha}$ and $\delta_{\boldsymbol{\beta},\boldsymbol{\alpha}} = 0$ otherwise. Hence, since $\{\boldsymbol{x} \mapsto \boldsymbol{x}^{\boldsymbol{\alpha}} \,|\, |\boldsymbol{\alpha}| = k\}$ is a basis of $\mathbb{H}_k$, the set $\{D^{\boldsymbol{\alpha}} \,|\, |\boldsymbol{\alpha}| = k\}$ is a basis of its topological dual $\mathbb{H}_k'$. Thus each linear functional $l \in \mathbb{H}_k'$ allows the representation $l = p(D)$ for some $p \in \mathbb{H}_k$ (here D stands for the differential).

By the multinomial formula

$$(\boldsymbol{w}\cdot\boldsymbol{x})^k = \left(\sum_{j=1}^d w_j x_j\right)^k = \sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|=k\}} \frac{k!}{\boldsymbol{\alpha}!}\boldsymbol{w}^{\boldsymbol{\alpha}}\boldsymbol{x}^{\boldsymbol{\alpha}}.$$

Therefore, we have that $(\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k) \in \mathbb{H}_k$. Moreover, for every $l = p(D) \in \mathbb{H}_k'$ and all $\boldsymbol{w} \in \mathbb{R}^d$ we have that

$$l(\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k) = k! p(\boldsymbol{w}).$$

Hence, if $l(\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k) = p(D)(\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k) = 0$ for all $\boldsymbol{w} \in \mathbb{R}^d$, then $p \equiv 0$ and thus $l \equiv 0$.

This implies $\operatorname{span}\{\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k \,|\, \boldsymbol{w} \in \mathbb{R}^d\} = \mathbb{H}_k$. Indeed, if there exists $h \in \mathbb{H}_k$ which is not in $\operatorname{span}\{\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k \,|\, \boldsymbol{w} \in \mathbb{R}^d\}$, then by the theorem of Hahn-Banach (see Theorem B.8), there exists a non-zero functional in $\mathbb{H}_k'$ vanishing on $\operatorname{span}\{\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k \,|\, \boldsymbol{w} \in \mathbb{R}^d\}$. This contradicts the previous observation.

By the universality of $\mathcal{H}$ it is not hard to see that $\boldsymbol{x} \mapsto (\boldsymbol{w} \cdot \boldsymbol{x})^k \in X$ for all $\boldsymbol{w} \in \mathbb{R}^d$. Therefore, we have $\mathbb{H}_k \subseteq X$ for all $k \in \mathbb{N}_0$. □

By the above lemma, in order to verify that $\mathcal{N}_d^1(\sigma; 1)$ is a universal approximator, it suffices to show that $\mathcal{N}_1^1(\sigma; 1)$ is a universal approximator. We first show that this is the case for sigmoidal activations.

Definition 3.11. An activation function $\sigma : \mathbb{R} \to \mathbb{R}$ is called **sigmoidal**, if $\sigma \in C^0(\mathbb{R})$, $\lim_{x \to \infty} \sigma(x) = 1$ and $\lim_{x \to -\infty} \sigma(x) = 0$.

For sigmoidal activation functions we can now conclude the universality in the univariate case.

Lemma 3.12. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be monotonically increasing and sigmoidal. Then $C^0(\mathbb{R}) \subseteq \overline{\mathcal{N}_1^1(\sigma; 1)}^{\mathrm{cc}}$.*

We prove Lemma 3.12 in Exercise 3.24. Lemma 3.10 and Lemma 3.12 show Theorem 3.8 in the special case where σ is monotonically increasing and sigmoidal. For the general case, let us continue with (ii) and consider C^∞ activations.

Lemma 3.13. *If $\sigma \in C^\infty(\mathbb{R})$ and σ is not a polynomial, then $\mathcal{N}_1^1(\sigma; 1)$ is dense in $C^0(\mathbb{R})$.*

Proof. Denote $X := \overline{\mathcal{N}_1^1(\sigma; 1)}^{\mathrm{cc}}$. We show again that all polynomials belong to X. An application of the Stone-Weierstrass theorem then gives the statement.

Fix $b \in \mathbb{R}$ and denote $f_x(w) := \sigma(wx + b)$ for all x, $w \in \mathbb{R}$. By Taylor's theorem, for $h \neq 0$

$$\begin{aligned} \frac{\sigma((w+h)x + b) - \sigma(wx + b)}{h} &= \frac{f_x(w+h) - f_x(w)}{h} \\ &= f_x'(w) + \frac{h}{2} f_x''(\xi) \\ &= f_x'(w) + \frac{h}{2} x^2 \sigma''(\xi x + b) \end{aligned} \tag{3.1.3}$$

for some $\xi = \xi(h)$ between w and $w+h$. Note that the left-hand side belongs to $\mathcal{N}_1^1(\sigma;1)$ as a function of x. Since $\sigma'' \in C^0(\mathbb{R})$, for every compact set $K \subseteq \mathbb{R}$

$$\sup_{x\in K}\sup_{|h|\leq 1}|x^2\sigma''(\xi(h)x+b)| \leq \sup_{x\in K}\sup_{\eta\in[w-1,w+1]}|x^2\sigma''(\eta x+b)| < \infty.$$

Letting $h \to 0$, as a function of x the term in (3.1.3) thus converges uniformly towards $K \ni x \mapsto f_x'(w)$. Since K was arbitrary, $x \mapsto f_x'(w)$ belongs to X. Inductively applying the same argument to $f_x^{(k-1)}(w)$, we find that $x \mapsto f_x^{(k)}(w)$ belongs to X for all $k \in \mathbb{N}$, $w \in \mathbb{R}$. Observe that $f_x^{(k)}(w) = x^k\sigma^{(k)}(wx+b)$. Since σ is not a polynomial, for each $k \in \mathbb{N}$ there exists $b_k \in \mathbb{R}$ such that $\sigma^{(k)}(b_k) \neq 0$. Choosing $w = 0$, we obtain that $x \mapsto x^k$ belongs to X. □

Finally, we come to the proof of (iii)—the claim that there exists at least one non-polynomial $C^\infty(\mathbb{R})$ function in the closure of $\mathcal{N}_1^1(\sigma;1)$. The argument is split into two lemmata. Denote in the following by $C_c^\infty(\mathbb{R})$ the set of compactly supported $C^\infty(\mathbb{R})$ functions.

Lemma 3.14. *Let $\sigma \in \mathcal{M}$. Then for each $\varphi \in C_c^\infty(\mathbb{R})$ it holds $\sigma * \varphi \in \overline{\mathcal{N}_1^1(\sigma;1)}^{\mathrm{cc}}$.*

Proof. Fix $\varphi \in C_c^\infty(\mathbb{R})$ and let $a > 0$ such that $\operatorname{supp}\varphi \subseteq [-a,a]$. We have

$$\sigma * \varphi(x) = \int_{\mathbb{R}} \sigma(x-y)\varphi(y)\,\mathrm{d}y.$$

Denote $y_j := -a + 2aj/n$ for $j = 0,\dots,n$ and define for $x \in \mathbb{R}$

$$f_n(x) := \frac{2a}{n}\sum_{j=0}^{n-1}\sigma(x-y_j)\varphi(y_j).$$

Clearly, $f_n \in \mathcal{N}_1^1(\sigma;1)$. We will show $f_n \xrightarrow{\mathrm{cc}} \sigma*\varphi$ as $n \to \infty$. To do so we verify uniform convergence of f_n towards $\sigma * \varphi$ on the interval $[-b,b]$ with $b > 0$ arbitrary but fixed.

For $x \in [-b,b]$

$$|\sigma*\varphi(x) - f_n(x)| \leq \sum_{j=0}^{n-1}\left|\int_{y_j}^{y_{j+1}}\sigma(x-y)\varphi(y) - \sigma(x-y_j)\varphi(y_j)\,\mathrm{d}y\right|. \tag{3.1.4}$$

Fix $\varepsilon \in (0,1)$. Since $\sigma \in \mathcal{M}$, there exist $z_1,\dots,z_M \in \mathbb{R}$ such that σ is continuous on $\mathbb{R}\backslash\{z_1,\dots,z_M\}$ (cp. (3.1.1)). With $D_\varepsilon := \bigcup_{j=1}^M (z_j-\varepsilon, z_j+\varepsilon)$, observe that σ is uniformly continuous on the compact set $K_\varepsilon := [-a-b, a+b] \cap D_\varepsilon^c$. Now let $J_c \cup J_d = \{0,\dots,n-1\}$ be a partition (depending on x), such that $j \in J_c$ if and only if $[x-y_{j+1}, x-y_j] \subseteq K_\varepsilon$. Hence, $j \in J_d$ implies the existence of $i \in \{1,\dots,M\}$ such that the distance of z_i to $[x-y_{j+1}, x-y_j]$ is at most ε. Due to the interval

$[x - y_{j+1}, x - y_j]$ having length $2a/n$, we can bound

$$\begin{aligned}\sum_{j\in J_d} y_{j+1} - y_j &= \left|\bigcup_{j\in J_d} [x - y_{j+1}, x - y_j]\right| \\ &\le \left|\bigcup_{i=1}^{M} \Big[z_i - \varepsilon - \frac{2a}{n}, z_i + \varepsilon + \frac{2a}{n}\Big]\right| \\ &\le M \cdot \Big(2\varepsilon + \frac{4a}{n}\Big).\end{aligned}$$

Next, because of the local boundedness of σ and the fact that $\varphi \in C_c^\infty$, it holds $\sup_{|y|\le a+b} |\sigma(y)| + \sup_{|y|\le a} |\varphi(y)| =: \gamma < \infty$. Hence

$$\begin{aligned}&|\sigma * \varphi(x) - f_n(x)| \\ &\quad\le \sum_{j\in J_c\cup J_d} \left|\int_{y_j}^{y_{j+1}} \sigma(x-y)\varphi(y) - \sigma(x-y_j)\varphi(y_j)\,\mathrm{d}y\right| \\ &\quad\le 2\gamma^2 M \cdot \left(2\varepsilon + \frac{4a}{n}\right) \\ &\qquad + 2a \sup_{j\in J_c} \max_{y\in[y_j, y_{j+1}]} |\sigma(x-y)\varphi(y) - \sigma(x-y_j)\varphi(y_j)|. \qquad (3.1.5)\end{aligned}$$

We can bound the term in the last maximum by

$$\begin{aligned}&|\sigma(x-y)\varphi(y) - \sigma(x-y_j)\varphi(y_j)| \\ &\quad\le |\sigma(x-y) - \sigma(x-y_j)||\varphi(y)| + |\sigma(x-y_j)||\varphi(y) - \varphi(y_j)| \\ &\quad\le \gamma \cdot \left(\sup_{\substack{z_1,z_2\in K_\varepsilon \\ |z_1-z_2|\le \frac{2a}{n}}} |\sigma(z_1) - \sigma(z_2)| + \sup_{\substack{z_1,z_2\in[-a,a] \\ |z_1-z_2|\le \frac{2a}{n}}} |\varphi(z_1) - \varphi(z_2)|\right).\end{aligned}$$

Finally, uniform continuity of σ on K_ε and φ on $[-a, a]$ imply that the last term tends to 0 as $n \to \infty$ uniformly for all $x \in [-b, b]$. This shows that there exist $C < \infty$ (independent of ε and x) and $n_\varepsilon \in \mathbb{N}$ (independent of x) such that the term in (3.1.5) is bounded by $C\varepsilon$ for all $n \ge n_\varepsilon$. Since ε was arbitrary, this yields the claim. □

Lemma 3.15. *If $\sigma \in \mathcal{M}$ and $\sigma * \varphi$ is a polynomial for all $\varphi \in C_c^\infty(\mathbb{R})$, then σ is a polynomial.*

Proof. Fix $-\infty < a < b < \infty$ and consider $C_c^\infty(a, b) := \{\varphi \in C^\infty(\mathbb{R}) \mid \operatorname{supp}\varphi \subseteq [a, b]\}$. Define a metric ρ on $C_c^\infty(a, b)$ via

$$\rho(\varphi, \psi) := \sum_{j\in\mathbb{N}_0} 2^{-j} \frac{|\varphi - \psi|_{C^j(a,b)}}{1 + |\varphi - \psi|_{C^j(a,b)}},$$

where

$$|\varphi|_{C^j(a,b)} := \sup_{x\in[a,b]} |\varphi^{(j)}(x)|.$$

Since the space of j times differentiable functions on $[a,b]$ is complete with respect to the norm $\sum_{i=0}^{j} |\cdot|_{C^i(a,b)}$, see for instance [89, Satz 104.3], the space $C_c^\infty(a,b)$ is complete with the metric ρ. For $k \in \mathbb{N}$ set

$$V_k := \{\varphi \in C_c^\infty(a,b) \,|\, \sigma * \varphi \in \mathcal{P}_k\},$$

where $\mathcal{P}_k := \operatorname{span}\{\mathbb{R} \ni x \mapsto x^j \,|\, 0 \le j \le k\}$ denotes the space of polynomials of degree at most k. Then V_k is closed with respect to the metric ρ. To see this, we only need to observe that for a converging sequence $\varphi_j \to \varphi^*$ with respect to ρ and $\varphi_j \in V_k$, it follows that $D^{k+1}(\sigma * \varphi^*) = 0$ and hence $\sigma * \varphi^*$ is a polynomial. Since $D^{k+1}(\sigma * \varphi_j) = 0$ we compute with the linearity of the convolution and the fact that $D^{k+1}(f * g) = f * D^{k+1}(g)$ for differentiable g and if both sides are well-defined that

$$\begin{aligned}
&\sup_{x\in[a,b]} |D^{k+1}(\sigma * \varphi^*)(x)| \\
&\quad = \sup_{x\in[a,b]} |\sigma * D^{k+1}(\varphi^* - \varphi_j)(x)| \\
&\quad \le |b-a| \sup_{z\in[a-b,b-a]} |\sigma(z)| \cdot \sup_{x\in[a,b]} |D^{k+1}(\varphi_j - \varphi^*)(x)|
\end{aligned}$$

and since σ is locally bounded, the right hand-side converges to 0.

By assumption we have

$$\bigcup_{k\in\mathbb{N}} V_k = C_c^\infty(a,b).$$

Baire's category theorem implies the existence of $k_0 \in \mathbb{N}$ (depending on a, b) such that V_{k_0} contains an open subset of $C_c^\infty(a,b)$. Since V_{k_0} is a vector space, it must hold $V_{k_0} = C_c^\infty(a,b)$.

We now show that $\varphi * \sigma \in \mathcal{P}_{k_0}$ for every $\varphi \in C_c^\infty(\mathbb{R})$; in other words, $k_0 = k_0(a,b)$ can be chosen independent of a and b. First consider a shift $s \in \mathbb{R}$ and let $\tilde{a} := a + s$ and $\tilde{b} := b + s$. Then with $S(x) := x + s$, for any $\varphi \in C_c^\infty(\tilde{a},\tilde{b})$ holds $\varphi \circ S \in C_c^\infty(a,b)$, and thus $(\varphi \circ S) * \sigma \in \mathcal{P}_{k_0}$. Since $(\varphi \circ S) * \sigma(x) = \varphi * \sigma(x+s)$, we conclude that $\varphi * \sigma \in \mathcal{P}_{k_0}$. Next let $-\infty < \tilde{a} < \tilde{b} < \infty$ be *arbitrary*. Then, for an integer $n > (\tilde{b} - \tilde{a})(b-a)$ we can cover $(\tilde{a},\tilde{b})$ with $n \in \mathbb{N}$ overlapping open intervals $(a_1,b_1),\dots,(a_n,b_n)$, each of length $b-a$. Any $\varphi \in C_c^\infty(\tilde{a},\tilde{b})$ can be written as $\varphi = \sum_{j=1}^n \varphi_j$ where $\varphi_j \in C_c^\infty(a_j,b_j)$. Then $\varphi * \sigma = \sum_{j=1}^n \varphi_j * \sigma \in \mathcal{P}_{k_0}$, and thus $\varphi * \sigma \in \mathcal{P}_{k_0}$ for every $\varphi \in C_c^\infty(\mathbb{R})$.

Finally, Exercise 3.25 implies $\sigma \in \mathcal{P}_{k_0}$. □

Now we can put everything together to show Theorem 3.8.

of Theorem 3.8. By Exercise 3.23 we have the implication "⇒".

For the other direction we assume that $\sigma \in \mathcal{M}$ is not a polynomial. Then by Lemma 3.15 there exists $\varphi \in C_c^\infty(\mathbb{R})$ such that $\sigma * \varphi$ is not a polynomial. According to Lemma 3.14 we have $\sigma * \varphi \in \overline{\mathcal{N}_1^1(\sigma;1)}^{cc}$. We conclude with Lemma 3.13 that $\mathcal{N}_1^1(\sigma;1)$ is a universal approximator of $C^0(\mathbb{R})$.

Finally, by Lemma 3.10, $\mathcal{N}_d^1(\sigma;1)$ is a universal approximator of $C^0(\mathbb{R}^d)$. □

3.1.3 Deep neural networks

Theorem 3.8 shows the universal approximation capability of single-hidden-layer neural networks with activation functions $\sigma \in \mathcal{M}\backslash\mathcal{P}$: they can approximate every continuous function on every compact set to arbitrary precision, given sufficient width. This result directly extends to neural networks of any fixed depth $L \geq 1$. The idea is to use the fact that the identity function can be approximated with a shallow neural network. Composing a shallow neural network approximation of the target function f with (multiple) shallow neural networks approximating the identity function, gives a deep neural network approximation of f.

Instead of directly applying Theorem 3.8, we first establish the following proposition regarding the approximation of the identity function. Rather than $\sigma \in \mathcal{M}\backslash\mathcal{P}$, it requires a different (mild) assumption on the activation function. This allows for a constructive proof, yielding explicit bounds on the neural network size, which will prove useful later in the book.

Proposition 3.16. *Let d, $L \in \mathbb{N}$, let $K \subseteq \mathbb{R}^d$ be compact, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be such that there exists an open set on which σ is differentiable and not constant. Then, for every $\varepsilon > 0$, there exists a neural network $\Phi \in \mathcal{N}_d^d(\sigma; L, d)$ such that*

$$\|\Phi(\boldsymbol{x}) - \boldsymbol{x}\|_\infty < \varepsilon \qquad \text{for all } \boldsymbol{x} \in K.$$

Proof. The proof uses the same idea as in Lemma 3.13, where we approximate the derivative of the activation function by a simple neural network. Let us first assume $d \in \mathbb{N}$ and $L = 1$.

Let $x^* \in \mathbb{R}$ be such that σ is differentiable on a neighborhood of x^* and $\sigma'(x^*) = \theta \neq 0$. Moreover, let $\boldsymbol{x}^* = (x^*, \dots, x^*) \in \mathbb{R}^d$. Then, for $\lambda > 0$ we define

$$\Phi_\lambda(\boldsymbol{x}) \coloneqq \frac{\lambda}{\theta}\sigma\left(\frac{\boldsymbol{x}}{\lambda} + \boldsymbol{x}^*\right) - \frac{\lambda}{\theta}\sigma(\boldsymbol{x}^*),$$

Then, we have, for all $\boldsymbol{x} \in K$,

$$\Phi_\lambda(\boldsymbol{x}) - \boldsymbol{x} = \lambda\frac{\sigma(\boldsymbol{x}/\lambda + \boldsymbol{x}^*) - \sigma(\boldsymbol{x}^*)}{\theta} - \boldsymbol{x}. \tag{3.1.6}$$

If $x_i = 0$ for $i \in \{1, \dots, d\}$, then (3.1.6) shows that $(\Phi_\lambda(\boldsymbol{x}) - \boldsymbol{x})_i = 0$. Otherwise

$$|(\Phi_\lambda(\boldsymbol{x}) - \boldsymbol{x})_i| = \frac{|x_i|}{|\theta|}\left|\frac{\sigma(x_i/\lambda + x^*) - \sigma(x^*)}{x_i/\lambda} - \theta\right|.$$

By the definition of the derivative, we have that $|(\Phi_\lambda(\boldsymbol{x}) - \boldsymbol{x})_i| \to 0$ for $\lambda \to \infty$ uniformly for all $\boldsymbol{x} \in K$ and $i \in \{1, \dots, d\}$. Therefore, $|\Phi_\lambda(\boldsymbol{x}) - \boldsymbol{x}| \to 0$ for $\lambda \to \infty$ uniformly for all $\boldsymbol{x} \in K$.

The extension to $L > 1$ is straight forward and is the content of Exercise 3.27. □

Using the aforementioned generalization of Proposition 3.16 to arbitrary non-polynomial activation functions $\sigma \in \mathcal{M}$, we obtain the following extension of Theorem 3.8.

Corollary 3.17. *Let $d \in \mathbb{N}$, $L \in \mathbb{N}$ and $\sigma \in \mathcal{M}$. Then $\mathcal{N}_d^1(\sigma; L)$ is a universal approximator of $C^0(\mathbb{R}^d)$ if and only if σ is not a polynomial.*

Proof. We only show the implication "$\Leftarrow$". The other direction is again left as an exercise, see Exercise 3.23.

Assume $\sigma \in \mathcal{M}$ is not a polynomial, let $K \subseteq \mathbb{R}^d$ be compact, and let $f \in C^0(\mathbb{R}^d)$. Fix $\varepsilon \in (0, 1)$. We need to show that there exists a neural network $\Phi \in \mathcal{N}_d^1(\sigma; L)$ such that $\sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| < \varepsilon$. The case $L = 1$ holds by Theorem 3.8, so let $L > 1$.

By Theorem 3.8, there exist $\Phi_{\text{shallow}} \in \mathcal{N}_d^1(\sigma; 1)$ such that

$$\sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x}) - \Phi_{\text{shallow}}(\boldsymbol{x})| < \frac{\varepsilon}{2}. \tag{3.1.7}$$

Compactness of $\{f(\boldsymbol{x}) \,|\, \boldsymbol{x} \in K\}$ implies that we can find $n > 0$ such that

$$\{\Phi_{\text{shallow}}(\boldsymbol{x}) \,|\, \boldsymbol{x} \in K\} \subseteq [-n, n]. \tag{3.1.8}$$

Let $\Phi_{\text{id}} \in \mathcal{N}_1^1(\sigma; L - 1)$ be an approximation to the identity such that

$$\sup_{x \in [-n,n]} |x - \Phi_{\text{id}}(x)| < \frac{\varepsilon}{2}, \tag{3.1.9}$$

which is possibly by the extension of Proposition 3.16 to general non-polynomial activation functions $\sigma \in \mathcal{M}$.

Denote $\Phi := \Phi_{\text{id}} \circ \Phi_{\text{shallow}}$. According to Proposition 2.3 (iv) holds $\Phi \in \mathcal{N}_d^1(\sigma; L)$ as desired. Moreover (3.1.7), (3.1.8), (3.1.9) imply

$$\begin{aligned}
\sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| &= \sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x}) - \Phi_{\text{id}}(\Phi_{\text{shallow}}(\boldsymbol{x}))| \\
&\leq \sup_{\boldsymbol{x} \in K} \big(|f(\boldsymbol{x}) - \Phi_{\text{shallow}}(\boldsymbol{x})| + |\Phi_{\text{shallow}}(\boldsymbol{x}) - \Phi_{\text{id}}(\Phi_{\text{shallow}}(\boldsymbol{x}))| \big) \\
&\leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.
\end{aligned}$$

This concludes the proof. □

3.1.4 Other norms

Additional to the continuous functions, universal approximation theorems can be shown for various other function classes and topologies, which may also allow for the approximation of functions exhibiting discontinuities or singularities. To give but one example, we next state such a result for Lebesgue spaces on compact sets. The proof is left to the reader, see Exercise 3.26.

Corollary 3.18. *Let $d \in \mathbb{N}$, $L \in \mathbb{N}$, $p \in [1, \infty)$, and let $\sigma \in \mathcal{M}$ not be a polynomial. Then for every $\varepsilon > 0$, every compact $K \subseteq \mathbb{R}^d$, and every $f \in L^p(K)$ there exists $\Phi^{f,\varepsilon} \in \mathcal{N}_d^1(\sigma; L)$ such that*

$$\left(\int_K |f(\boldsymbol{x}) - \Phi(\boldsymbol{x})|^p \, \mathrm{d}\boldsymbol{x} \right)^{1/p} \leq \varepsilon.$$

3.2 Superexpressive activations and Kolmogorov's superposition theorem

In the previous section, we saw that a large class of activation functions allow for universal approximation. However, these results did not provide any insights into the necessary neural network size for achieving a specific accuracy.

Before exploring this topic further in the following chapters, we next present a remarkable result that shows how the required neural network size is significantly influenced by the choice of activation function. The result asserts that, with the appropriate activation function, every $f \in C^0(K)$ on a compact set $K \subseteq \mathbb{R}^d$ can be approximated to *every desired accuracy* $\varepsilon > 0$ using a neural network of size $O(d^2)$; in particular the neural network size is independent of $\varepsilon > 0$, K, and f. We will first discuss the one-dimensional case.

Proposition 3.19. *There exists a continuous activation function $\sigma : \mathbb{R} \to \mathbb{R}$ such that for every compact $K \subseteq \mathbb{R}$, every $\varepsilon > 0$ and every $f \in C^0(K)$ there exists $\Phi(x) = \sigma(wx + b) \in \mathcal{N}_1^1(\sigma; 1, 1)$ such that*

$$\sup_{x \in K} |f(x) - \Phi(x)| < \varepsilon.$$

Proof. Denote by $\tilde{\mathcal{P}}_n$ all polynomials $p(x) = \sum_{j=0}^n q_j x^j$ with rational coefficients, i.e. such that $q_j \in \mathbb{Q}$ for all $j = 0, \dots, n$. Then $\tilde{\mathcal{P}}_n$ can be identified with the n-fold cartesian product $\mathbb{Q} \times \cdots \times \mathbb{Q}$, and thus $\tilde{\mathcal{P}}_n$ is a countable set. Consequently also the set $\tilde{\mathcal{P}} := \bigcup_{n \in \mathbb{N}} \tilde{\mathcal{P}}_n$ of all polynomials with rational coefficients is countable. Let $(p_i)_{i \in \mathbb{Z}}$ be an enumeration of these polynomials, and set

$$\sigma(x) := \begin{cases} p_i(x - 2i) & \text{if } x \in [2i, 2i + 1] \\ p_i(1)(2i + 2 - x) + p_{i+1}(0)(x - 2i - 1) & \text{if } x \in (2i + 1, 2i + 2). \end{cases}$$

In words, σ equals p_i on even intervals $[2i, 2i + 1]$ and is linear on odd intervals $[2i + 1, 2i + 2]$, resulting in a continuous function overall.

We first assume $K = [0, 1]$. By Example 3.5, for every $\varepsilon > 0$ exists $p(x) = \sum_{j=1}^n r_j x^j$ such that $\sup_{x \in [0,1]} |p(x) - f(x)| < \varepsilon/2$. Now choose $q_j \in \mathbb{Q}$ so close to r_j such that $\tilde{p}(x) := \sum_{j=1}^n q_j x^j$ satisfies $\sup_{x \in [0,1]} |\tilde{p}(x) - p(x)| < \varepsilon/2$. Let $i \in \mathbb{Z}$ such that $\tilde{p}(x) = p_i(x)$, i.e., $p_i(x) = \sigma(2i + x)$ for all $x \in [0, 1]$. Then $\sup_{x \in [0,1]} |f(x) - \sigma(x + 2i)| < \varepsilon$.

For general compact K assume that $K \subseteq [a, b]$. By Tietze's extension theorem, f allows a continuous extension to $[a, b]$, so without loss of generality $K = [a, b]$. By the first case we can find $i \in \mathbb{Z}$ such that with $y = (x - a)/(b - a)$ (i.e. $y \in [0, 1]$ if $x \in [a, b]$)

$$\sup_{x \in [a,b]} \left| f(x) - \sigma\left(\frac{x - a}{b - a} + 2i \right) \right| = \sup_{y \in [0,1]} |f(y \cdot (b - a) + a) - \sigma(y + 2i)| < \varepsilon,$$

which gives the statement with $w = 1/(b - a)$ and $b = -a \cdot (b - a) + 2i$. □

To extend this result to arbitrary dimension, we will use Kolmogorov's superposition theorem. It states that every continuous function of d variables can be expressed as a composition of functions that each depend only on one variable. We omit the technical proof, which can be found in [120].

Theorem 3.20 (Kolmogorov)**.** *For every $d \in \mathbb{N}$ there exist $2d^2 + d$ monotonically increasing functions $\varphi_{i,j} \in C^0(\mathbb{R})$, $i = 1, \dots, d$, $j = 1, \dots, 2d+1$, such that for every $f \in C^0([0,1]^d)$ there exist functions $f_j \in C^0(\mathbb{R})$, $j = 1, \dots, 2d+1$ satisfying*

$$f(\boldsymbol{x}) = \sum_{j=1}^{2d+1} f_j \left(\sum_{i=1}^{d} \varphi_{i,j}(x_i) \right) \qquad \text{for all } \boldsymbol{x} \in [0,1]^d.$$

Corollary 3.21. *Let $d \in \mathbb{N}$. With the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ from Proposition 3.19, for every compact $K \subseteq \mathbb{R}^d$, every $\varepsilon > 0$ and every $f \in C^0(K)$ there exists $\Phi \in \mathcal{N}_d^1(\sigma; 2, 2d^2 + d)$ (i.e.* $\operatorname{width}(\Phi) = 2d^2 + d$ *and* $\operatorname{depth}(\Phi) = 2$*) such that*

$$\sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| < \varepsilon.$$

Proof. Without loss of generality we can assume $K = [0,1]^d$: the extension to the general case then follows by Tietze's extension theorem and a scaling argument as in the proof of Proposition 3.19.

Let f_j, $\varphi_{i,j}$, $i = 1, \dots, d$, $j = 1, \dots, 2d+1$ be as in Theorem 3.20. Fix $\varepsilon > 0$. Let $a > 0$ be so large that

$$\sup_{i,j} \sup_{x \in [0,1]} |\varphi_{i,j}(x)| \leq a.$$

Since each f_j is uniformly continuous on the compact set $[-da, da]$, we can find $\delta > 0$ such that

$$\sup_{j} \sup_{\substack{|y-\tilde{y}|<\delta \\ |y|,|\tilde{y}| \leq da}} |f_j(y) - f_j(\tilde{y})| < \frac{\varepsilon}{2(2d+1)}. \tag{3.2.1}$$

By Proposition 3.19 there exist $w_{i,j}$, $b_{i,j} \in \mathbb{R}$ such that

$$\sup_{i,j} \sup_{x \in [0,1]} |\varphi_{i,j}(x) - \underbrace{\sigma(w_{i,j}x + b_{i,j})}_{=:\tilde{\varphi}_{i,j}(x)}| < \frac{\delta}{d} \tag{3.2.2}$$

and w_j, $b_j \in \mathbb{R}$ such that

$$\sup_{j} \sup_{|y| \leq a+\delta} |f_j(y) - \underbrace{\sigma(w_j y + b_j)}_{=:\tilde{f}_j(y)}| < \frac{\varepsilon}{2(2d+1)}. \tag{3.2.3}$$

Then for all $\boldsymbol{x} \in [0,1]^d$ by (3.2.2)

$$\left| \sum_{i=1}^{d} \varphi_{i,j}(x_i) - \sum_{i=1}^{d} \tilde{\varphi}_{i,j}(x_i) \right| < d\frac{\delta}{d} = \delta.$$

Thus with

$$y_j := \sum_{j=1}^{d} \varphi_{i,j}(x_i), \qquad \tilde{y}_j := \sum_{j=1}^{d} \tilde{\varphi}_{i,j}(x_i)$$

it holds $|y_j - \tilde{y}_j| < \delta$. Using (3.2.1) and (3.2.3) we conclude

$$\begin{aligned}
\left| f(\boldsymbol{x}) - \sum_{j=1}^{2d+1} \sigma\left(w_j \cdot \left(\sum_{i=1}^{d} \sigma(w_{i,j} x_i + b_{i,j}) \right) + b_j \right) \right| &= \left| \sum_{j=1}^{2d+1} (f_j(y_j) - \tilde{f}_j(\tilde{y}_j)) \right| \\
&\leq \sum_{j=1}^{2d+1} \Big(|f_j(y_j) - f_j(\tilde{y}_j)| + |f_j(\tilde{y}_j) - \tilde{f}_j(\tilde{y}_j)| \Big) \\
&\leq \sum_{j=1}^{2d+1} \left(\frac{\varepsilon}{2(2d+1)} + \frac{\varepsilon}{2(2d+1)} \right) \leq \varepsilon.
\end{aligned}$$

This concludes the proof. □

Kolmogorov's superposition theorem is intriguing as it shows that approximating d-dimensional functions can be reduced to the (generally much simpler) one-dimensional case through compositions. Neural networks, by nature, are well suited to approximate functions with compositional structures. However, the functions f_j in Theorem 3.20, even though only one-dimensional, could become very complex and challenging to approximate themselves if d is large.

Similarly, the "magic" activation function in Proposition 3.19 encodes the information of all rational polynomials on the unit interval, which is why a neural network of size $O(1)$ suffices to approximate every function to arbitrary accuracy. Naturally, no practical algorithm can efficiently identify appropriate neural network weights and biases for this architecture. As such, the results presented in Section 3.2 should be taken with a pinch of salt as their practical relevance is highly limited. Nevertheless, they highlight that while universal approximation is a fundamental and important property of neural networks, it leaves many aspects unexplored. To gain further insight into practically relevant architectures, in the following chapters, we investigate neural networks with activation functions such as the ReLU.

Bibliography and further reading

The foundation of universal approximation theorems goes back to the late 1980s with seminal works by Cybenko [44], Hornik et al. [95, 94], Funahashi [63] and Carroll and Dickinson [33]. These results were subsequently extended to a wider range of activation functions and architectures. The present analysis in Section 3.1 closely follows the arguments in [132], where it was essentially shown that universal approximation can be achieved if the activation function is not polynomial.

Kolmogorov's superposition theorem stated in Theorem 3.20 was originally proven in 1957 [120]. For a more recent and constructive proof see for instance [26]. Kolmogorov's theorem and its obvious connections to neural networks have inspired various research in this field, e.g. [162, 124, 151, 205, 104], with its practical relevance being debated [68, 123]. The idea for the "magic" activation function in Section 3.2 comes from [140] where it is shown that such an activation function can even be chosen monotonically increasing.

Exercises

Exercise 3.22. Write down a generator of a (minimal) topology on $C^0(\mathbb{R}^d)$ such that $f_n \to f \in C^0(\mathbb{R}^d)$ if and only if $f_n \xrightarrow{\text{cc}} f$, and show this equivalence. This topology is referred to as the topology of compact convergence.

Exercise 3.23. Show the implication "$\Rightarrow$" of Theorem 3.8 and Corollary 3.17.

Exercise 3.24. Prove Lemma 3.12. *Hint*: Consider $\sigma(nx)$ for large $n \in \mathbb{N}$.

Exercise 3.25. Let $k \in \mathbb{N}$, $\sigma \in \mathcal{M}$ and assume that $\sigma * \varphi \in \mathcal{P}_k$ for all $\varphi \in C_c^\infty(\mathbb{R})$. Show that $\sigma \in \mathcal{P}_k$.

Hint: Consider $\psi \in C_c^\infty(\mathbb{R})$ such that $\psi \geq 0$ and $\int_{\mathbb{R}} \psi(x)\,\mathrm{d}x = 1$ and set $\psi_\varepsilon(x) := \psi(x/\varepsilon)/\varepsilon$. Use that away from the discontinuities of σ it holds $\psi_\varepsilon * \sigma(x) \to \sigma(x)$ as $\varepsilon \to 0$. Conclude that σ is piecewise in $\mathcal{P}_k$, and finally show that $\sigma \in C^k(\mathbb{R})$.

Exercise 3.26. Prove Corollary 3.18 with the use of Corollary 3.17.

Exercise 3.27. Complete the proof of Proposition 3.16 for $L > 1$.

Chapter 4

Splines

In Chapter 3, we saw that sufficiently large neural networks can approximate every continuous function to arbitrary accuracy. However, these results did not further specify the meaning of "sufficiently large" or what constitutes a suitable architecture. Ideally, given a function f, and a desired accuracy $\varepsilon > 0$, we would like to have a (possibly sharp) bound on the required size, depth, and width guaranteeing the existence of a neural network approximating f up to error ε.

The field of approximation theory establishes such trade-offs between properties of the function f (e.g., its smoothness), the approximation accuracy, and the number of parameters needed to achieve this accuracy. For example, given k, $d \in \mathbb{N}$, how many parameters are required to approximate a function $f : [0,1]^d \to \mathbb{R}$ with $\|f\|_{C^k([0,1]^d)} \leq 1$ up to uniform error ε? Splines are known to achieve this approximation accuracy with a superposition of $O(\varepsilon^{-d/k})$ simple (piecewise polynomial) basis functions. In this chapter, following [146], we show that certain sigmoidal neural networks can match this performance in terms of the neural network size. In fact, from an approximation theoretical viewpoint we show that the considered neural networks are at least as expressive as superpositions of splines.

4.1 B-splines and smooth functions

We introduce a simple type of spline and its approximation properties below.

Definition 4.1. For $n \in \mathbb{N}$, the **univariate cardinal B-spline** order $n \in \mathbb{N}$ is given by

$$\mathcal{S}_n(x) \coloneqq \frac{1}{(n-1)!} \sum_{\ell=0}^{n} (-1)^\ell \binom{n}{\ell} \sigma_{\mathrm{ReLU}}(x-\ell)^{n-1} \qquad \text{for } x \in \mathbb{R}, \tag{4.1.1}$$

where $0^0 \coloneqq 0$ and σ_{ReLU} denotes the ReLU activation function.

By shifting and dilating the cardinal B-spline, we obtain a system of univariate splines. Taking tensor products of these univariate splines yields a set of higher-dimensional functions known as the multivariate B-splines.

Definition 4.2. For $t \in \mathbb{R}$ and n, $\ell \in \mathbb{N}$ we define $\mathcal{S}_{\ell,t,n} \coloneqq \mathcal{S}_n(2^\ell(\cdot - t))$. Additionally, for $d \in \mathbb{N}$, $\boldsymbol{t} \in \mathbb{R}^d$, and n, $\ell \in \mathbb{N}$, we define the **the multivariate B-spline** $\mathcal{S}^d_{\ell,\boldsymbol{t},n}$ as

$$\mathcal{S}^d_{\ell,\boldsymbol{t},n}(\boldsymbol{x}) \coloneqq \prod_{i=1}^{d} \mathcal{S}_{\ell,t_i,n}(x_i) \qquad \text{for } \boldsymbol{x} = (x_1, \dots x_d) \in \mathbb{R}^d,$$

and

$$\mathcal{B}^n \coloneqq \left\{ \mathcal{S}^d_{\ell,\boldsymbol{t},n} \,\middle|\, \ell \in \mathbb{N}, \boldsymbol{t} \in \mathbb{R}^d \right\}$$

is the **dictionary of B-splines of order** n.

Having introduced the system $\mathcal{B}^n$, we would like to understand how well we can represent each smooth function by superpositions of elements of $\mathcal{B}^n$. The following theorem is adapted from the more general result [168, Theorem 7]; also see [141, Theorem D.3] for a presentation closer to the present formulation.

Theorem 4.3. *Let d, n, $k \in \mathbb{N}$ such that $0 < k \leq n$. Then there exists C such that for every $f \in C^k([0,1]^d)$ and every $N \in \mathbb{N}$, there exist $c_i \in \mathbb{R}$ with $|c_i| \leq C\|f\|_{L^\infty([0,1]^d)}$ and $B_i \in \mathcal{B}^n$ for $i = 1, \dots, N$, such that*

$$\left\| f - \sum_{i=1}^{N} c_i B_i \right\|_{L^\infty([0,1]^d)} \leq C N^{-\frac{k}{d}} \|f\|_{C^k[0,1]^d}.$$

Remark 4.4. There are a couple of critical concepts in Theorem 4.3 that will reappear throughout this book. The number of parameters N determines the approximation accuracy $N^{-k/d}$. This implies that achieving accuracy $\varepsilon > 0$ requires $O(\varepsilon^{-d/k})$ parameters (according to this upper bound), which grows exponentially in d. This exponential dependence on d is referred to as the "curse of dimension" and will be discussed again in the subsequent chapters. The smoothness parameter k has the opposite effect of d, and improves the convergence rate. Thus, smoother functions can be approximated with fewer B-splines than rougher functions. This more efficient approximation requires the use of B-splines of order n with $n \geq k$. We will see in the following, that the order of the B-spline is closely linked to the concept of depth in neural networks.

4.2 Reapproximation of B-splines with sigmoidal activations

We now show that the approximation rates of B-splines can be transfered to certain neural networks. The following argument is based on [144].

Definition 4.5. A function $\sigma : \mathbb{R} \to \mathbb{R}$ is called **sigmoidal of order** $q \in \mathbb{N}$, if $\sigma \in C^{q-1}(\mathbb{R})$ and there exists $C > 0$ such that

$$\begin{aligned} &\frac{\sigma(x)}{x^q} \to 0 && \text{as } x \to -\infty, \\ &\frac{\sigma(x)}{x^q} \to 1 && \text{as } x \to \infty, \\ &|\sigma(x)| \leq C \cdot (1 + |x|)^q && \text{for all } x \in \mathbb{R}. \end{aligned}$$

Example 4.6. The rectified power unit $x \mapsto \sigma_{\mathrm{ReLU}}(x)^q$ is sigmoidal of order q.

Our goal in the following is to show that neural networks can approximate a linear combination of N B-splines with a number of parameters that is proportional to N. As an immediate consequence of Theorem 4.3, we then obtain a convergence rate for neural networks. Let us start by approximating a single univariate B-spline with a neural network of fixed size.

Proposition 4.7. *Let $n \in \mathbb{N}$, $n \geq 2$, $K > 0$, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be sigmoidal of order $q \geq 2$. There exists a constant $C > 0$ such that for every $\varepsilon > 0$ there is a neural network $\Phi^{\mathcal{S}_n}$ with activation function σ, $\lceil \log_q(n-1) \rceil$ layers, and size C, such that*

$$\left\| \mathcal{S}_n - \Phi^{\mathcal{S}_n} \right\|_{L^\infty([-K,K])} \leq \varepsilon.$$

Proof. By definition (4.1.1), $\mathcal{S}_n$ is a linear combination of $n+1$ shifts of $\sigma_{\mathrm{ReLU}}^{n-1}$. We start by approximating $\sigma_{\mathrm{ReLU}}^{n-1}$. It is not hard to see (Exercise 4.10) that, for every $K' > 0$ and every $t \in \mathbb{N}$

$$\left| a^{-q^t} \underbrace{\sigma \circ \sigma \circ \cdots \circ \sigma(ax)}_{t- \text{ times}} - \sigma_{\mathrm{ReLU}}(x)^{q^t} \right| \to 0 \qquad \text{as } a \to \infty \tag{4.2.1}$$

uniformly for all $x \in [-K', K']$.

Set $t := \lceil \log_q(n-1) \rceil$. Then $t \geq 1$ since $n \geq 2$, and $q^t \geq n-1$. Thus, for every $K' > 0$ and $\varepsilon > 0$ there exists a neural network $\Phi_\varepsilon^{q^t}$ with $\lceil \log_q(n-1) \rceil$ layers satisfying

$$\left| \Phi_\varepsilon^{q^t}(x) - \sigma_{\mathrm{ReLU}}(x)^{q^t} \right| \leq \varepsilon \qquad \text{for all } x \in [-K', K']. \tag{4.2.2}$$

This shows that we can approximate the ReLU to the power of $q^t \geq n-1$. However, our goal is to obtain an approximation of the ReLU raised to the power $n-1$, which could be smaller than q^t. To reduce the order, we emulate approximate derivatives of $\Phi_\varepsilon^{q^t}$. Concretely, we show the following claim: For all $1 \leq p \leq q^t$ for every $K' > 0$ and $\varepsilon > 0$ there exists a neural network Φ_ε^p having $\lceil \log_q(n-1) \rceil$ layers and satisfying

$$|\Phi_\varepsilon^p(x) - \sigma_{\mathrm{ReLU}}(x)^p| \leq \varepsilon \qquad \text{for all } x \in [-K', K']. \tag{4.2.3}$$

The claim holds for $p = q^t$. We now proceed by induction over $p = q^t, q^t - 1, \dots$ Assume (4.2.3) holds for some $p \in \{2, \dots, q^t\}$. Fix $\delta \geq 0$. Then

$$\begin{aligned}&\left|\frac{\Phi^p_{\delta^2}(x+\delta) - \Phi^p_{\delta^2}(x)}{p\delta} - \sigma_{\mathrm{ReLU}}(x)^{p-1}\right| \\ &\qquad \leq 2\frac{\delta}{p} + \left|\frac{\sigma_{\mathrm{ReLU}}(x+\delta)^p - \sigma_{\mathrm{ReLU}}(x)^p}{p\delta} - \sigma_{\mathrm{ReLU}}(x)^{p-1}\right|.\end{aligned}$$

Hence, by the binomial theorem it follows that there exists $\delta_* > 0$ such that

$$\left|\frac{\Phi^p_{\delta_*^2}(x+\delta_*) - \Phi^p_{\delta_*^2}(x)}{p\delta_*} - \sigma_{\mathrm{ReLU}}(x)^{p-1}\right| \leq \varepsilon,$$

for all $x \in [-K', K']$. By Proposition 2.3, $(\Phi^p_{\delta_*^2}(x+\delta_*) - \Phi^p_{\delta_*^2})/(p\delta_*)$ is a neural network with $\lceil \log_q(n-1) \rceil$ layers and size independent from ε. Calling this neural network Φ^{p-1}_ε shows that (4.2.3) holds for $p-1$, which concludes the induction argument and proves the claim.

For every neural network Φ, every spatial translation $\Phi(\cdot - t)$ is a neural network of the same architecture. Hence, every term in the sum (4.1.1) can be approximated to arbitrary accuracy by a neural network of a fixed size. Since by Proposition 2.3, sums of neural networks of the same depth are again neural networks of the same depth, the result follows. □

Next, we extend Proposition 4.7 to the multivariate splines $\mathcal{S}^d_{\ell,\boldsymbol{t},n}$ for arbitrary ℓ, $d \in \mathbb{N}$, $\boldsymbol{t} \in \mathbb{R}^d$.

Proposition 4.8. *Let n, $d \in \mathbb{N}$, $n \geq 2$, $K > 0$, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be sigmoidal of order $q \geq 2$. Further let $\ell \in \mathbb{N}$ and $\boldsymbol{t} \in \mathbb{R}^d$.*

Then, there exists a constant $C > 0$ such that for every $\varepsilon > 0$ there is a neural network $\Phi^{\mathcal{S}^d_{\ell,\boldsymbol{t},n}}$ with activation function σ, $\lceil \log_2(d) \rceil + \lceil \log_q(k-1) \rceil$ layers, and size C, such that

$$\left\| \mathcal{S}^d_{\ell,\boldsymbol{t},n} - \Phi^{\mathcal{S}^d_{\ell,\boldsymbol{t},n}} \right\|_{L^\infty([-K,K]^d)} \leq \varepsilon.$$

Proof. By definition $\mathcal{S}^d_{\ell,\boldsymbol{t},n}(\boldsymbol{x}) = \prod_{i=1}^d \mathcal{S}_{\ell,t_i,n}(x_i)$ where

$$\mathcal{S}_{\ell,t_i,n}(x_i) = \mathcal{S}_n(2^\ell(x_i - t_i)).$$

By Proposition 4.7 there exist a constant $C' > 0$ such that for each $i = 1, \dots, d$ and all $\varepsilon > 0$, there is a neural network $\Phi^{\mathcal{S}_{\ell,t_i,n}}$ with size C' and $\lceil \log_q(n-1) \rceil$ layers such that

$$\left\| \mathcal{S}_{\ell,t_i,n} - \Phi^{\mathcal{S}_{\ell,t_i,n}} \right\|_{L^\infty([-K,K]^d)} \leq \varepsilon.$$

If $d = 1$, this shows the statement. For general d, it remains to show that the product of the $\Phi^{\mathcal{S}_{\ell,t_i,n}}$ for $i = 1, \dots, d$ can be approximated.

We first prove the following claim by induction: For every $d \in \mathbb{N}$, $d \geq 2$, there exists a constant $C'' > 0$, such that for all $K' \geq 1$ and all $\varepsilon > 0$ there exists a neural network $\Phi_{\mathrm{mult},\varepsilon,d}$ with size

C'', $\lceil \log_2(d) \rceil$ layers, and activation function σ such that for all $x_1, \dots, x_d$ with $|x_i| \leq K'$ for all $i = 1, \dots, d$,

$$\left| \Phi_{\text{mult},\varepsilon,d}(x_1, \dots, x_d) - \prod_{i=1}^{d} x_i \right| < \varepsilon. \tag{4.2.4}$$

For the base case, let $d = 2$. Similar to the proof of Proposition 4.7, one can show that there exists $C''' > 0$ such that for every $\varepsilon > 0$ and $K' > 0$ there exists a neural network $\Phi_{\text{square},\varepsilon}$ with one hidden layer and size C''' such that

$$|\Phi_{\text{square},\varepsilon} - \sigma_{\text{ReLU}}(x)^2| \leq \varepsilon \qquad \text{for all } |x| \leq K'.$$

For every $x = (x_1, x_2) \in \mathbb{R}^2$

$$\begin{aligned} x_1 x_2 &= \frac{1}{2}\left((x_1 + x_2)^2 - x_1^2 - x_2^2\right) \\ &= \frac{1}{2}\left(\sigma_{\text{ReLU}}(x_1 + x_2)^2 + \sigma_{\text{ReLU}}(-x_1 - x_2)^2 - \sigma_{\text{ReLU}}(x_1)^2 \right. \\ &\quad \left. - \sigma_{\text{ReLU}}(-x_1)^2 - \sigma_{\text{ReLU}}(x_2)^2 - \sigma_{\text{ReLU}}(-x_2)^2\right). \end{aligned} \tag{4.2.5}$$

Each term on the right-hand side can be approximated up to uniform error $\varepsilon/6$ with a network of size C''' and one hidden layer. By Proposition 2.3, we conclude that there exists a neural network $\Phi_{\text{mult},\varepsilon,2}$ satisfying (4.2.4) for $d = 2$.

Assume the induction hypothesis (4.2.4) holds for $d - 1 \geq 1$, and let $\varepsilon > 0$ and $K' \geq 1$. We have

$$\prod_{i=1}^{d} x_i = \prod_{i=1}^{\lfloor d/2 \rfloor} x_i \cdot \prod_{i=\lfloor d/2 \rfloor + 1}^{d} x_i. \tag{4.2.6}$$

We will now approximate each of the terms in the product on the right-hand side of (4.2.6) by a neural network using the induction assumption.

For simplicity assume in the following that $\lceil \log_2(\lfloor d/2 \rfloor) \rceil = \lceil \log_2(d - \lfloor d/2 \rfloor) \rceil$. The general case can be addressed via Proposition 3.16. By the induction assumption there then exist neural networks $\Phi_{\text{mult},1}$ and $\Phi_{\text{mult},2}$ both with $\lceil \log_2(\lfloor d/2 \rfloor) \rceil$ layers, such that for all x_i with $|x_i| \leq K'$ for $i = 1, \dots, d$

$$\left| \Phi_{\text{mult},1}(x_1, \dots, x_{\lfloor d/2 \rfloor}) - \prod_{i=1}^{\lfloor d/2 \rfloor} x_i \right| < \frac{\varepsilon}{4((K')^{\lfloor d/2 \rfloor} + \varepsilon)},$$

$$\left| \Phi_{\text{mult},2}(x_{\lfloor d/2 \rfloor + 1}, \dots, x_d) - \prod_{i=\lfloor d/2 \rfloor + 1}^{d} x_i \right| < \frac{\varepsilon}{4((K')^{\lfloor d/2 \rfloor} + \varepsilon)}.$$

By Proposition 2.3, $\Phi_{\text{mult},\varepsilon,d} \coloneqq \Phi_{\text{mult},\varepsilon/2,2} \circ (\Phi_{\text{mult},1}, \Phi_{\text{mult},2})$ is a neural network with $1 + \lceil \log_2(\lfloor d/2 \rfloor) \rceil = \lceil \log_2(d) \rceil$ layers. By construction, the size of $\Phi_{\text{mult},\varepsilon,d}$ does not depend on K' or ε. Thus, to complete the induction, it only remains to show (4.2.4).

For all a, b, c, $d \in \mathbb{R}$ holds

$$|ab - cd| \leq |a||b - d| + |d||a - c|.$$

Hence, for $x_1, \dots, x_d$ with $|x_i| \leq K'$ for all $i = 1, \dots, d$, we have that

$$\begin{aligned}
&\left| \prod_{i=1}^{d} x_i - \Phi_{\text{mult},\varepsilon,d}(x_1, \dots, x_d) \right| \\
&\leq \frac{\varepsilon}{2} + \left| \prod_{i=1}^{\lfloor d/2 \rfloor} x_i \cdot \prod_{i=\lfloor d/2 \rfloor + 1}^{d} x_i - \Phi_{\text{mult},1}(x_1, \dots, x_{\lfloor d/2 \rfloor}) \Phi_{\text{mult},2}(x_{\lfloor d/2 \rfloor + 1}, \dots, x_d) \right| \\
&\leq \frac{\varepsilon}{2} + |K'|^{\lfloor d/2 \rfloor} \frac{\varepsilon}{4((K')^{\lfloor d/2 \rfloor} + \varepsilon)} + (|K'|^{\lceil d/2 \rceil} + \varepsilon) \frac{\varepsilon}{4((K')^{\lfloor d/2 \rfloor} + \varepsilon)} < \varepsilon.
\end{aligned}$$

This completes the proof of (4.2.4).

The overall result follows by using Proposition 2.3 to show that the multiplication network can be composed with a neural network comprised of the $\Phi^{\mathcal{S}_{\ell,t_i,n}}$ for $i = 1, \dots, d$. Since in no step above the size of the individual networks was dependent on the approximation accuracy, this is also true for the final network. □

Proposition 4.8 shows that we can approximate a single multivariate B-spline with a neural network with a size that is independent of the accuracy. Combining this observation with Theorem 4.3 leads to the following result.

Theorem 4.9. *Let d, n, $k \in \mathbb{N}$ such that $0 < k \leq n$ and $n \geq 2$. Let $q \geq 2$, and let σ be sigmoidal of order q.*

Then there exists C such that for every $f \in C^k([0,1]^d)$ and every $N \in \mathbb{N}$ there exists a neural network Φ^N with activation function σ, $\lceil \log_2(d) \rceil + \lceil \log_q(k-1) \rceil$ layers, and size bounded by CN, such that

$$\left\| f - \Phi^N \right\|_{L^\infty([0,1]^d)} \leq C N^{-\frac{k}{d}} \|f\|_{C^k([0,1]^d)}.$$

Proof. Fix $N \in \mathbb{N}$. By Theorem 4.3, there exist coefficients $|c_i| \leq C\|f\|_{L^\infty([0,1]^d)}$ and $B_i \in \mathcal{B}^n$ for $i = 1, \dots, N$, such that

$$\left\| f - \sum_{i=1}^{N} c_i B_i \right\|_{L^\infty([0,1]^d)} \leq C N^{-\frac{k}{d}} \|f\|_{C^k([0,1]^d)}.$$

Moreover, by Proposition 4.8, for each $i = 1, \dots, N$ exists a neural network Φ^{B_i} with $\lceil \log_2(d) \rceil + \lceil \log_q(k-1) \rceil$ layers, and a fixed size, which approximates B_i on $[-1,1]^d \supseteq [0,1]^d$ up to error of $\varepsilon := N^{-k/d}/N$. The size of Φ^{B_i} is independent of i and N.

By Proposition 2.3, there exists a neural network Φ^N that uniformly approximates $\sum_{i=1}^{N} c_i B_i$ up to error ε on $[0,1]^d$, and has $\lceil \log_2(d) \rceil + \lceil \log_q(k-1) \rceil$ layers. The size of this network is linear in N (see Exercise 4.11). This concludes the proof. □

Theorem 4.9 shows that neural networks with higher-order sigmoidal functions can approximate smooth functions with the same accuracy as spline approximations while having a comparable number of parameters. The network depth is required to behave like $O(\log(k))$ in terms of the smoothness parameter k, cp. Remark 4.4.

Bibliography and further reading

The argument of linking sigmoidal activation functions with spline based approximation was first introduced in [146, 144]. For further details on spline approximation, see [168] or the book [207].

The general strategy of approximating basis functions by neural networks, and then lifting approximation results for those bases has been employed widely in the literature, and will also reappear again in this book. While the following chapters primarily focus on ReLU activation, we highlight a few notable approaches with non-ReLU activations based on the outlined strategy: To approximate analytic functions, [145] emulates a monomial basis. To approximate periodic functions, a basis of trigonometric polynomials is recreated in [147]. Wavelet bases have been emulated in [171]. Moreover, neural networks have been studied through the representation system of ridgelets [30] and ridge functions [103]. A general framework describing the emulation of representation systems to transfer approximation results was presented in [21].

Exercises

Exercise 4.10. Show that (4.2.1) holds.

Exercise 4.11. Let $L \in \mathbb{N}$, $\sigma\colon \mathbb{R} \to \mathbb{R}$, and let Φ_1, Φ_2 be two neural networks with architecture $(\sigma; d_0, d_1^{(1)}, \dots, d_L^{(1)}, d_{L+1})$ and $(\sigma; d_0, d_1^{(2)}, \dots, d_L^{(2)}, d_{L+1})$. Show that $\Phi_1 + \Phi_2$ is a neural network with $\mathrm{size}(\Phi_1 + \Phi_2) \leq \mathrm{size}(\Phi_1) + \mathrm{size}(\Phi_2)$.

Exercise 4.12. Show that, for $\sigma = \sigma_{\mathrm{ReLU}}^2$ and $k \leq 2$, for all $f \in C^k([0,1]^d)$ all weights of the approximating neural network of Theorem 4.9 can be bounded in absolute value by $O(\max\{2, \|f\|_{C^k([0,1]^d)}\})$.

Chapter 5

ReLU neural networks

In this chapter, we discuss feedforward neural networks using the ReLU activation function σ_{ReLU} introduced in Section 2.3. We refer to these functions as ReLU neural networks. Due to its simplicity and the fact that it reduces the vanishing and exploding gradients phenomena, the ReLU is one of the most widely used activation functions in practice.

A key component of the proofs in the previous chapters was the approximation of derivatives of the activation function to emulate polynomials. Since the ReLU is piecewise linear, this trick is not applicable. This makes the analysis fundamentally different from the case of smoother activation functions. Nonetheless, we will see that even this extremely simple activation function yields a very rich class of functions possessing remarkable approximation capabilities.

To formalize these results, we begin this chapter by adopting a framework from [174]. This framework enables the tracking of the number of network parameters for basic manipulations such as adding up or composing two neural networks. This will allow to bound the network complexity, when constructing more elaborate networks from simpler ones. With these preliminaries at hand, the rest of the chapter is dedicated to the exploration of links between ReLU neural networks and the class of "continuous piecewise linear functions." In Section 5.2, we will see that every such function can be exactly represented by a ReLU neural network. Afterwards, in Section 5.3 we will give a more detailed analysis of the required network complexity. Finally, we will use these results to prove a first approximation theorem for ReLU neural networks in Section 5.4. The argument is similar in spirit to Chapter 4, in that we *transfer* established approximation theory for piecewise linear functions to the class of ReLU neural networks of a certain architecture.

5.1 Basic ReLU calculus

The goal of this section is to formalize how to combine and manipulate ReLU neural networks. We have seen an instance of such a result already in Proposition 2.3. Now we want to make this result more precise under the assumption that the activation function is the ReLU. We sharpen Proposition 2.3 by adding bounds on the number of weights that the resulting neural networks have. The following four operations form the basis of all constructions in the sequel.

- *Reproducing an identity:* We have seen in Proposition 3.16 that for most activation functions, an approximation to the identity can be built by neural networks. For ReLUs, we can have an even stronger result and reproduce the identity exactly. This identity will play a crucial

role in order to extend certain neural networks to deeper neural networks, and to facilitate an efficient composition operation.

- *Composition:* We saw in Proposition 2.3 that we can produce a composition of two neural networks and the resulting function is a neural network as well. There we did not study the size of the resulting neural networks. For ReLU activation functions, this composition can be done in a very efficient way leading to a neural network that has up to a constant not more than the number of weights of the two initial neural networks.
- *Parallelization:* Also the parallelization of two neural networks was discussed in Proposition 2.3. We will refine this notion and make precise the size of the resulting neural networks.
- *Linear combinations:* Similarly, for the sum of two neural networks, we will give precise bounds on the size of the resulting neural network.

5.1.1 Identity

We start with expressing the identity on $\mathbb{R}^d$ as a neural network of depth $L \in \mathbb{N}$.

Lemma 5.1 (Identity)**.** *Let $L \in \mathbb{N}$. Then, there exists a ReLU neural network Φ_L^{id} such that $\Phi_L^{\mathrm{id}}(\boldsymbol{x}) = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Moreover,* $\mathrm{depth}(\Phi_L^{\mathrm{id}}) = L$, $\mathrm{width}(\Phi_L^{\mathrm{id}}) = 2d$, *and* $\mathrm{size}(\Phi_L^{\mathrm{id}}) = 2d \cdot (L+1)$.

Proof. Writing $\boldsymbol{I}_d \in \mathbb{R}^{d \times d}$ for the identity matrix, we choose the weights

$$\begin{aligned}&(\boldsymbol{W}^{(0)}, \boldsymbol{b}^{(0)}), \dots, (\boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)}) \\ &\quad := \left(\begin{pmatrix} \boldsymbol{I}_d \\ -\boldsymbol{I}_d \end{pmatrix}, 0 \right), \underbrace{(\boldsymbol{I}_{2d}, 0), \dots, (\boldsymbol{I}_{2d}, 0)}_{L-1 \text{ times}}, ((\boldsymbol{I}_d, -\boldsymbol{I}_d), 0).\end{aligned}$$

Using that $x = \sigma_{\mathrm{ReLU}}(x) - \sigma_{\mathrm{ReLU}}(-x)$ for all $x \in \mathbb{R}$ and $\sigma_{\mathrm{ReLU}}(x) = x$ for all $x \geq 0$ it is obvious that the neural network Φ_L^{id} associated to the weights above satisfies the assertion of the lemma. □

We will see in Exercise 5.23 that the property to exactly represent the identity is not shared by sigmoidal activation functions. It does hold for polynomial activation functions, though.

5.1.2 Composition

Assume we have two neural networks Φ_1, Φ_2 with architectures $(\sigma_{\mathrm{ReLU}}; d_0^1, \dots, d_{L_1+1}^1)$ and $(\sigma_{\mathrm{ReLU}}; d_0^2, \dots, d_{L_1+1}^2)$ respectively. Moreover, we assume that they have weights and biases given by

$$(\boldsymbol{W}_1^{(0)}, \boldsymbol{b}_1^{(0)}), \dots, (\boldsymbol{W}_1^{(L_1)}, \boldsymbol{b}_1^{(L_1)}), \text{ and } (\boldsymbol{W}_2^{(0)}, \boldsymbol{b}_2^{(0)}), \dots, (\boldsymbol{W}_2^{(L_2)}, \boldsymbol{b}_2^{(L_2)}),$$

respectively. If the output dimension $d_{L_1+1}^1$ of Φ_1 equals the input dimension d_0^2 of Φ_2, we can define two types of concatenations: First $\Phi_2 \circ \Phi_1$ is the neural network with weights and biases given by

$$\begin{aligned}&\left(\boldsymbol{W}_1^{(0)}, \boldsymbol{b}_1^{(0)}\right), \dots, \left(\boldsymbol{W}_1^{(L_1-1)}, \boldsymbol{b}_1^{(L_1-1)}\right), \left(\boldsymbol{W}_2^{(0)} \boldsymbol{W}_1^{(L_1)}, \boldsymbol{W}_2^{(0)} \boldsymbol{b}_1^{(L_1)} + \boldsymbol{b}_2^{(0)}\right), \\ &\quad \left(\boldsymbol{W}_2^{(1)}, \boldsymbol{b}_2^{(1)}\right), \dots, \left(\boldsymbol{W}_2^{(L_2)}, \boldsymbol{b}_2^{(L_2)}\right).\end{aligned}$$

Second, $\Phi_2 \bullet \Phi_1$ is the neural network defined as $\Phi_2 \circ \Phi_1^{\mathrm{id}} \circ \Phi_1$. In terms of weighs and biases, $\Phi_2 \bullet \Phi_1$ is given as

$$\begin{aligned}&\left(\boldsymbol{W}_1^{(0)}, \boldsymbol{b}_1^{(0)}\right), \ldots, \left(\boldsymbol{W}_1^{(L_1-1)}, \boldsymbol{b}_1^{(L_1-1)}\right), \left(\begin{pmatrix} \boldsymbol{W}_1^{(L_1)} \\ -\boldsymbol{W}_1^{(L_1)} \end{pmatrix}, \begin{pmatrix} \boldsymbol{b}_1^{(L_1)} \\ -\boldsymbol{b}_1^{(L_1)} \end{pmatrix}\right), \\ &\left(\left(\boldsymbol{W}_2^{(0)}, -\boldsymbol{W}_2^{(0)}\right), \boldsymbol{b}_2^{(0)}\right), \left(\boldsymbol{W}_2^{(1)}, \boldsymbol{b}_2^{(1)}\right), \ldots, \left(\boldsymbol{W}_2^{(L_2)}, \boldsymbol{b}_2^{(L_2)}\right).\end{aligned}$$

The following lemma collects the properties of the construction above.

Lemma 5.2 (Composition)**.** *Let Φ_1, Φ_2 be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^1, \ldots, d_{L_1+1}^1)$ and $(\sigma_{\mathrm{ReLU}}; d_0^2, \ldots, d_{L_2+1}^2)$. Assume $d_{L_1+1}^1 = d_0^2$. Then $\Phi_2 \circ \Phi_1(\boldsymbol{x}) = \Phi_2 \bullet \Phi_1(\boldsymbol{x}) = \Phi_2(\Phi_1(\boldsymbol{x}))$ for all $\boldsymbol{x} \in \mathbb{R}^{d_1^0}$. Moreover,*

$$\begin{aligned}\mathrm{width}(\Phi_2 \circ \Phi_1) &\leq \max\{\mathrm{width}(\Phi_1), \mathrm{width}(\Phi_2)\}, \\ \mathrm{depth}(\Phi_2 \circ \Phi_1) &= \mathrm{depth}(\Phi_1) + \mathrm{depth}(\Phi_2), \\ \mathrm{size}(\Phi_2 \circ \Phi_1) &\leq \mathrm{size}(\Phi_1) + \mathrm{size}(\Phi_2) + (d_{L_1}^1 + 1) d_2^1,\end{aligned}$$

and

$$\begin{aligned}\mathrm{width}(\Phi_2 \bullet \Phi_1) &\leq 2\max\{\mathrm{width}(\Phi_1), \mathrm{width}(\Phi_2)\}, \\ \mathrm{depth}(\Phi_2 \bullet \Phi_1) &= \mathrm{depth}(\Phi_1) + \mathrm{depth}(\Phi_2) + 1, \\ \mathrm{size}(\Phi_2 \bullet \Phi_1) &\leq 2(\mathrm{size}(\Phi_1) + \mathrm{size}(\Phi_2)).\end{aligned}$$

Proof. The fact that $\Phi_2 \circ \Phi_1(\boldsymbol{x}) = \Phi_2 \bullet \Phi_1(\boldsymbol{x}) = \Phi_2(\Phi_1(\boldsymbol{x}))$ for all $\boldsymbol{x} \in \mathbb{R}^{d_0^1}$ follows immediately from the construction. The same can be said for the width and depth bounds. To confirm the size bound, we note that $\boldsymbol{W}_2^{(0)} \boldsymbol{W}_1^{(L_1)} \in \mathbb{R}^{d_1^2 \times d_{L_1}^1}$ and hence $\boldsymbol{W}_2^{(0)} \boldsymbol{W}_1^{(L_1)}$ has not more than $d_1^2 \times d_{L_1}^1$ (nonzero) entries. Moreover, $\boldsymbol{W}_2^{(0)} \boldsymbol{b}_1^{(L_1)} + \boldsymbol{b}_2^{(0)} \in \mathbb{R}^{d_1^2}$. Thus, the L_1-th layer of $\Phi_2 \circ \Phi_1(\boldsymbol{x})$ has at most $d_1^2 \times (1 + d_{L_1}^1)$ entries. The rest is obvious from the construction. □

Interpreting linear transformations as neural networks of depth 0, the previous lemma is also valid in case Φ_1 or Φ_2 is a linear mapping.

5.1.3 Parallelization

Let $(\Phi_i)_{i=1}^m$ be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \ldots, d_{L_i+1}^i)$, respectively. We proceed to build a neural network $(\Phi_1, \ldots, \Phi_m)$ realizing the function

$$\begin{aligned}(\Phi_1, \ldots, \Phi_m) \colon \mathbb{R}^{\sum_{j=1}^m d_0^j} &\to \mathbb{R}^{\sum_{j=1}^m d_{L_j+1}^j} \\ (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) &\mapsto (\Phi_1(\boldsymbol{x}_1), \ldots, \Phi_m(\boldsymbol{x}_m)).\end{aligned} \tag{5.1.1}$$

To do so we first assume $L_1 = \cdots = L_m = L$, and define $(\Phi_1, \dots, \Phi_m)$ via the following sequence of weight-bias tuples:

$$\left(\begin{pmatrix}\boldsymbol{W}_1^{(0)} & & \\ & \ddots & \\ & & \boldsymbol{W}_m^{(0)}\end{pmatrix}, \begin{pmatrix}\boldsymbol{b}_1^{(0)} \\ \vdots \\ \boldsymbol{b}_m^{(0)}\end{pmatrix}\right), \dots, \left(\begin{pmatrix}\boldsymbol{W}_1^{(L)} & & \\ & \ddots & \\ & & \boldsymbol{W}_m^{(L)}\end{pmatrix}, \begin{pmatrix}\boldsymbol{b}_1^{(L)} \\ \vdots \\ \boldsymbol{b}_m^{(L)}\end{pmatrix}\right) \tag{5.1.2}$$

where these matrices are understood as block-diagonal filled up with zeros. For the general case where the Φ_j might have different depths, let $L_{\max} := \max_{1 \leq i \leq m} L_i$ and $I := \{1 \leq i \leq m \,|\, L_i < L_{\max}\}$. For $j \in I^c$ set $\widetilde{\Phi}_j := \Phi_j$, and for each $j \in I$

$$\widetilde{\Phi}_j := \Phi^{\mathrm{id}}_{L_{\max} - L_j} \circ \Phi_j. \tag{5.1.3}$$

Finally,

$$(\Phi_1, \dots, \Phi_m) := (\widetilde{\Phi}_1, \dots, \widetilde{\Phi}_m). \tag{5.1.4}$$

We collect the properties of the parallelization in the lemma below.

Lemma 5.3 (Parallelization)**.** *Let $m \in \mathbb{N}$ and $(\Phi_i)_{i=1}^m$ be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \dots, d_{L_i+1}^i)$, respectively. Then the neural network $(\Phi_1, \dots, \Phi_m)$ satisfies*

$$(\Phi_1, \dots, \Phi_m)(\boldsymbol{x}) = (\Phi_1(\boldsymbol{x}_1), \dots, \Phi_m(\boldsymbol{x}_m)) \text{ for all } \boldsymbol{x} \in \mathbb{R}^{\sum_{j=1}^m d_0^j}.$$

Moreover, with $L_{\max} := \max_{j \leq m} L_j$ it holds that

$$\mathrm{width}((\Phi_1, \dots, \Phi_m)) \leq 2 \sum_{j=1}^m \mathrm{width}(\Phi_j), \tag{5.1.5a}$$

$$\mathrm{depth}((\Phi_1, \dots, \Phi_m)) = \max_{j \leq m} \mathrm{depth}(\Phi_j), \tag{5.1.5b}$$

$$\mathrm{size}((\Phi_1, \dots, \Phi_m)) \leq 2 \sum_{j=1}^m \mathrm{size}(\Phi_j) + 2 \sum_{j=1}^m (L_{\max} - L_j) d_{L_j+1}^j. \tag{5.1.5c}$$

Proof. All statements except for the bound on the size follow immediately from the construction. To obtain the bound on the size, we note that by construction the sizes of the $(\widetilde{\Phi}_i)_{i=1}^m$ in (5.1.3) will simply be added. The size of each $\widetilde{\Phi}_i$ can be bounded with Lemma 5.2. □

If all input dimensions $d_0^1 = \cdots = d_0^m =: d_0$ are the same, we will also use **parallelization with shared inputs** to realize the function $\boldsymbol{x} \mapsto (\Phi_1(\boldsymbol{x}), \dots, \Phi_m(\boldsymbol{x}))$ from $\mathbb{R}^{d_0} \to \mathbb{R}^{d^1_{L_1+1} + \cdots + d^m_{L_m+1}}$. In terms of the construction (5.1.2), the only required change is that the block-diagonal matrix $\mathrm{diag}(\boldsymbol{W}_1^{(0)}, \dots, \boldsymbol{W}_m^{(0)})$ becomes the matrix in $\mathbb{R}^{\sum_{j=1}^m d_1^j \times d_0^1}$ which stacks $\boldsymbol{W}_1^{(0)}, \dots, \boldsymbol{W}_m^{(0)}$ on top of each other. Similarly, we will allow Φ_j to only take some of the entries of $\boldsymbol{x}$ as input. For parallelization with shared inputs we will use the same notation $(\Phi_j)_{j=1}^m$ as before, where the precise meaning will always be clear from context. Note that Lemma 5.3 remains valid in this case.

5.1.4 Linear combinations

Let $m \in \mathbb{N}$ and let $(\Phi_i)_{i=1}^m$ be ReLU neural networks that have architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \dots, d_{L_i+1}^i)$, respectively. Assume that $d_{L_1+1}^1 = \cdots = d_{L_m+1}^m$, i.e., all $\Phi_1, \dots, \Phi_m$ have the same output dimension. For scalars $\alpha_j \in \mathbb{R}$, we wish to construct a ReLU neural network $\sum_{j=1}^m \alpha_j \Phi_j$ realizing the function

$$\begin{cases} \mathbb{R}^{\sum_{j=1}^m d_0^j} \to \mathbb{R}^{d_{L_1+1}^1} \\ (\boldsymbol{x}_1, \dots, \boldsymbol{x}_m) \mapsto \sum_{j=1}^m \alpha_j \Phi_j(\boldsymbol{x}_j). \end{cases}$$

This corresponds to the parallelization $(\Phi_1, \dots, \Phi_m)$ composed with the linear transformation $(\boldsymbol{z}_1, \dots, \boldsymbol{z}_m) \mapsto \sum_{j=1}^m \alpha_j \boldsymbol{z}_j$. The following result holds.

Lemma 5.4 (Linear combinations)**.** *Let $m \in \mathbb{N}$ and $(\Phi_i)_{i=1}^m$ be neural networks with architectures $(\sigma_{\mathrm{ReLU}}; d_0^i, \dots, d_{L_i+1}^i)$, respectively. Assume that $d_{L_1+1}^1 = \cdots = d_{L_m+1}^m$, let $\alpha \in \mathbb{R}^m$ and set $L_{\max} := \max_{j \le m} L_j$. Then, there exists a neural network $\sum_{j=1}^m \alpha_j \Phi_j$ such that $(\sum_{j=1}^m \alpha_j \Phi_j)(\boldsymbol{x}) = \sum_{j=1}^m \alpha_j \Phi_j(\boldsymbol{x}_j)$ for all $\boldsymbol{x} = (\boldsymbol{x}_j)_{j=1}^m \in \mathbb{R}^{\sum_{j=1}^m d_0^j}$. Moreover,*

$$\operatorname{width}\left(\sum_{j=1}^m \alpha_j \Phi_j\right) \le 2 \sum_{j=1}^m \operatorname{width}(\Phi_j), \tag{5.1.6a}$$

$$\operatorname{depth}\left(\sum_{j=1}^m \alpha_j \Phi_j\right) = \max_{j \le m} \operatorname{depth}(\Phi_j), \tag{5.1.6b}$$

$$\operatorname{size}\left(\sum_{j=1}^m \alpha_j \Phi_j\right) \le 2 \sum_{j=1}^m \operatorname{size}(\Phi_j) + 2 \sum_{j=1}^m (L_{\max} - L_j) d_{L_j+1}^j. \tag{5.1.6c}$$

Proof. The construction of $\sum_{j=1}^m \alpha_j \Phi_j$ is analogous to that of $(\Phi_1, \dots, \Phi_m)$, i.e., we first define the linear combination of neural networks with the same depth. Then the weights are chosen as in (5.1.2), but with the last linear transformation replaced by

$$\left((\alpha_1 \boldsymbol{W}_1^{(L)} \cdots \alpha_m \boldsymbol{W}_m^{(L)}), \sum_{j=1}^m \alpha_j \boldsymbol{b}_j^{(L)} \right).$$

For general depths, we define the sum of the neural networks to be the sum of the extended neural networks $\widetilde{\Phi}_i$ as of (5.1.3). All statements of the lemma follow immediately from this construction. □

In case $d_0^1 = \cdots = d_0^m =: d_0$ (all neural networks have the same input dimension), we will also consider **linear combinations with shared inputs**, i.e., a neural network realizing

$$\boldsymbol{x} \mapsto \sum_{j=1}^m \alpha_j \Phi_j(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in \mathbb{R}^{d_0}.$$

This requires the same minor adjustment as discussed at the end of Section 5.1.3. Lemma 5.4 remains valid in this case and again we do not distinguish in notation for linear combinations with or without shared inputs.

5.2 Continuous piecewise linear functions

In this section, we will relate ReLU neural networks to a large class of functions. We first formally introduce the set of continuous piecewise linear functions from a set $\Omega \subseteq \mathbb{R}^d$ to $\mathbb{R}$. Note that we admit in particular $\Omega = \mathbb{R}^d$ in the following definition.

Definition 5.5. Let $\Omega \subseteq \mathbb{R}^d$, $d \in \mathbb{N}$. We call a function $f : \Omega \to \mathbb{R}$ **continuous, piecewise linear (cpwl)** if $f \in C^0(\Omega)$ and there exist $n \in \mathbb{N}$ affine functions $g_j \colon \mathbb{R}^d \to \mathbb{R}$, $g_j(\boldsymbol{x}) = \boldsymbol{w}_j^\top \boldsymbol{x} + b_j$ such that for each $\boldsymbol{x} \in \Omega$ it holds that $f(\boldsymbol{x}) = g_j(\boldsymbol{x})$ for at least one $j \in \{1, \dots, n\}$. For $m > 1$ we call $f : \Omega \to \mathbb{R}^m$ cpwl if and only if each component of f is cpwl.

Remark 5.6. A "continuous piecewise linear function" as in Definition 5.5 is actually piecewise *affine*. To maintain consistency with the literature, we use the terminology cpwl.

In the following, we will refer to the connected domains on which f is equal to one of the functions g_j, also as **regions** or **pieces**. If f is cpwl with $q \in \mathbb{N}$ regions, then with $n \in \mathbb{N}$ denoting the number of affine functions it holds $n \leq q$.

Note that, the mapping $\boldsymbol{x} \mapsto \sigma_{\mathrm{ReLU}}(\boldsymbol{w}^\top \boldsymbol{x} + b)$, which is a ReLU neural network with a single neuron, is cpwl (with two regions). Consequently, every ReLU neural network is a repeated composition of linear combinations of cpwl functions. It is not hard to see that the set of cpwl functions is closed under compositions and linear combinations. Hence, *every ReLU neural network is a cpwl function.* Interestingly, the reverse direction of this statement is also true, meaning that *every cpwl function can be represented by a ReLU neural network* as we shall demonstrate below. Therefore, we can identify the class of functions realized by arbitrary ReLU neural networks as the class of cpwl functions.

Theorem 5.7. *Let $d \in \mathbb{N}$, let $\Omega \subseteq \mathbb{R}^d$ be convex, and let $f : \Omega \to \mathbb{R}$ be cpwl with $n \in \mathbb{N}$ as in Definition 5.5. Then, there exists a ReLU neural network Φ^f such that $\Phi^f(\boldsymbol{x}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$ and*

$$\mathrm{size}(\Phi^f) = O(dn2^n), \quad \mathrm{width}(\Phi^f) = O(dn2^n), \quad \mathrm{depth}(\Phi^f) = O(n).$$

A statement similar to Theorem 5.7 can be found in [4, 85]. There, the authors give a construction with a depth that behaves logarithmic in d and is independent of n, but with significantly larger bounds on the size. As we shall see, the proof of Theorem 5.7 is a simple consequence of the following well-known result from [225]; also see [169, 237]. It states that every cpwl function can be expressed as a finite maximum of a finite minimum of certain affine functions.

Proposition 5.8. *Let $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$ be convex, and let $f : \Omega \to \mathbb{R}$ be cpwl with $n \in \mathbb{N}$ affine functions as in Definition 5.5. Then there exists $m \in \mathbb{N}$ and sets $s_j \subseteq \{1, \dots, n\}$ for $j \in \{1, \dots, m\}$, such that*

$$f(\boldsymbol{x}) = \max_{1 \le j \le m} \min_{i \in s_j} (g_i(\boldsymbol{x})) \qquad \text{for all } \boldsymbol{x} \in \Omega. \tag{5.2.1}$$

Proof. **Step 1.** We start with $d = 1$, i.e., $\Omega \subseteq \mathbb{R}$ is a (possibly unbounded) interval and for each $x \in \Omega$ there exists $j \in \{1, \dots, n\}$ such that with $g_j(x) \coloneqq w_j x + b_j$ it holds that $f(x) = g_j(x)$. Without loss of generality, we can assume that $g_i \neq g_j$ for all $i \neq j$. Since the graphs of the g_j are lines, they intersect at (at most) finitely many points in Ω.

Since f is continuous, we conclude that there exist finitely many intervals covering Ω, such that f coincides with one of the g_j on each interval. For each $x \in \Omega$ let

$$s_x \coloneqq \{1 \le j \le n \,|\, g_j(x) \ge f(x)\}$$

and

$$f_x(y) \coloneqq \min_{j \in s_x} g_j(y) \qquad \text{for all } y \in \Omega.$$

Clearly, $f_x(x) = f(x)$. We claim that, additionally,

$$f_x(y) \le f(y) \qquad \text{for all } y \in \Omega. \tag{5.2.2}$$

This then shows that

$$f(y) = \max_{x \in \Omega} f_x(y) = \max_{x \in \Omega} \min_{j \in s_x} g_j(y) \qquad \text{for all } y \in \mathbb{R}.$$

Since there exist only finitely many possibilities to choose a subset of $\{1, \dots, n\}$, we conclude that (5.2.1) holds for $d = 1$.

It remains to verify the claim (5.2.2). Fix $y \neq x \in \Omega$. Without loss of generality, let $x < y$ and let $x = x_0 < \cdots < x_k = y$ be such that $f|_{[x_{i-1}, x_i]}$ equals some g_j for each $i \in \{1, \dots, k\}$. In order to show (5.2.2), it suffices to prove that there exists at least one j such that $g_j(x_0) \ge f(x_0)$ and $g_j(x_k) \le f(x_k)$. The claim is trivial for $k = 1$. We proceed by induction. Suppose the claim holds for $k - 1$, and consider the partition $x_0 < \cdots < x_k$. Let $r \in \{1, \dots, n\}$ be such that $f|_{[x_0, x_1]} = g_r|_{[x_0, x_1]}$. Applying the induction hypothesis to the interval $[x_1, x_k]$, we can find $j \in \{1, \dots, n\}$ such that $g_j(x_1) \ge f(x_1)$ and $g_j(x_k) \le f(x_k)$. If $g_j(x_0) \ge f(x_0)$, then g_j is the desired function. Otherwise, $g_j(x_0) < f(x_0)$. Then $g_r(x_0) = f(x_0) > g_j(x_0)$ and $g_r(x_1) = f(x_1) \le g_j(x_1)$. Therefore $g_r(x) \le g_j(x)$ for all $x \ge x_1$, and in particular $g_r(x_k) \le g_j(x_k)$. Thus g_r is the desired function.

Step 2. For general $d \in \mathbb{N}$, let $g_j(\boldsymbol{x}) \coloneqq \boldsymbol{w}_j^\top \boldsymbol{x} + b_j$ for $j = 1, \dots, n$. For each $\boldsymbol{x} \in \Omega$, let

$$s_{\boldsymbol{x}} \coloneqq \{1 \le j \le n \,|\, g_j(\boldsymbol{x}) \ge f(\boldsymbol{x})\}$$

and for all $\boldsymbol{y} \in \Omega$, let

$$f_{\boldsymbol{x}}(\boldsymbol{y}) := \min_{j \in s_{\boldsymbol{x}}} g_j(\boldsymbol{y}).$$

For an arbitrary 1-dimensional affine subspace $S \subseteq \mathbb{R}^d$ passing through $\boldsymbol{x}$ consider the line (segment) $I := S \cap \Omega$, which is connected since Ω is convex. By Step 1, it holds

$$f(\boldsymbol{y}) = \max_{\boldsymbol{x} \in \Omega} f_{\boldsymbol{x}}(\boldsymbol{y}) = \max_{\boldsymbol{x} \in \Omega} \min_{j \in s_{\boldsymbol{x}}} g_j(\boldsymbol{y})$$

on all of I. Since I was arbitrary the formula is valid for all $\boldsymbol{y} \in \Omega$. This again implies (5.2.1) as in Step 1. □

Remark 5.9. Using $\min(a, b) = -\max(-a, -b)$, there exists $\tilde{m} \in \mathbb{N}$ and sets $\tilde{s}_j \subseteq \{1, \dots, n\}$ for $j = 1, \dots, \tilde{m}$, such that for all $\boldsymbol{x} \in \mathbb{R}$

$$\begin{aligned} f(\boldsymbol{x}) = -(-f(\boldsymbol{x})) &= -\min_{1 \le j \le \tilde{m}} \max_{i \in \tilde{s}_j} (-\boldsymbol{w}_i^\top \boldsymbol{x} - b_i) \\ &= \max_{1 \le j \le \tilde{m}} (-\max_{i \in \tilde{s}_j} (-\boldsymbol{w}_i^\top \boldsymbol{x} - b_i)) \\ &= \max_{1 \le j \le \tilde{m}} (\min_{i \in \tilde{s}_j} (\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)). \end{aligned}$$

To prove Theorem 5.7, it therefore suffices to show that the minimum and the maximum are expressible by ReLU neural networks.

Lemma 5.10. *For every x, $y \in \mathbb{R}$ it holds that*

$$\min\{x, y\} = \sigma_{\mathrm{ReLU}}(y) - \sigma_{\mathrm{ReLU}}(-y) - \sigma_{\mathrm{ReLU}}(y - x) \in \mathcal{N}_2^1(\sigma_{\mathrm{ReLU}}; 1, 3)$$

and

$$\max\{x, y\} = \sigma_{\mathrm{ReLU}}(y) - \sigma_{\mathrm{ReLU}}(-y) + \sigma_{\mathrm{ReLU}}(x - y) \in \mathcal{N}_2^1(\sigma_{\mathrm{ReLU}}; 1, 3).$$

Proof. We have

$$\begin{aligned} \max\{x, y\} &= y + \begin{cases} 0 & \text{if } y > x \\ x - y & \text{if } x \ge y \end{cases} \\ &= y + \sigma_{\mathrm{ReLU}}(x - y). \end{aligned}$$

Using $y = \sigma_{\mathrm{ReLU}}(y) - \sigma_{\mathrm{ReLU}}(-y)$, the claim for the maximum follows. For the minimum observe that $\min\{x, y\} = -\max\{-x, -y\}$. □

The minimum of $n \ge 2$ inputs can be computed by repeatedly applying the construction of Lemma 5.10. The resulting neural network is described in the next lemma.

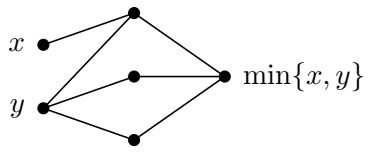


Figure 5.1: Sketch of the neural network in Lemma 5.10. Only edges with non-zero weights are drawn.

Lemma 5.11. *For every $n \geq 2$ there exists a neural network $\Phi_n^{\min} : \mathbb{R}^n \to \mathbb{R}$ with*

$$\mathrm{size}(\Phi_n^{\min}) \leq 16n, \qquad \mathrm{width}(\Phi_n^{\min}) \leq 3n, \qquad \mathrm{depth}(\Phi_n^{\min}) \leq \lceil \log_2(n) \rceil$$

such that $\Phi_n^{\min}(x_1, \ldots, x_n) = \min_{1 \leq j \leq n} x_j$. Similarly, there exists a neural network $\Phi_n^{\max} : \mathbb{R}^n \to \mathbb{R}$ realizing the maximum and satisfying the same complexity bounds.

Proof. Throughout denote by $\Phi_2^{\min} : \mathbb{R}^2 \to \mathbb{R}$ the neural network from Lemma 5.10. It is of depth 1 and size 7 (since all biases are zero, it suffices to count the number of connections in Figure 5.1).

Step 1. Consider first the case where $n = 2^k$ for some $k \in \mathbb{N}$. We proceed by induction of k. For $k = 1$ the claim is proven. For $k \geq 2$ set

$$\Phi_{2^k}^{\min} := \Phi_2^{\min} \circ (\Phi_{2^{k-1}}^{\min}, \Phi_{2^{k-1}}^{\min}). \tag{5.2.3}$$

By Lemma 5.2 and Lemma 5.3 we have

$$\mathrm{depth}(\Phi_{2^k}^{\min}) \leq \mathrm{depth}(\Phi_2^{\min}) + \mathrm{depth}(\Phi_{2^{k-1}}^{\min}) \leq \cdots \leq k.$$

Next, we bound the size of the neural network. Note that all biases in this neural network are set to 0, since the $\Phi_2^{\min}$ neural network in Lemma 5.10 has no biases. Thus, the size of the neural network $\Phi_{2^k}^{\min}$ corresponds to the number of connections in the graph (the number of nonzero weights). Careful inspection of the neural network architecture, see Figure 5.2, reveals that

$$\begin{aligned} \mathrm{size}(\Phi_{2^k}^{\min}) &= 4 \cdot 2^{k-1} + \sum_{j=0}^{k-2} 12 \cdot 2^j + 3 \\ &= 2n + 12 \cdot (2^{k-1} - 1) + 3 = 2n + 6n - 9 \leq 8n, \end{aligned}$$

and that $\mathrm{width}(\Phi_{2^k}^{\min}) \leq (3/2)2^k$. This concludes the proof for the case $n = 2^k$.

Step 2. For the general case, we first let

$$\Phi_1^{\min}(x) := x \qquad \text{for all } x \in \mathbb{R}$$

be the identity on $\mathbb{R}$, i.e. a linear transformation and thus formally a depth 0 neural network. Then, for all $n \geq 2$

$$\Phi_n^{\min} := \Phi_2^{\min} \circ \begin{cases} (\Phi_1^{\mathrm{id}} \circ \Phi_{\lfloor \frac{n}{2} \rfloor}^{\min}, \Phi_{\lceil \frac{n}{2} \rceil}^{\min}) & \text{if } n \in \{2^k + 1 \,|\, k \in \mathbb{N}\} \\ (\Phi_{\lfloor \frac{n}{2} \rfloor}^{\min}, \Phi_{\lceil \frac{n}{2} \rceil}^{\min}) & \text{otherwise.} \end{cases} \tag{5.2.4}$$

This definition extends (5.2.3) to arbitrary $n \geq 2$, since the first case in (5.2.4) never occurs if $n \geq 2$ is a power of two.

To analyze (5.2.4), we start with the depth and claim that

$$\operatorname{depth}(\Phi_n^{\min}) = k \qquad \text{for all } 2^{k-1} < n \leq 2^k$$

and all $k \in \mathbb{N}$. We proceed by induction over k. The case $k = 1$ is clear. For the induction step, assume the statement holds for some fixed $k \in \mathbb{N}$ and fix an integer n with $2^k < n \leq 2^{k+1}$. Then

$$\left\lceil \frac{n}{2} \right\rceil \in (2^{k-1}, 2^k] \cap \mathbb{N}$$

and

$$\left\lfloor \frac{n}{2} \right\rfloor \in \begin{cases} \{2^{k-1}\} & \text{if } n = 2^k + 1 \\ (2^{k-1}, 2^k] \cap \mathbb{N} & \text{otherwise.} \end{cases}$$

Using the induction assumption, (5.2.4) and Lemmas 5.1 and 5.2, this shows

$$\operatorname{depth}(\Phi_n^{\min}) = \operatorname{depth}(\Phi_2^{\min}) + k = 1 + k,$$

and proves the claim.

For the size and width bounds, we only sketch the argument: Fix $n \in \mathbb{N}$ such that $2^{k-1} < n \leq 2^k$. Then $\Phi_n^{\min}$ is constructed from at most as many subnetworks as $\Phi_{2^k}^{\min}$, but with some $\Phi_2^{\min} : \mathbb{R}^2 \to \mathbb{R}$ blocks replaced by $\Phi_1^{\mathrm{id}} : \mathbb{R} \to \mathbb{R}$, see Figure 5.3. Since Φ_1^{id} has the same depth as $\Phi_2^{\min}$, but is smaller in width and number of connections, the width and size of $\Phi_n^{\min}$ is bounded by the width and size of $\Phi_{2^k}^{\min}$. Due to $2^k \leq 2n$, the bounds from Step 1 give the bounds stated in the lemma.

Step 3. For the maximum, define

$$\Phi_n^{\max}(x_1, \dots, x_n) := -\Phi_n^{\min}(-x_1, \dots, -x_n).$$

□

of Theorem 5.7. By Proposition 5.8 the neural network

$$\Phi := \Phi_m^{\max} \bullet (\Phi_{|s_j|}^{\min})_{j=1}^m \bullet ((\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)_{i \in s_j})_{j=1}^m$$

realizes the function f.

Since the number of possibilities to choose subsets of $\{1, \dots, n\}$ equals 2^n we have $m \leq 2^n$. Since each s_j is a subset of $\{1, \dots, n\}$, the cardinality $|s_j|$ of s_j is bounded by n. By Lemma 5.2, Lemma 5.3, and Lemma 5.11

$$\begin{aligned} \operatorname{depth}(\Phi) &\leq 2 + \operatorname{depth}(\Phi_m^{\max}) + \max_{1 \leq j \leq n} \operatorname{depth}(\Phi_{|s_j|}^{\min}) \\ &\leq 1 + \lceil \log_2(2^n) \rceil + \lceil \log_2(n) \rceil = O(n) \end{aligned}$$

and

$$\begin{aligned} \operatorname{width}(\Phi) &\leq 2 \max \left\{ \operatorname{width}(\Phi_m^{\max}), \sum_{j=1}^m \operatorname{width}(\Phi_{|s_j|}^{\min}), \sum_{j=1}^m \operatorname{width}((\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)_{i \in s_j})) \right\} \\ &\leq 2 \max\{3m, 3mn, mdn\} = O(dn2^n) \end{aligned}$$

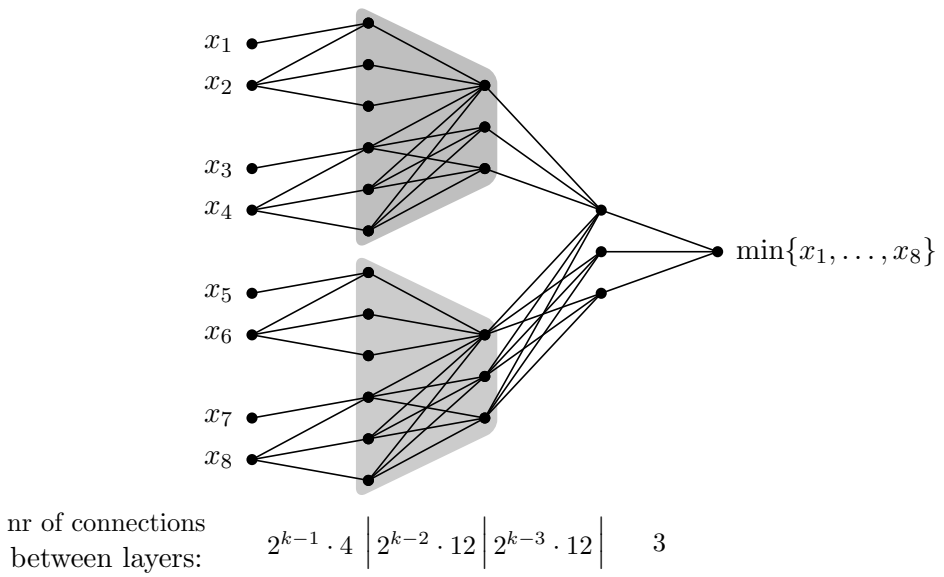


Figure 5.2: Architecture of the $\Phi^{\min}_{2^k}$ neural network in Step 1 of the proof of Lemma 5.11 and the number of connections in each layer for $k = 3$. Each grey box corresponds to 12 connections in the graph.

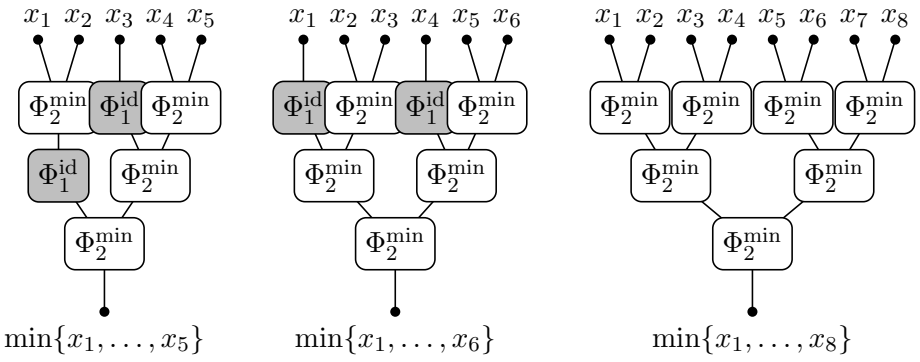


Figure 5.3: Construction of $\Phi^{\min}_n$ for general n in Step 2 of the proof of Lemma 5.11.

and

$$\begin{aligned}\operatorname{size}(\Phi) &\leq 4\Big(\operatorname{size}(\Phi_m^{\max}) + \operatorname{size}((\Phi_{|s_j|}^{\min})_{j=1}^m) + \operatorname{size}((\boldsymbol{w}_i^\top \boldsymbol{x} + b_i)_{i\in s_j})_{j=1}^m)\Big) \\ &\leq 4\left(16m + 2\sum_{j=1}^m (16|s_j| + 2\lceil \log_2(n)\rceil) + nm(d+1)\right) = O(dn2^n).\end{aligned}$$

This concludes the proof. □

5.3 Simplicial pieces

This section studies the case, were we do not have arbitrary cpwl functions, but where the regions on which f is affine are simplices. Under this condition, we can construct neural networks that scale merely *linearly* in the number of such regions, which is a serious improvement from the *exponential* dependence of the size on the number of regions that was found in Theorem 5.7.

5.3.1 Triangulations of Ω

For the ensuing discussion, we will consider $\Omega \subseteq \mathbb{R}^d$ to be partitioned into simplices. This partitioning will be termed a **triangulation** of Ω. Other notions prevalent in the literature include a **tessellation** of Ω, or a **simplicial mesh** on Ω. To give a precise definition, let us first recall some terminology. For a set $S \subseteq \mathbb{R}^d$ we denote the **convex hull** of S by

$$\operatorname{co}(S) := \left\{ \sum_{j=1}^n \alpha_j \boldsymbol{x}_j \;\middle|\; n \in \mathbb{N},\ \boldsymbol{x}_j \in S, \alpha_j \geq 0,\ \sum_{j=1}^n \alpha_j = 1 \right\}. \tag{5.3.1}$$

An n**-simplex** is the convex hull of $n \in \mathbb{N}$ points that are independent in a specific sense. This is made precise in the following definition.

Definition 5.12. Let $n \in \mathbb{N}_0$, $d \in \mathbb{N}$ and $n \leq d$. We call $\boldsymbol{x}_0, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$ **affinely independent** if and only if either $n = 0$ or $n \geq 1$ and the vectors $\boldsymbol{x}_1 - \boldsymbol{x}_0, \dots, \boldsymbol{x}_n - \boldsymbol{x}_0$ are linearly independent. In this case, we call $\operatorname{co}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_n) := \operatorname{co}(\{\boldsymbol{x}_0, \dots, \boldsymbol{x}_n\})$ an n**-simplex**.

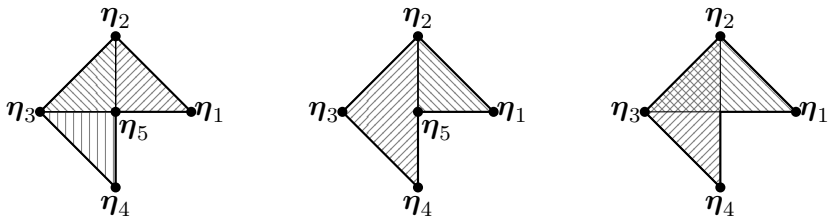
As mentioned before, a triangulation refers to a partition of a space into simplices. We give a formal definition below.

Definition 5.13. Let $d \in \mathbb{N}$, and $\Omega \subseteq \mathbb{R}^d$ be compact. Let $\mathcal{T}$ be a finite set of d-simplices, and for each $\tau \in \mathcal{T}$ let $V(\tau) \subseteq \Omega$ have cardinality $d+1$ such that $\tau = \operatorname{co}(V(\tau))$. We call $\mathcal{T}$ a **regular triangulation** of Ω, if and only if

(i) $\bigcup_{\tau\in\mathcal{T}} \tau = \Omega$,

(ii) for all $\tau, \tau' \in \mathcal{T}$ it holds that $\tau \cap \tau' = \operatorname{co}(V(\tau) \cap V(\tau'))$.

We call $\boldsymbol{\eta} \in \mathcal{V} := \bigcup_{\tau\in\mathcal{T}} V(\tau)$ a **node** (or vertex) and $\tau \in \mathcal{T}$ an **element** of the triangulation.



$\tau_1 = \mathrm{co}(\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \boldsymbol{\eta}_5)$
$\tau_2 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_3, \boldsymbol{\eta}_5)$
$\tau_3 = \mathrm{co}(\boldsymbol{\eta}_3, \boldsymbol{\eta}_4, \boldsymbol{\eta}_5)$

$\tau_1 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_3, \boldsymbol{\eta}_4)$
$\tau_2 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_5, \boldsymbol{\eta}_1)$

$\tau_1 = \mathrm{co}(\boldsymbol{\eta}_2, \boldsymbol{\eta}_3, \boldsymbol{\eta}_4)$
$\tau_2 = \mathrm{co}(\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \boldsymbol{\eta}_3)$

Figure 5.4: The first is a regular triangulation, while the second and the third are not.

For a regular triangulation $\mathcal{T}$ with nodes $\mathcal{V}$ we also introduce the constant

$$k_{\mathcal{T}} := \max_{\boldsymbol{\eta} \in \mathcal{V}} |\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}| \tag{5.3.2}$$

corresponding to the maximal number of elements shared by a single node.

5.3.2 Size bounds for regular triangulations

Throughout this subsection, let $\mathcal{T}$ be a regular triangulation of Ω, and we adhere to the notation of Definition 5.13. We will say that $f : \Omega \to \mathbb{R}$ is cpwl with respect to $\mathcal{T}$ if f is cpwl and $f|_\tau$ is affine for each $\tau \in \mathcal{T}$. The rest of this subsection is dedicated to proving the following result. It was first shown in [136] with a more technical argument, and extends an earlier statement from [85] to general triangulations (also see Section 5.3.3).

Theorem 5.14. *Let $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$ be a bounded domain, and let $\mathcal{T}$ be a regular triangulation of Ω. Let $f : \Omega \to \mathbb{R}$ be cpwl with respect to $\mathcal{T}$ and $f|_{\partial\Omega} = 0$. Then there exists a ReLU neural network $\Phi : \Omega \to \mathbb{R}$ realizing f, and it holds*

$$\mathrm{size}(\Phi) = O(|\mathcal{T}|), \qquad \mathrm{width}(\Phi) = O(|\mathcal{T}|), \qquad \mathrm{depth}(\Phi) = O(1), \tag{5.3.3}$$

where the constants in the Landau notation depend on d and $k_{\mathcal{T}}$ in (5.3.2).

We will split the proof into several lemmata. The strategy is to introduce a basis of the space of cpwl functions on $\mathcal{T}$ the elements of which vanish on the boundary of Ω. We will then show that there exist $O(|\mathcal{T}|)$ basis functions, each of which can be represented with a neural network the size of which depends only on $k_{\mathcal{T}}$ and d. To construct this basis, we first point out that an affine function on a simplex is uniquely defined by its values at the nodes.

Lemma 5.15. *Let $d \in \mathbb{N}$. Let $\tau := \mathrm{co}(\boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d)$ be a d-simplex. For every $y_0, \dots, y_d \in \mathbb{R}$, there exists a unique $g \in \mathbb{P}_1(\mathbb{R}^d)$ such that $g(\boldsymbol{\eta}_i) = y_i$, $i = 0, \dots, d$.*

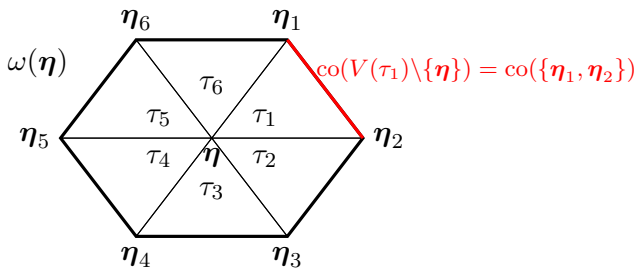


Figure 5.5: Visualization of Lemma 5.16 in two dimensions. The patch $\omega(\boldsymbol{\eta})$ consists of the union of all 2-simplices τ_i containing $\boldsymbol{\eta}$. Its boundary consists of the union of all 1-simplices made up by the nodes of each τ_i without the center node, i.e., the convex hulls of $V(\tau_i)\backslash\{\boldsymbol{\eta}\}$.

Proof. Since $\boldsymbol{\eta}_1 - \boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d - \boldsymbol{\eta}_0$ is a basis of $\mathbb{R}^d$, there is a unique $\boldsymbol{w} \in \mathbb{R}^d$ such that $\boldsymbol{w}^\top(\boldsymbol{\eta}_i - \boldsymbol{\eta}_0) = y_i - y_0$ for $i = 1, \dots, d$. Then $g(\boldsymbol{x}) := \boldsymbol{w}^\top \boldsymbol{x} + (y_0 - \boldsymbol{w}^\top \boldsymbol{\eta}_0)$ is as desired. Moreover, for every $g \in \mathbb{P}_1$ it holds that $g(\sum_{i=0}^d \alpha_i \boldsymbol{\eta}_i) = \sum_{i=0}^d \alpha_i g(\boldsymbol{\eta}_i)$ whenever $\sum_{i=0}^d \alpha_i = 1$ (this is in general not true if the coefficients do not sum to 1). Hence, g is uniquely determined by its values at the nodes. □

Since Ω is the union of the simplices $\tau \in \mathcal{T}$, every cpwl function with respect to $\mathcal{T}$ is thus uniquely defined through its values at the nodes. Hence, the desired basis consists of cpwl functions $\varphi_{\boldsymbol{\eta}} : \Omega \to \mathbb{R}$ with respect to $\mathcal{T}$ such that

$$\varphi_{\boldsymbol{\eta}}(\boldsymbol{\mu}) = \delta_{\boldsymbol{\eta}\boldsymbol{\mu}} \qquad \text{for all } \boldsymbol{\mu} \in \mathcal{V}, \tag{5.3.4}$$

where $\delta_{\boldsymbol{\eta}\boldsymbol{\mu}}$ denotes the Kronecker delta. Assuming $\varphi_{\boldsymbol{\eta}}$ to be well-defined for the moment, we can then represent every cpwl function $f : \Omega \to \mathbb{R}$ that vanishes on the boundary $\partial\Omega$ as

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}} f(\boldsymbol{\eta}) \varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \Omega.$$

Note that it suffices to sum over the set of **interior nodes** $\mathcal{V} \cap \mathring{\Omega}$, since $f(\boldsymbol{\eta}) = 0$ whenever $\boldsymbol{\eta} \in \partial\Omega$. To formally verify existence and well-definedness of $\varphi_{\boldsymbol{\eta}}$, we first need a lemma characterizing the boundary of so-called "patches" of the triangulation: For each $\boldsymbol{\eta} \in \mathcal{V}$, we introduce the **patch** $\omega(\boldsymbol{\eta})$ of the node $\boldsymbol{\eta}$ as the union of all elements containing $\boldsymbol{\eta}$, i.e.,

$$\omega(\boldsymbol{\eta}) := \bigcup_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} \tau. \tag{5.3.5}$$

Lemma 5.16. *Let $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$ be an interior node. Then,*

$$\partial\omega(\boldsymbol{\eta}) = \bigcup_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} \mathrm{co}(V(\tau)\backslash\{\boldsymbol{\eta}\}).$$

We refer to Figure 5.5 for a visualization of Lemma 5.16. The proof of Lemma 5.16 is quite technical but nonetheless elementary. We therefore only outline the general argument but leave the details to the reader in Excercise 5.27: The boundary of $\omega(\boldsymbol{\eta})$ must be contained in the union of the boundaries of all τ in the patch $\omega(\boldsymbol{\eta})$. Since $\boldsymbol{\eta}$ is an interior point of Ω, it must also be an interior point of $\omega(\boldsymbol{\eta})$. This can be used to show that for every $S := \{\boldsymbol{\eta}_{i_0}, \dots, \boldsymbol{\eta}_{i_k}\} \subseteq V(\tau)$ of cardinality $k+1 \leq d$, the interior of (the k-dimensional manifold) $\mathrm{co}(S)$ belongs to the interior of $\omega(\boldsymbol{\eta})$ whenever $\boldsymbol{\eta} \in S$. Using Exercise 5.27, it then only remains to check that $\mathrm{co}(S) \subseteq \partial\omega(\boldsymbol{\eta})$ whenever $\boldsymbol{\eta} \notin S$, which yields the claimed formula. We are now in position to show well-definedness of the basis functions in (5.3.4).

Lemma 5.17. *For each interior node $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$ there exists a unique cpwl function $\varphi_{\boldsymbol{\eta}} : \Omega \to \mathbb{R}$ satisfying* (5.3.4)*. Moreover, $\varphi_{\boldsymbol{\eta}}$ can be expressed by a ReLU neural network with size, width, and depth bounds that only depend on d and $k_{\mathcal{T}}$.*

Proof. By Lemma 5.15, on each $\tau \in \mathcal{T}$, the affine function $\varphi_{\boldsymbol{\eta}}|_\tau$ is uniquely defined through the values at the nodes of τ. This defines a continuous function $\varphi_{\boldsymbol{\eta}} : \Omega \to \mathbb{R}$. Indeed, whenever $\tau \cap \tau' \neq \emptyset$, then $\tau \cap \tau'$ is a subsimplex of both τ and τ' in the sense of Definition 5.13 (ii). Thus, applying Lemma 5.15 again, the affine functions on τ and τ' coincide on $\tau \cap \tau'$.

Using Lemma 5.15, Lemma 5.16 and the fact that $\varphi_{\boldsymbol{\eta}}(\boldsymbol{\mu}) = 0$ whenever $\boldsymbol{\mu} \neq \boldsymbol{\eta}$, we find that $\varphi_{\boldsymbol{\eta}}$ vanishes on the boundary of the patch $\omega(\boldsymbol{\eta}) \subseteq \Omega$. Thus, $\varphi_{\boldsymbol{\eta}}$ vanishes on the boundary of Ω. Extending by zero, it becomes a cpwl function $\varphi_{\boldsymbol{\eta}} : \mathbb{R}^d \to \mathbb{R}$. This function is nonzero only on elements τ for which $\boldsymbol{\eta} \in \tau$. Hence, it is a cpwl function with at most $n := k_{\mathcal{T}} + 1$ affine functions. By Theorem 5.7, $\varphi_{\boldsymbol{\eta}}$ can be expressed as a ReLU neural network with the claimed size, width and depth bounds. □

Finally, Theorem 5.14 is now an easy consequence of the above lemmata.

of Theorem 5.14. With

$$\Phi(\boldsymbol{x}) := \sum_{\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}} f(\boldsymbol{\eta}) \varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) \qquad \text{for } \boldsymbol{x} \in \Omega, \tag{5.3.6}$$

it holds that $\Phi : \Omega \to \mathbb{R}$ satisfies $\Phi(\boldsymbol{\eta}) = f(\boldsymbol{\eta})$ for all $\boldsymbol{\eta} \in \mathcal{V}$. By Lemma 5.15 this implies that f equals Φ on each τ, and thus f equals Φ on all of Ω. Since each element τ is the convex hull of $d+1$ nodes $\boldsymbol{\eta} \in \mathcal{V}$, the cardinality of $\mathcal{V}$ is bounded by the cardinality of $\mathcal{T}$ times $d+1$. Thus, the summation in (5.3.6) is over $O(|\mathcal{T}|)$ terms. Using Lemma 5.4 and Lemma 5.17 we obtain the claimed bounds on size, width, and depth of the neural network. □

5.3.3 Size bounds for locally convex triangulations

Assuming local convexity of the triangulation, in this section we make the dependence of the constants in Theorem 5.14 explicit in the dimension d and in the maximal number of simplices $k_{\mathcal{T}}$ touching a node, see (5.3.2). As such the improvement over Theorem 5.14 is modest, and the reader may choose to skip this section on a first pass. Nonetheless, the proof, originally from [85], is entirely constructive and gives some further insight on how ReLU networks express functions. Let us start by stating the required convexity constraint.

Definition 5.18. A regular triangulation $\mathcal{T}$ is called **locally convex** if and only if $\omega(\boldsymbol{\eta})$ is convex for all interior nodes $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$.

The following theorem is a variant of [85, Theorem 3.1].

Theorem 5.19. *Let $d \in \mathbb{N}$, and let $\Omega \subseteq \mathbb{R}^d$ be a bounded domain. Let $\mathcal{T}$ be a locally convex regular triangulation of Ω. Let $f : \Omega \to \mathbb{R}$ be cpwl with respect to $\mathcal{T}$ and $f|_{\partial\Omega} = 0$. Then, there exists a constant $C > 0$ (independent of d, f and $\mathcal{T}$) and there exists a neural network $\Phi^f : \Omega \to \mathbb{R}$ such that $\Phi^f = f$,*

$$\begin{aligned} \operatorname{size}(\Phi^f) &\leq C \cdot (1 + d^2 k_{\mathcal{T}} |\mathcal{T}|), \\ \operatorname{width}(\Phi^f) &\leq C \cdot (1 + d \log(k_{\mathcal{T}}) |\mathcal{T}|), \\ \operatorname{depth}(\Phi^f) &\leq C \cdot (1 + \log_2(k_{\mathcal{T}})). \end{aligned}$$

Assume in the following that $\mathcal{T}$ is a locally convex triangulation. We will split the proof of the theorem again into a few lemmata. First, we will show that a convex patch can be written as an intersection of finitely many half-spaces. Specifically, with the **affine hull** of a set S defined as

$$\operatorname{aff}(S) := \left\{ \sum_{j=1}^{n} \alpha_j \boldsymbol{x}_j \,\middle|\, n \in \mathbb{N},\ \boldsymbol{x}_j \in S,\ \alpha_j \in \mathbb{R},\ \sum_{j=1}^{n} \alpha_j = 1 \right\} \tag{5.3.7}$$

let in the following for $\tau \in \mathcal{T}$ and $\boldsymbol{\eta} \in V(\tau)$

$$H_0(\tau, \boldsymbol{\eta}) := \operatorname{aff}(V(\tau) \backslash \{\boldsymbol{\eta}\})$$

be the affine hyperplane passing through all nodes in $V(\tau)\backslash\{\boldsymbol{\eta}\}$, and let further

$$H_+(\tau, \boldsymbol{\eta}) := \{\boldsymbol{x} \in \mathbb{R}^d \,|\, \boldsymbol{x} \text{ is on the same side of } H_0(\tau, \boldsymbol{\eta}) \text{ as } \boldsymbol{\eta}\} \cup H_0(\tau, \boldsymbol{\eta}).$$

Lemma 5.20. *Let $\boldsymbol{\eta}$ be an interior node. Then a patch $\omega(\boldsymbol{\eta})$ is convex if and only if*

$$\omega(\boldsymbol{\eta}) = \bigcap_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \mathcal{T}\}} H_+(\tau, \boldsymbol{\eta}). \tag{5.3.8}$$

Proof. The right-hand side is a finite intersection of (convex) half-spaces, and thus itself convex. It remains to show that if $\omega(\boldsymbol{\eta})$ is convex, then (5.3.8) holds. We start with "$\supset$". Suppose $\boldsymbol{x} \notin \omega(\boldsymbol{\eta})$. Then the straight line $\operatorname{co}(\{\boldsymbol{x}, \boldsymbol{\eta}\})$ must pass through $\partial\omega(\boldsymbol{\eta})$, and by Lemma 5.16 this implies that there exists $\tau \in \mathcal{T}$ with $\boldsymbol{\eta} \in \tau$ such that $\operatorname{co}(\{\boldsymbol{x}, \boldsymbol{\eta}\})$ passes through $\operatorname{aff}(V(\tau)\backslash\{\boldsymbol{\eta}\}) = H_0(\tau, \boldsymbol{\eta})$.

Hence $\boldsymbol{\eta}$ and $\boldsymbol{x}$ lie on different sides of this affine hyperplane, which shows "$\supseteq$". Now we show "$\subseteq$". Let $\tau \in \mathcal{T}$ be such that $\boldsymbol{\eta} \in \tau$ and fix $\boldsymbol{x}$ in the complement of $H_+(\tau, \boldsymbol{\eta})$. Suppose that $\boldsymbol{x} \in \omega(\boldsymbol{\eta})$. By convexity, we then have $\mathrm{co}(\{\boldsymbol{x}\} \cup \tau) \subseteq \omega(\boldsymbol{\eta})$. This implies that there exists a point in $\mathrm{co}(V(\tau)\backslash\{\boldsymbol{\eta}\})$ belonging to the interior of $\omega(\boldsymbol{\eta})$. This contradicts Lemma 5.16. Thus, $\boldsymbol{x} \notin \omega(\boldsymbol{\eta})$. □

The above lemma allows us to explicitly construct the basis functions $\varphi_{\boldsymbol{\eta}}$ in (5.3.4). To see this, denote in the following for $\tau \in \mathcal{T}$ and $\boldsymbol{\eta} \in V(\tau)$ by $g_{\tau,\boldsymbol{\eta}} \in \mathbb{P}_1(\mathbb{R}^d)$ the affine function such that

$$g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\mu}) = \begin{cases} 1 & \text{if } \boldsymbol{\eta} = \boldsymbol{\mu} \\ 0 & \text{if } \boldsymbol{\eta} \neq \boldsymbol{\mu} \end{cases} \qquad \text{for all } \boldsymbol{\mu} \in V(\tau).$$

This function exists and is unique by Lemma 5.15. Observe that $\varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) = g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \tau$.

Lemma 5.21. *Let $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$ be an interior node and let $\omega(\boldsymbol{\eta})$ be a convex patch. Then*

$$\varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) = \max\left\{0, \min_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x})\right\} \qquad \textit{for all } \boldsymbol{x} \in \mathbb{R}^d. \tag{5.3.9}$$

Proof. First let $\boldsymbol{x} \notin \omega(\boldsymbol{\eta})$. By Lemma 5.20 there exists $\tau \in V(\boldsymbol{\eta})$ such that $\boldsymbol{x}$ is in the complement of $H_+(\tau, \boldsymbol{\eta})$. Observe that

$$g_{\tau,\boldsymbol{\eta}}|_{H_+(\tau,\boldsymbol{\eta})} \geq 0 \qquad \text{and} \qquad g_{\tau,\boldsymbol{\eta}}|_{H_+(\tau,\boldsymbol{\eta})^c} < 0. \tag{5.3.10}$$

Thus

$$\min_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}} g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) < 0 \qquad \text{for all } \boldsymbol{x} \in \omega(\boldsymbol{\eta})^c,$$

i.e., (5.3.9) holds for all $\boldsymbol{x} \in \mathbb{R}\backslash\omega(\boldsymbol{\eta})$. Next, let τ, $\tau' \in \mathcal{T}$ such that $\boldsymbol{\eta} \in \tau$ and $\boldsymbol{\eta} \in \tau'$. We wish to show that $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) \leq g_{\tau',\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \tau$. Since $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) = \varphi_{\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \tau$, this then concludes the proof of (5.3.9). By Lemma 5.20 it holds

$$\boldsymbol{\mu} \in H_+(\tau', \boldsymbol{\eta}) \qquad \text{for all} \qquad \boldsymbol{\mu} \in V(\tau).$$

Hence, by (5.3.10)

$$g_{\tau',\boldsymbol{\eta}}(\boldsymbol{\mu}) \geq 0 = g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\mu}) \qquad \text{for all} \qquad \boldsymbol{\mu} \in V(\tau)\backslash\{\boldsymbol{\eta}\}.$$

Moreover, $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\eta}) = g_{\tau',\boldsymbol{\eta}}(\boldsymbol{\eta}) = 1$. Thus, $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{\mu}) \geq g_{\tau',\boldsymbol{\eta}}(\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in V(\tau')$ and therefore

$$g_{\tau',\boldsymbol{\eta}}(\boldsymbol{x}) \geq g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}) \qquad \text{for all } \boldsymbol{x} \in \mathrm{co}(V(\tau')) = \tau'.$$

□

of Theorem 5.19. For every interior node $\boldsymbol{\eta} \in \mathcal{V} \cap \mathring{\Omega}$, the cpwl basis function $\varphi_{\boldsymbol{\eta}}$ in (5.3.4) can be expressed as in (5.3.9), i.e.,

$$\varphi_{\boldsymbol{\eta}}(\boldsymbol{x}) = \sigma \bullet \Phi^{\min}_{|\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}|} \bullet (g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}))_{\{\tau \in \mathcal{T} \,|\, \boldsymbol{\eta} \in \tau\}},$$

where $(g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x}))_{\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}}$ denotes the parallelization with shared inputs of the functions $g_{\tau,\boldsymbol{\eta}}(\boldsymbol{x})$ for all $\tau\in\mathcal{T}$ such that $\boldsymbol{\eta}\in\tau$.

For this neural network, with $|\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}|\leq k_{\mathcal{T}}$, we have by Lemma 5.2

$$\begin{aligned}\operatorname{size}(\varphi_{\boldsymbol{\eta}}) &\leq 4\big(\operatorname{size}(\sigma)+\operatorname{size}(\Phi^{\min}_{|\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}|})+\operatorname{size}((g_{\tau,\boldsymbol{\eta}})_{\{\tau\in\mathcal{T}\,|\,\boldsymbol{\eta}\in\tau\}})\big)\\ &\leq 4(2+16k_{\mathcal{T}}+k_{\mathcal{T}}d)\end{aligned}\tag{5.3.11}$$

and similarly

$$\operatorname{depth}(\varphi_{\boldsymbol{\eta}})\leq 4+\lceil\log_2(k_{\mathcal{T}})\rceil,\qquad \operatorname{width}(\varphi_{\boldsymbol{\eta}})\leq\max\{1,3k_{\mathcal{T}},d\}.\tag{5.3.12}$$

Since for every interior node, the number of simplices touching the node must be larger or equal to d, we can assume $\max\{k_{\mathcal{T}},d\}=k_{\mathcal{T}}$ in the following (otherwise there exist no interior nodes, and the function f is constant 0). As in the proof of Theorem 5.14, the neural network

$$\Phi(\boldsymbol{x}):=\sum_{\boldsymbol{\eta}\in\mathcal{V}\cap\mathring{\Omega}} f(\boldsymbol{\eta})\varphi_{\boldsymbol{\eta}}(\boldsymbol{x})$$

realizes the function f on all of Ω. Since the number of nodes $|\mathcal{V}|$ is bounded by $(d+1)|\mathcal{T}|$, an application of Lemma 5.4 yields the desired bounds. □

5.4 Convergence rates for Hölder continuous functions

Theorem 5.14 immediately implies convergence rates for certain classes of (low regularity) functions. Recall for example the **space of Hölder continuous functions**: for $s\in(0,1]$ and a bounded domain $\Omega\subseteq\mathbb{R}^d$ we define

$$\|f\|_{C^{0,s}(\Omega)}:=\sup_{\boldsymbol{x}\in\Omega}|f(\boldsymbol{x})|+\sup_{\boldsymbol{x}\neq\boldsymbol{y}\in\Omega}\frac{|f(\boldsymbol{x})-f(\boldsymbol{y})|}{\|\boldsymbol{x}-\boldsymbol{y}\|_2^s}.\tag{5.4.1}$$

Then, $C^{0,s}(\Omega)$ is the set of functions $f\in C^0(\Omega)$ for which $\|f\|_{C^{0,s}(\Omega)}<\infty$.

Hölder continuous functions can be approximated well by certain cpwl functions. Therefore, we obtain the following result.

Theorem 5.22. *Let $d\in\mathbb{N}$. There exists a constant $C=C(d)$ such that for every $f\in C^{0,s}([0,1]^d)$ and every N there exists a ReLU neural network Φ_N^f with*

$$\operatorname{size}(\Phi_N^f)\leq CN,\qquad \operatorname{width}(\Phi_N^f)\leq CN,\qquad \operatorname{depth}(\Phi_N^f)=C$$

and

$$\sup_{\boldsymbol{x}\in[0,1]^d}\Big|f(\boldsymbol{x})-\Phi_N^f(\boldsymbol{x})\Big|\leq C\|f\|_{C^{0,s}([0,1]^d)}N^{-\frac{s}{d}}.$$

Proof. For $M \geq 2$, consider the set of nodes $\{\boldsymbol{\nu}/M \,|\, \boldsymbol{\nu} \in \{-1, \dots, M+1\}^d\}$ where $\boldsymbol{\nu}/M = (\nu_1/M, \dots, \nu_d/M)$. These nodes suggest a partition of $[-1/M, 1+1/M]^d$ into $(2+M)^d$ sub-hypercubes. Each such sub-hypercube can be partitioned into $d!$ simplices, such that we obtain a regular triangulation $\mathcal{T}$ with $d!(2+M)^d$ elements on $[0,1]^d$. According to Theorem 5.14 there exists a neural network Φ that is cpwl with respect to $\mathcal{T}$ and $\Phi(\boldsymbol{\nu}/M) = f(\boldsymbol{\nu}/M)$ whenever $\boldsymbol{\nu} \in \{0, \dots, M\}^d$ and $\Phi(\boldsymbol{\nu}/M) = 0$ for all other (boundary) nodes. It holds

$$\begin{aligned} \operatorname{size}(\Phi) &\leq C|\mathcal{T}| = Cd!(2+M)^d, \\ \operatorname{width}(\Phi) &\leq C|\mathcal{T}| = Cd!(2+M)^d, \\ \operatorname{depth}(\Phi) &\leq C \end{aligned} \tag{5.4.2}$$

for a constant C that only depends on d (since for our regular triangulation $\mathcal{T}$, $k_{\mathcal{T}}$ in (5.3.2) is a fixed d-dependent constant).

Let us bound the error. Fix a point $\boldsymbol{x} \in [0,1]^d$. Then $\boldsymbol{x}$ belongs to one of the interior simplices τ of the triangulation. Two nodes of the simplex have distance at most

$$\left(\sum_{j=1}^{d} \left(\frac{1}{M} \right)^2 \right)^{1/2} = \frac{\sqrt{d}}{M} =: \varepsilon.$$

Since $\Phi|_\tau$ is the linear interpolant of f at the nodes $V(\tau)$ of the simplex τ, $\Phi(\boldsymbol{x})$ is a convex combination of the $(f(\boldsymbol{\eta}))_{\boldsymbol{\eta} \in V(\tau)}$. Fix an arbitrary node $\boldsymbol{\eta}_0 \in V(\tau)$. Then $\|\boldsymbol{x} - \boldsymbol{\eta}_0\|_2 \leq \varepsilon$ and

$$\begin{aligned} |\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{\eta}_0)| \leq \max_{\boldsymbol{\eta}, \boldsymbol{\mu} \in V(\tau)} |f(\boldsymbol{\eta}) - f(\boldsymbol{\mu})| &\leq \sup_{\substack{\boldsymbol{x}, \boldsymbol{y} \in [0,1]^d \\ \|\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \varepsilon}} |f(\boldsymbol{x}) - f(\boldsymbol{y})| \\ &\leq \|f\|_{C^{0,s}([0,1]^d)} \varepsilon^s. \end{aligned}$$

Hence, using $f(\boldsymbol{\eta}_0) = \Phi(\boldsymbol{\eta}_0)$,

$$\begin{aligned} |f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| &\leq |f(\boldsymbol{x}) - f(\boldsymbol{\eta}_0)| + |\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{\eta}_0)| \\ &\leq 2\|f\|_{C^{0,s}([0,1]^d)} \varepsilon^s \\ &= 2\|f\|_{C^{0,s}([0,1]^d)} d^{\frac{s}{2}} M^{-s} \\ &= 2d^{\frac{s}{2}} \|f\|_{C^{0,s}([0,1]^d)} N^{-\frac{s}{d}} \end{aligned} \tag{5.4.3}$$

where $N := M^d$. The statement follows by (5.4.2) and (5.4.3). □

The principle behind Theorem 5.22 can be applied in even more generality. Since we can represent every cpwl function on a regular triangulation with a neural network of size $O(N)$, where N denotes the number of elements, all of classical (e.g. finite element) approximation theory for cpwl functions can be lifted to generate statements about ReLU approximation. For instance, it is well-known, that functions in the Sobolev space $H^2([0,1]^d)$ can be approximated by cpwl functions on a regular triangulation in terms of $L^2([0,1]^d)$ with the rate $2/d$. Similar as in the proof of Theorem 5.22, for every $f \in H^2([0,1]^d)$ and every $N \in \mathbb{N}$ there then exists a ReLU neural network Φ_N such that $\operatorname{size}(\Phi_N) = O(N)$ and

$$\|f - \Phi_N\|_{L^2([0,1]^d)} \leq C\|f\|_{H^2([0,1]^d)} N^{-\frac{2}{d}}.$$

Finally, we can wonder how to approximate even smoother functions, i.e., those that have many continuous derivatives. Since more smoothness is a restrictive assumption on the set of functions to approximate, we would hope that this will allow us to have smaller neural networks. Essentially, we desire a result similar to Theorem 4.9, but with the ReLU activation function.

However, we will see in the following chapter, that the emulation of piecewise affine functions on regular triangulations cannot yield the approximation rates of Theorem 4.9. To harness the smoothness, it will be necessary to build ReLU neural networks that emulate polynomials. Surprisingly, we will see in Chapter 7 that polynomials can be very efficiently approximated by *deep* ReLU neural networks.

Bibliography and further reading

The ReLU calculus introduced in Section 5.1 was similarly given in [174]. The fact that every cpwl function can be expressed as a maximum over a minimum of linear functions goes back to the papers [226, 225]; also see [169, 237].

The main result of Section 5.2, which shows that every cpwl function can be expressed by a ReLU network, is then a straightforward consequence. This was first observed in [4], which also provided bounds on the network size. These bounds were significantly improved in [85] for cwpl functions on triangular meshes that satisfy a local convexity condition. Under this assumption, it was shown that the network size essentially only grows linearly with the number of pieces. The paper [136] showed that the convexity assumption is not necessary for this statement to hold. We give a similar result in Section 5.3.2, using a simpler argument than [136]. The locally convex case from [85] is separately discussed in Section 5.3.3, as it allows for further improvements in some constants.

The implications for the approximation of Hölder continuous functions discussed in Section 5.4, follows by standard approximation theory for cpwl functions. For a general reference on splines and piecewise polynomial approximation see for instance [207]. Finally we mention that similar convergence results can also be shown for other activation functions, see, e.g., [144].

Exercises

Exercise 5.23. Let $p : \mathbb{R} \to \mathbb{R}$ be a polynomial of degree $n \geq 1$ (with leading coefficient nonzero) and let $s : \mathbb{R} \to \mathbb{R}$ be a continuous sigmoidal activation function. Show that the identity map $x \mapsto x : \mathbb{R} \to \mathbb{R}$ belongs to $\mathcal{N}_1^1(p; 1, n+1)$ but not to $\mathcal{N}_1^1(s; L)$ for any $L \in \mathbb{N}$.

Exercise 5.24. Consider cpwl functions $f : \mathbb{R} \to \mathbb{R}$ with $n \in \mathbb{N}_0$ breakpoints (points where the function is not C^1). Determine the minimal size required to exactly express every such f with a depth-1 ReLU neural network.

Exercise 5.25. Show that, the notion of affine independence is invariant under permutations of the points.

Exercise 5.26. Let $\tau = \operatorname{co}(\boldsymbol{x}_0, \dots, \boldsymbol{x}_d)$ be a d-simplex. Show that the coefficients $\alpha_i \geq 0$ such that $\sum_{i=0}^d \alpha_i = 1$ and $\boldsymbol{x} = \sum_{i=0}^d \alpha_i \boldsymbol{x}_i$ are unique for every $\boldsymbol{x} \in \tau$.

Exercise 5.27. Let $\tau = \operatorname{co}(\boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d)$ be a d-simplex. Show that the boundary of τ is given by $\bigcup_{i=0}^d \operatorname{co}(\{\boldsymbol{\eta}_0, \dots, \boldsymbol{\eta}_d\} \backslash \{\boldsymbol{\eta}_i\})$.

Chapter 6

Affine pieces for ReLU neural networks

In the previous chapters, we observed some remarkable approximation results of shallow ReLU neural networks. In practice, however, deeper architectures are more common. To understand why, we in this chapter we discuss some potential shortcomings of shallow ReLU networks compared to deep ReLU networks.

Traditionally, an insightful approach to study limitations of ReLU neural networks has been to analyze the number of linear regions these functions can generate.

Definition 6.1. Let $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$, and let $f\colon \Omega \to \mathbb{R}$ be cpwl (see Definition 5.5). We say that f has $p \in \mathbb{N}$ **pieces (or linear regions)**, if p is the smallest number of connected open sets $(\Omega_i)_{i=1}^p$ such that $\bigcup_{i=1}^p \overline{\Omega_i} = \Omega$, and $f|_{\Omega_i}$ is an affine function for all $i = 1, \dots, p$. We denote $\mathrm{Pieces}(f, \Omega) \coloneqq p$.

For $d = 1$ we call every point where f is not differentiable a **break point** of f.

To get an accurate cpwl approximation of a function, the approximating function needs to have many pieces. The next theorem, corresponding to [62, Theorem 2], quantifies this statement.

Theorem 6.2. *Let $-\infty < a < b < \infty$ and $f \in C^3([a,b])$ so that f is not affine. Then there exists a constant $c > 0$ depending only on $\int_a^b \sqrt{|f''(x)|}\,\mathrm{d}x$ so that*

$$\|g - f\|_{L^\infty([a,b])} > cp^{-2}$$

for all cpwl g with at most $p \in \mathbb{N}$ pieces.

The proof of the theorem is left to the reader, see Exercise 6.12.

Theorem 6.2 implies that for ReLU neural networks we need architectures allowing for many pieces, if we want to approximate non-linear functions to high accuracy. But how many pieces can

we create for a fixed depth and width? We will establish a simple theoretical upper bound in Section 6.1. Subsequently, we will investigate under which conditions these upper bounds are attainable in Section 6.2. This will reveal that certain functions necessitate very large shallow networks for approximation, whereas relatively small deep networks can also approximate them. These findings are presented in Section 6.3.

Finally, we will question the practical relevance of this analysis by examining how many pieces typical neural networks possess. Surprisingly, in Section 6.4 we will find that randomly initialized deep neural networks on average do not have a number of pieces that is anywhere close to the theoretical upper bound.

6.1 Upper bounds

Neural networks are based on the composition and addition of neurons. These two operations increase the possible number of pieces in a very specific way. Figure 6.1 depicts the two operations and their effect. They can be described as follows:

- *Summation:* Let $\Omega \subseteq \mathbb{R}$. The sum of two cpwl functions $f_1,\ f_2 : \Omega \to \mathbb{R}$ satisfies

$$\mathrm{Pieces}(f_1 + f_2, \Omega) \leq \mathrm{Pieces}(f_1, \Omega) + \mathrm{Pieces}(f_2, \Omega) - 1. \tag{6.1.1}$$

  This holds because the sum is affine in every point where both f_1 and f_2 are affine. Therefore, the sum has at most as many break points as f_1 and f_2 combined. Moreover, the number of pieces of a univariate function equals the number of its break points plus one.

- *Composition:* Let again $\Omega \subseteq \mathbb{R}$. The composition of two functions $f_1 \colon \mathbb{R}^d \to \mathbb{R}$ and $f_2 \colon \Omega \to \mathbb{R}^d$ satisfies

$$\mathrm{Pieces}(f_1 \circ f_2, \Omega) \leq \mathrm{Pieces}(f_1, \mathbb{R}^d) \cdot \mathrm{Pieces}(f_2, \Omega). \tag{6.1.2}$$

  This is because for each of the affine pieces of f_2—let us call one of those pieces $A \subseteq \mathbb{R}$—we have that f_2 is either constant or injective on A. If it is constant, then $f_1 \circ f_2$ is constant. If it is injective, then $\mathrm{Pieces}(f_1 \circ f_2, A) = \mathrm{Pieces}(f_1, f_2(A)) \leq \mathrm{Pieces}(f_1, \mathbb{R}^d)$. Since this holds for all pieces of f_2 we get (6.1.2).

These considerations give the following result, which follows the argument of [227, Lemma 2.1]. We state it for general cpwl activation functions. The ReLU activation function corresponds to $p = 2$.

Theorem 6.3. *Let $L \in \mathbb{N}$. Let σ be cpwl with p pieces. Then, every neural network with architecture $(\sigma; 1, d_1, \ldots, d_L, 1)$ has at most $(p \cdot \mathrm{width}(\Phi))^L$ pieces.*

Proof. The proof is via induction over the depth L. Let $L = 1$, and let $\Phi : \mathbb{R} \to \mathbb{R}$ be a neural network of architecture $(\sigma; 1, d_1, 1)$. Then

$$\Phi(x) = \sum_{k=1}^{d_1} w_k^{(1)} \sigma(w_k^{(0)} x + b_k^{(0)}) + b^{(1)} \qquad \text{for } x \in \mathbb{R},$$

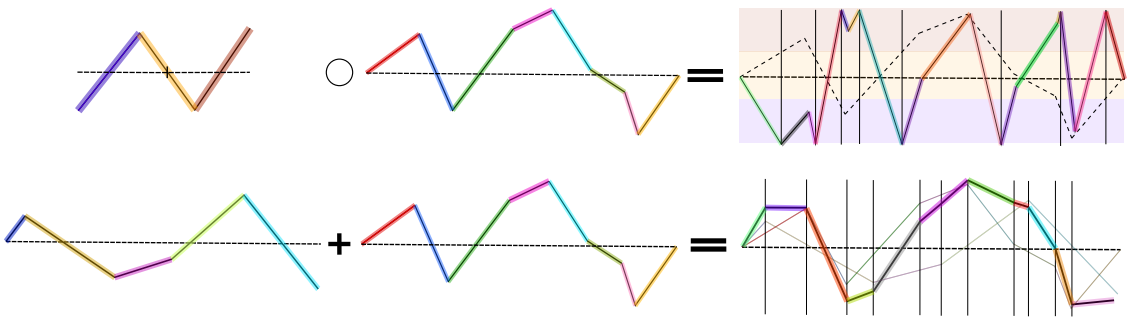


Figure 6.1: **Top:** Composition of two cpwl functions $f_1 \circ f_2$ can create a piece whenever the value of f_2 crosses a level that is associated to a break point of f_1. **Bottom:** Addition of two cpwl functions $f_1 + f_2$ produces a cpwl function that can have break points at positions where either f_1 or f_2 has a break point.

for certain $\boldsymbol{w}^{(0)}$, $\boldsymbol{w}^{(1)}$, $\boldsymbol{b}^{(0)} \in \mathbb{R}^{d_1}$ and $b^{(1)} \in \mathbb{R}$. By (6.1.1), $\mathrm{Pieces}(\Phi) \leq p \cdot \mathrm{width}(\Phi)$.

For the induction step, assume the statement holds for $L \in \mathbb{N}$, and let $\Phi : \mathbb{R} \to \mathbb{R}$ be a neural network of architecture $(\sigma; 1, d_1, \dots, d_{L+1}, 1)$. Then, we can write

$$\Phi(x) = \sum_{j=1}^{d_{L+1}} w_j \sigma(h_j(x)) + b \qquad \text{for } x \in \mathbb{R},$$

for some $\boldsymbol{w} \in \mathbb{R}^{d_{L+1}}$, $b \in \mathbb{R}$, and where each h_j is a neural network of architecture $(\sigma; 1, d_1, \dots, d_L, 1)$. Using the induction hypothesis, each $\sigma \circ h_\ell$ has at most $p \cdot (p \cdot \mathrm{width}(\Phi))^L$ affine pieces. Hence Φ has at most $\mathrm{width}(\Phi) \cdot p \cdot (p \cdot \mathrm{width}(\Phi))^L = (p \cdot \mathrm{width}(\Phi))^{L+1}$ affine pieces. This completes the proof. □

Theorem 6.3 shows that there are limits to how many pieces can be created with a certain architecture. It is noteworthy that the effects of the depth and the width of a neural network are vastly different. While increasing the width can polynomially increase the number of pieces, increasing the depth can result in exponential increase. This is a first indication of the prowess of depth of neural networks.

To understand the effect of this on the approximation problem, we apply the bound of Theorem 6.3 to Theorem 6.2.

Theorem 6.4. *Let $d_0 \in \mathbb{N}$ and $f \in C^3([0,1]^{d_0})$. Assume there exists a line segment $\mathfrak{s} \subseteq [0,1]^{d_0}$ of positive length such that $0 < c := \int_{\mathfrak{s}} \sqrt{|f''(x)|}\,\mathrm{d}x$. Then, there exists $C > 0$ solely depending on c, such that for all ReLU neural networks $\Phi : \mathbb{R}^{d_0} \to \mathbb{R}$ with L layers*

$$\|f - \Phi\|_{L^\infty([0,1]^{d_0})} \geq c \cdot (2\mathrm{width}(\Phi))^{-2L}.$$

Theorem 6.4 gives a lower bound on achievable approximation rates in dependence of the depth L. As target functions become smoother, we expect that we can achieve faster convergence rates

(cp. Chapter 4). However, without increasing the depth, it seems to be impossible to leverage such additional smoothness.

This observation strongly indicates that deeper architectures can be superior. Before we can make such statements, we first explore whether the upper bounds of Theorem 6.3 are even achievable.

6.2 Tightness of upper bounds

To construct a ReLU neural network, that realizes the upper bound of Theorem 6.3, we first let $h_1 : [0,1] \to \mathbb{R}$ be the hat function

$$h_1(x) := \begin{cases} 2x & \text{if } x \in [0, \frac{1}{2}] \\ 2 - 2x & \text{if } x \in [\frac{1}{2}, 1]. \end{cases}$$

This function can be expressed by a ReLU neural network of depth one and with two nodes

$$h_1(x) = \sigma_{\mathrm{ReLU}}(2x) - \sigma_{\mathrm{ReLU}}(4x - 2) \qquad \text{for all } x \in [0,1]. \tag{6.2.1}$$

We recursively set $h_n := h_{n-1} \circ h_1$ for all $n \geq 2$, i.e., $h_n = h_1 \circ \cdots \circ h_1$ is the n-fold composition of h_1. Since $h_1 : [0,1] \to [0,1]$, we have $h_n : [0,1] \to [0,1]$ and

$$h_n \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 2).$$

It turns out that this function has a rather interesting behavior. It is a "sawtooth" function with 2^{n-1} spikes, see Figure 6.2.

Lemma 6.5. *Let $n \in \mathbb{N}$. It holds for all $x \in [0,1]$*

$$h_n(x) = \begin{cases} 2^n(x - i2^{-n}) & \text{if } i \geq 0 \text{ is even and } x \in [i2^{-n}, (i+1)2^{-n}] \\ 2^n((i+1)2^{-n} - x) & \text{if } i \geq 1 \text{ is odd and } x \in [i2^{-n}, (i+1)2^{-n}]. \end{cases}$$

Proof. The case $n = 1$ holds by definition. We proceed by induction, and assume the statement holds for n. Let $x \in [0, 1/2]$ and $i \geq 0$ even such that $x \in [i2^{-(n+1)}, (i+1)2^{-(n+1)}]$. Then $2x \in [i2^{-n}, (i+1)2^{-n}]$. Thus

$$h_n(h_1(x)) = h_n(2x) = 2^n(2x - i2^{-n}) = 2^{n+1}(x - i2^{-n+1}).$$

Similarly, if $x \in [0, 1/2]$ and $i \geq 1$ odd such that $x \in [i2^{-(n+1)}, (i+1)2^{-(n+1)}]$, then $h_1(x) = 2x \in [i2^{-n}, (i+1)2^{-n}]$ and

$$h_n(h_1(x)) = h_n(2x) = 2^n(2x - (i+1)2^{-n}) = 2^{n+1}(x - (i+1)2^{-n+1}).$$

The case $x \in [1/2, 1]$ follows by observing that h_{n+1} is symmetric around $1/2$. □

The neural network h_n has size $O(n)$ and is piecewise linear on at least 2^n pieces. This shows that the number of pieces can indeed increase exponentially in the neural network size, also see the upper bound in Theorem 6.3.

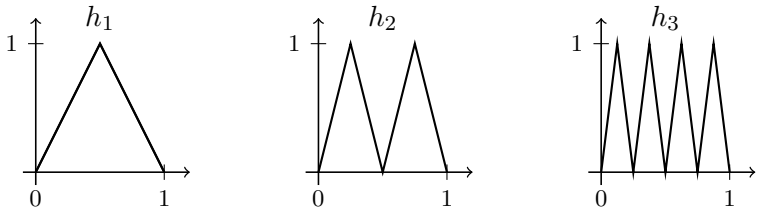


Figure 6.2: The functions h_n in Lemma 6.5.

6.3 Depth separation

Now that we have established how increasing the depth can lead to exponentially more pieces than increasing the width, we can deduce a so-called "depth-separation" result shown by Telgarsky in [227, 228]. Such statements verify the existence of functions that can easily be approximated by deep neural networks, but require much larger size when approximated by shallow neural networks. The following theorem, along with its proof, is presented similarly in Telgarsky's lecture notes [229].

Theorem 6.6. *For every $n \in \mathbb{N}$ there exists a neural network $f \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n^2+3, 2)$ such that for any $g \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 2^{n-1})$ holds*

$$\int_0^1 |f(x) - g(x)| \, \mathrm{d}x \geq \frac{1}{32}.$$

The neural network f may have quadratically more layers than g, but $\mathrm{width}(g) = 2^{n-1}$ and $\mathrm{width}(f) = 2$. Hence the *size of g may be exponentially larger than the size of f*, but nonetheless no such g can approximate f. Thus even exponential increase in width cannot necessarily compensate for increase in depth. The proof is based on the following observations stated in [228]:

(i) Functions with few oscillations poorly approximate functions with many oscillations,

(ii) neural networks with few layers have few oscillations,

(iii) neural networks with many layers can have many oscillations.

Proof of Theorem 6.6. Fix $n \in \mathbb{N}$. Let $f := h_{n^2+3} \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n^2+3, 2)$. For arbitrary $g \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 2^{n-1})$, by Theorem 6.3, g is piecewise linear with at most $(2 \cdot 2^{n-1})^n = 2^{n^2}$ break points. The function f is the sawtooth function with 2^{n^2+2} spikes. The number of triangles formed by the graph of f and the constant line at $1/2$ equals $2^{n^2+3} - 1$, each with area $2^{-(n^2+5)}$, see Figure 6.3. For the m triangles in between two break points of g, the graph of g does *not* cross at

Figure 6.3: **Left:** The functions h_n form $2^n - 1$ triangles with the line at $1/2$, each with area $2^{-(n+2)}$. **Right:** For an affine function with m (in this sketch $m = 5$) triangles in between two break points, the function can cross at most $\lceil m/2 \rceil + 1 \leq m/2 + 2$ of them. Figure adapted from [229, Section 5].

least $m - (m/2 + 2) = m/2 - 2$ of them. Thus we can bound

$$\begin{aligned}\int_0^1 |f(x) - g(x)| \, \mathrm{d}x &\geq \underbrace{\left(\frac{1}{2}(\underbrace{2^{n^2+3} - 1 - 2^{n^2}}_{\substack{\geq \text{triangles on an interval} \\ \text{without break point of } g}}) - \underbrace{2 \cdot 2^{n^2}}_{\geq 2 \cdot (\text{pieces of } g)} \right)}_{\geq \text{missed triangles}} \underbrace{2^{-(n^2+5)}}_{\text{area of a triangle}} \\ &\geq (2^{n^2+2} - 3 \cdot 2^{n^2}) \cdot 2^{-(n^2+5)} \\ &\geq 2^{n^2} \cdot 2^{-(n^2+5)} = \frac{1}{32},\end{aligned}$$

which concludes the proof. □

6.4 Number of pieces in practice

We have seen in Theorem 6.3 that deep neural networks *can* have many more pieces than their shallow counterparts. This begs the question if deep neural networks tend to generate more pieces in practice. More formally: If we randomly initialize the weights of a neural network, what is the expected number of linear regions? Will this number scale exponentially with the depth? This question was analyzed in [82], and surprisingly, it was found that the number of pieces of randomly initialized neural networks typically does *not* depend exponentially on the depth. In Figure 6.4, we depict two neural networks, one shallow and one deep, that were randomly initialized according to He initialization [86]. Both neural networks have essentially the same number of pieces (114 and 110) and there is no clear indication that one has a deeper architecture than the other.

In the following, we will give a simplified version of the main result of [82] to show why random deep neural networks often behave like shallow neural networks.

We recall from Figure 6.1 that pieces are generated through composition of two functions f_1 and f_2, if the values of f_2 cross a level that is associated to a break point of f_1. In the case of a simple neuron of the form

$$\boldsymbol{x} \mapsto \sigma_{\mathrm{ReLU}}(\langle \boldsymbol{a}, h(\boldsymbol{x}) \rangle + b)$$

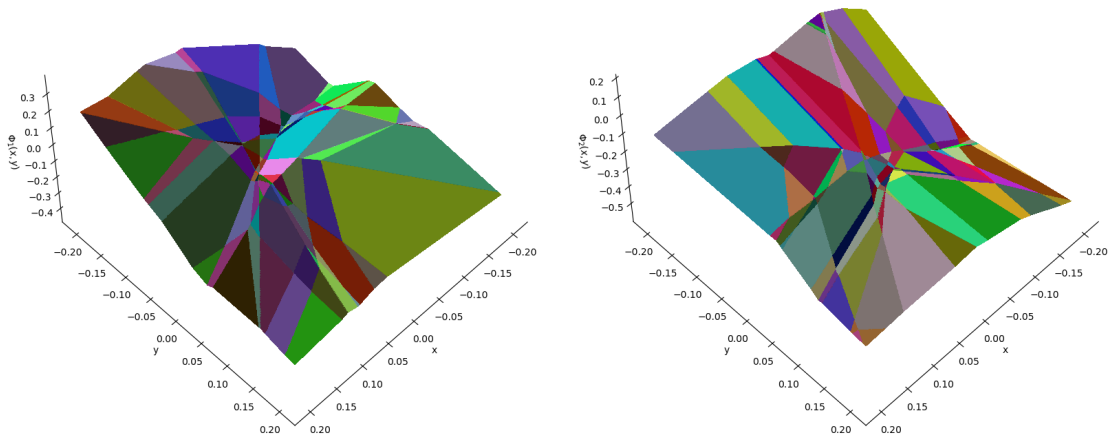


Figure 6.4: Two randomly initialized neural networks Φ_1 and Φ_2 with architectures $(\sigma_{\mathrm{ReLU}}; 1, 10, 10, 1)$ and $(\sigma_{\mathrm{ReLU}}; 1, 5, 5, 5, 5, 5, 1)$. The initialization scheme was He initialization [86]. The number of linear regions equals 114 and 110, respectively.

where h is a cpwl function, $\boldsymbol{a}$ is a vector, and b is a scalar, many pieces can be generated if $\langle \boldsymbol{a}, h(\boldsymbol{x}) \rangle$ crosses the $-b$ level often.

If $\boldsymbol{a}$, b are random variables, and we know that h does not oscillate too much, then we can quantify the probability of $\langle \boldsymbol{a}, h(\boldsymbol{x}) \rangle$ crossing the $-b$ level often. The following lemma from [115, Lemma 3.1] provides the details.

Lemma 6.7. *Let $c > 0$ and let $h\colon [0, c] \to \mathbb{R}$ be a cpwl function on $[0, c]$. Let $t \in \mathbb{N}$, let $A \subseteq \mathbb{R}$ be a Lebesgue measurable set, and assume that for every $y \in A$ it holds that*

$$|\{x \in [0, c] \,|\, h(x) = y\}| \geq t.$$

Then, $c\|h'\|_{L^\infty} \geq \|h'\|_{L^1} \geq |A| \cdot t$, where $|A|$ is the Lebesgue measure of A.

In particular, if h has at most $P \in \mathbb{N}$ pieces and $\|h'\|_{L^1}$ is finite, then it holds for all $\delta > 0$ that for all $t \leq P$

$$\mathbb{P}\left[|\{x \in [0, c] \,|\, h(x) = U\}| \geq t\right] \leq \frac{\|h'\|_{L^1}}{\delta t},$$
$$\mathbb{P}\left[|\{x \in [0, c] \,|\, h(x) = U\}| > P\right] = 0,$$

where U is a uniformly distributed variable on $[-\delta/2, \delta/2]$.

Proof. We will assume $c = 1$. The general case then follows by considering $\tilde{h}(x) = h(x/c)$.

Let for $(c_i)_{i=1}^{P+1} \subseteq [0, 1]$ with $c_1 = 0$, $c_{P+1} = 1$ and $c_i \leq c_{i+1}$ for all $i = 1, \ldots, P+1$ the pieces of h be given by $((c_i, c_{i+1}))_{i=1}^{P}$. We denote

$$V_1 \coloneqq [0, c_2], \quad V_i \coloneqq (c_i, c_{i+1}] \text{ for } i = 1, \ldots, P$$

and for $j = i, \dots, P$

$$\widetilde{V}_i \coloneqq \bigcup_{j=1}^{i-1} V_j.$$

We define, for $n \in \mathbb{N} \cup \{\infty\}$

$$T_{i,n} \coloneqq h(V_i) \cap \left\{ y \in A \,\middle|\, |\{x \in \widetilde{V}_i \,|\, h(x) = y\}| = n - 1 \right\}.$$

In words, $T_{i,n}$ contains the values of A that are hit on V_i for the nth time. Since h is cpwl, we observe that for all $i = 1, \dots, P$

(i) $T_{i,n_1} \cap T_{i,n_2} = \emptyset$ for all $n_1, n_2 \in \mathbb{N} \cup \{\infty\}$, $n_1 \neq n_2$,

(ii) $T_{i,\infty} \cup \bigcup_{n=1}^{\infty} T_{i,n} = h(V_i) \cap A$,

(iii) $T_{i,n} = \emptyset$ for all $P < n < \infty$,

(iv) $|T_{i,\infty}| = 0$.

Note that, since h is affine on V_i it holds that $h' = |h(V_i)|/|V_i|$ on V_i. Hence, for $t \leq P$

$$\begin{aligned}
\|h'\|_{L^1} &\geq \sum_{i=1}^{P} |h(V_i)| \geq \sum_{i=1}^{P} |h(V_i) \cap A| \\
&= \sum_{i=1}^{P} \left(\sum_{n=1}^{\infty} |T_{i,n}| \right) + |T_{i,\infty}| \\
&= \sum_{i=1}^{P} \sum_{n=1}^{\infty} |T_{i,n}| \\
&\geq \sum_{n=1}^{t} \sum_{i=1}^{P} |T_{i,n}|,
\end{aligned}$$

where the first equality follows by (i), (ii), the second by (iv), and the last inequality by (iii). Note that, by assumption for all $n \leq t$ every $y \in A$ is an element of $T_{i,n}$ or $T_{i,\infty}$ for some $i \leq P$. Therefore, by (iv)

$$\sum_{i=1}^{P} |T_{i,n}| \geq |A|,$$

which completes the proof. □

Lemma 6.7 applied to neural networks essentially states that, in a single neuron, if the bias term is chosen uniformly randomly on an interval of length δ, then the probability of generating at least t pieces by composition scales reciprocal to t.

Next, we will analyze how Lemma 6.7 implies an upper bound on the number of pieces generated in a randomly initialized neural network. For simplicity, we only consider random biases in the following, but mention that similar results hold if both the biases and weights are random variables [82].

Definition 6.8. Let $L \in \mathbb{N}$, $(d_0, d_1, \dots, d_L, 1) \in \mathbb{N}^{L+2}$ and $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ for $\ell = 0, \dots, L$. Furthermore, let $\delta > 0$ and let the bias vectors $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$, for $\ell = 0, \dots, L$, be random variables such that each entry of each $\boldsymbol{b}^{(\ell)}$ is independently and uniformly distributed on the interval $[-\delta/2, \delta/2]$. We call the associated ReLU neural network a **random-bias neural network**.

To apply Lemma 6.7 to a single neuron with random biases, we also need some bound on the derivative of the input to the neuron.

Definition 6.9. Let $L \in \mathbb{N}$, $(d_0, d_1, \dots, d_L, 1) \in \mathbb{N}^{L+2}$, and $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ and $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ for $\ell = 0, \dots, L$. Moreover let $\delta > 0$.

For $\ell = 1, \dots, L+1$, $i = 1, \dots, d_\ell$ introduce the functions

$$\eta_{\ell,i}(\boldsymbol{x}; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}) = (\boldsymbol{W}^{(\ell-1)} \boldsymbol{x}^{(\ell-1)})_i \qquad \text{for } \boldsymbol{x} \in \mathbb{R}^{d_0},$$

where $\boldsymbol{x}^{(\ell-1)}$ is as in (2.1.1). We call

$$\nu\left((\boldsymbol{W}^{(\ell)})_{\ell=1}^{L}, \delta\right) \coloneqq \max\left\{ \left\| \eta'_{\ell,i}(\,\cdot\,; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}) \right\|_2 \,\middle|\, (\boldsymbol{b}^{(j)})_{j=0}^{L} \in \prod_{j=0}^{L} [-\delta/2, \delta/2]^{d_{j+1}}, \ell = 1, \dots, L, i = 1, \dots, d_\ell \right\}$$

the **maximal internal derivative** of Φ.

We can now formulate the main result of this section.

Theorem 6.10. *Let $L \in \mathbb{N}$ and let $(d_0, d_1, \dots, d_L, 1) \in \mathbb{N}^{L+2}$. Let $\delta \in (0, 1]$. Let $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$, for $\ell = 0, \dots, L$, be such that $\nu\left((\boldsymbol{W}^{(\ell)})_{\ell=0}^{L}, \delta\right) \leq C_\nu$ for a $C_\nu > 0$.*

For an associated random-bias neural network Φ, we have that for a line segment $\mathfrak{s} \subseteq \mathbb{R}^{d_0}$ of length 1

$$\mathbb{E}[\mathrm{Pieces}(\Phi, \mathfrak{s})] \leq 1 + d_1 + \frac{C_\nu}{\delta}(1 + (L-1)\ln(2\mathrm{width}(\Phi))) \sum_{j=2}^{L} d_j. \tag{6.4.1}$$

Proof. Let $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ for $\ell = 0, \dots, L$. Moreover, let $\boldsymbol{b}^{(\ell)} \in [-\delta/2, \delta/2]^{d_{\ell+1}}$ for $\ell = 0, \dots, L$ be uniformly distributed random variables. We denote

$$\begin{aligned} \theta_\ell \colon \mathfrak{s} &\to \mathbb{R}^{d_\ell} \\ \boldsymbol{x} &\mapsto (\eta_{\ell,i}(\boldsymbol{x}; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}))_{i=1}^{d_\ell}. \end{aligned}$$

Let $\kappa\colon \mathfrak{s} \to [0,1]$ be an isomorphism. Since each coordinate of θ_ℓ is cpwl, there are points $\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_{q_\ell} \in \mathfrak{s}$ with $\kappa(\boldsymbol{x}_j) < \kappa(\boldsymbol{x}_{j+1})$ for $j = 0, \dots, q_\ell - 1$, such that θ_ℓ is affine (as a function into $\mathbb{R}^{d_\ell}$) on $[\kappa(\boldsymbol{x}_j), \kappa(\boldsymbol{x}_{j+1})]$ for all $j = 0, \dots, q_\ell - 1$ as well as on $[0, \kappa(\boldsymbol{x}_0)]$ and $[\kappa(\boldsymbol{x}_{q_\ell}), 1]$.

We will now inductively find an upper bound on the q_ℓ.

Let $\ell = 2$, then

$$\theta_2(\boldsymbol{x}) = \boldsymbol{W}^{(1)} \sigma_{\mathrm{ReLU}}(\boldsymbol{W}^{(0)} \boldsymbol{x} + \boldsymbol{b}^{(0)}).$$

Since $\boldsymbol{W}^{(1)} \cdot + b^{(1)}$ is an affine function, it follows that θ_2 can only be non-affine in points where $\sigma_{\mathrm{ReLU}}(\boldsymbol{W}^{(0)} \cdot + \boldsymbol{b}^{(0)})$ is not affine. Therefore, θ_2 is only non-affine if one coordinate of $\boldsymbol{W}^{(0)} \cdot + \boldsymbol{b}^{(0)}$ intersects 0 nontrivially. This can happen at most d_1 times. We conclude that we can choose $q_2 = d_1$.

Next, let us find an upper bound on $q_{\ell+1}$ from q_ℓ. Note that

$$\theta_{\ell+1}(\boldsymbol{x}) = \boldsymbol{W}^{(\ell)} \sigma_{\mathrm{ReLU}}(\theta_\ell(\boldsymbol{x}) + \boldsymbol{b}^{(\ell-1)}).$$

Now $\theta_{\ell+1}$ is affine in every point $\boldsymbol{x} \in \mathfrak{s}$ where θ_ℓ is affine and $(\theta_\ell(\boldsymbol{x}) + \boldsymbol{b}^{(\ell-1)})_i \neq 0$ for all coordinates $i = 1, \dots, d_\ell$. As a result, we have that we can choose $q_{\ell+1}$ such that

$$q_{\ell+1} \leq q_\ell + \left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, (\theta_\ell(\boldsymbol{x}) + \boldsymbol{b}^{(\ell-1)})_i = 0 \text{ for at least one } i = 1, \dots, d_\ell \} \right|.$$

Therefore, for $\ell \geq 2$

$$\begin{aligned} q_{\ell+1} &\leq d_1 + \sum_{j=3}^{\ell} \left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, (\theta_j(\boldsymbol{x}) + \boldsymbol{b}^{(j)})_i = 0 \text{ for at least one } i = 1, \dots, d_j \} \right| \\ &\leq d_1 + \sum_{j=2}^{\ell} \sum_{i=1}^{d_j} \left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, \eta_{j,i}(\boldsymbol{x}) = -\boldsymbol{b}_i^{(j)} \} \right|. \end{aligned}$$

By Theorem 6.3, we have that

$$\mathrm{Pieces}\left(\eta_{\ell,i}(\,\cdot\,; (\boldsymbol{W}^{(j)}, \boldsymbol{b}^{(j)})_{j=0}^{\ell-1}), \mathfrak{s} \right) \leq (2\mathrm{width}(\Phi))^{\ell-1}.$$

We define for $k \in \mathbb{N} \cup \{\infty\}$

$$p_{k,\ell,i} \coloneqq \mathbb{P}\left[\left| \{ \boldsymbol{x} \in \mathfrak{s} \,|\, \eta_{\ell,i}(\boldsymbol{x}) = -\boldsymbol{b}_i^{(\ell)} \} \right| \geq k \right]$$

Then by Lemma 6.7

$$p_{k,\ell,i} \leq \frac{C_\nu}{\delta k}$$

and for $k > (2\mathrm{width}(\Phi))^{\ell-1}$

$$p_{k,\ell,i} = 0.$$

It holds

$$\begin{aligned}
&\mathbb{E}\left[\sum_{j=2}^{L}\sum_{i=1}^{d_j}\left|\left\{\boldsymbol{x}\in\mathfrak{s}\,\middle|\,\eta_{j,i}(\boldsymbol{x})=-\boldsymbol{b}_i^{(j)}\right\}\right|\right]\\
\leq&\sum_{j=2}^{L}\sum_{i=1}^{d_j}\sum_{k=1}^{\infty}k\cdot\mathbb{P}\left[\left|\left\{\boldsymbol{x}\in\mathfrak{s}\,\middle|\,\eta_{j,i}(\boldsymbol{x})=-\boldsymbol{b}_i^{(j)}\right\}\right|=k\right]\\
\leq&\sum_{j=2}^{L}\sum_{i=1}^{d_j}\sum_{k=1}^{\infty}k\cdot(p_{k,j,i}-p_{k+1,j,i}).
\end{aligned}$$

The inner sum can be bounded by

$$\begin{aligned}
\sum_{k=1}^{\infty}k\cdot(p_{k,j,i}-p_{k+1,j,i})&=\sum_{k=1}^{\infty}k\cdot p_{k,j,i}-\sum_{k=1}^{\infty}k\cdot p_{k+1,j,i}\\
&=\sum_{k=1}^{\infty}k\cdot p_{k,j,i}-\sum_{k=2}^{\infty}(k-1)\cdot p_{k,j,i}\\
&=p_{1,j,i}+\sum_{k=2}^{\infty}p_{k,j,i}\\
&=\sum_{k=1}^{\infty}p_{k,j,i}\\
&\leq C_\nu\delta^{-1}\sum_{k=1}^{(2\mathrm{width}(\Phi))^{L-1}}\frac{1}{k}\\
&\leq C_\nu\delta^{-1}\left(1+\int_1^{(2\mathrm{width}(\Phi))^{L-1}}\frac{1}{x}\,\mathrm{d}x\right)\\
&\leq C_\nu\delta^{-1}(1+(L-1)\ln((2\mathrm{width}(\Phi)))).
\end{aligned}$$

We conclude that, in expectation, we can bound q_{L+1} by

$$d_1+C_\nu\delta^{-1}(1+(L-1)\ln(2\mathrm{width}(\Phi)))\sum_{j=2}^{L}d_j.$$

Finally, since $\theta_L=\Phi_{L+1}|_{\mathfrak{s}}$, it follows that

$$\mathrm{Pieces}(\Phi,\mathfrak{s})\leq q_{L+1}+1$$

which yields the result. □

Remark 6.11. We make the following observations about Theorem 6.10:

- *Non-exponential dependence on depth:* If we consider (6.4.1), we see that the number of pieces scales in expectation essentially like $\mathcal{O}(LN)$, where N is the total number of neurons of the architecture. This shows that in expectation, the number of pieces is linear in the number of layers, as opposed to the exponential upper bound of Theorem 6.3.

- *Maximal internal derivative:* Theorem 6.10 requires the weights to be chosen such that the maximal internal derivative is bounded by a certain number. However, if they are randomly initialized in such a way that with high probability the maximal internal derivative is bounded by a small number, then similar results can be shown. In practice, weights in the ℓth layer are often initialized according to a centered normal distribution with standard deviation $\sqrt{2/d_\ell}$, [86]. Due to the anti-proportionality of the variance to the width of the layers it is achieved that the internal derivatives remain bounded with high probability, independent of the width of the neural networks. This explaines the observation from Figure 6.4.

Bibliography and further reading

Establishing bounds on the number of linear regions of a ReLU network has been a popular tool to investigate the complexity of ReLU neural networks, see [152, 185, 4, 210, 82]. The bound presented in Section 6.1, is based on [227]. In addition to this bound, the paper also presents the depth separation result discussed in Section 6.3. The proof techniques employed there have inspired numerous subsequent works in the field.

Together with the lower bound on the number of required linear regions given in [62], this analysis shows how depth can be a limiting factor in terms of achievable convergence rates, as stated in Theorem 6.4.

For the construction of the sawtooth function in Section 6.2, and the depth separation result in Section 6.3 follow the arguments in [227, 228, 229]. Beyond Telgarsky's work, other notable depth separation results include [60, 199, 4]. Moreover, closely related to such statements is the 1987 thesis by Håstad [101], which considers the limitations of logic circuits in terms of depth.

Finally, the analysis of the number of pieces deep neural networks attained with random intialization (Section 6.4) is based on [82] and [115].

Exercises

Exercise 6.12. Let $-\infty < a < b < \infty$ and let $f \in C^3([a,b]) \backslash \mathbb{P}_1$. Denote by $p(\varepsilon) \in \mathbb{N}$ the minimal number of intervals partitioning $[a,b]$, such that a (not necessarily continuous) piecewise linear function on $p(\varepsilon)$ intervals can approximate f on $[a,b]$ uniformly up to error $\varepsilon > 0$. In this exercise, we wish to show

$$\liminf_{\varepsilon \searrow 0} p(\varepsilon)\sqrt{\varepsilon} > 0. \tag{6.4.2}$$

Therefore, we can find a constant $C > 0$ such that $\varepsilon \geq C p(\varepsilon)^{-2}$ for all $\varepsilon > 0$. This shows a variant of Theorem 6.2. Proceed as follows to prove (6.4.2):

(i) Fix $\varepsilon > 0$ and let $a = x_0 < x_1 \cdots < x_{p(\varepsilon)} = b$ be a partitioning into $p(\varepsilon)$ pieces. For $i = 0, \ldots, p(\varepsilon) - 1$ and $x \in [x_i, x_{i+1}]$ let

$$e_i(x) := f(x) - \left(f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) \right).$$

Show that $|e_i(x)| \leq 2\varepsilon$ for all $x \in [x_i, x_{i+1}]$.

(ii) With $h_i := x_{i+1} - x_i$ and $m_i := (x_i + x_{i+1})/2$ show that

$$\max_{x \in [x_i, x_{i+1}]} |e_i(x)| = \frac{h_i^2}{8} |f''(m_i)| + O(h_i^3).$$

(iii) Assuming that $c := \inf_{x \in [a,b]} |f''(x)| > 0$ show that

$$\liminf_{\varepsilon \searrow 0} p(\varepsilon)\sqrt{\varepsilon} \geq \frac{1}{4} \int_a^b \sqrt{|f''(x)|}\, \mathrm{d}x.$$

(iv) Conclude that (6.4.2) holds for general non-linear $f \in C^3([a,b])$.

Exercise 6.13. Show that, for $L = 1$, Theorem 6.3 holds for piecewise smooth functions, when replacing the number of affine pieces by the number of smooth pieces. These are defined by replacing "affine" by "smooth" (meaning C^∞) in Definition 6.1.

Exercise 6.14. Show that, for $L > 1$, Theorem 6.3 does *not*x hold for piecewise smooth functions, when replacing the number of affine pieces by the number of smooth pieces.

Exercise 6.15. For $p \in \mathbb{N}$, $p > 2$ and $n \in \mathbb{N}$, construct a function $h_n^{(p)}$ similar to h_n of (6.5), such that $h_n^{(p)} \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, p)$ and such that $h_n^{(p)}$ has p^n pieces and size $O(p^2 n)$.

Chapter 7

Deep ReLU neural networks

In the previous chapter, we observed that many layers are a necessary prerequisite for ReLU neural networks to approximate smooth functions with high rates. We now analyze which depth is sufficient to achieve good approximation rates for smooth functions.

To approximate smooth functions efficiently, one of the main tools in Chapter 4 was to rebuild polynomial-based functions, such as higher-order B-splines. For smooth activation functions, we were able to reproduce polynomials by using the nonlinearity of the activation functions. This argument certainly cannot be repeated for the *piecewise linear* ReLU. On the other hand, up until now, we have seen that deep ReLU neural networks are extremely efficient at producing the strongly oscillating sawtooth functions discussed in Lemma 6.5.

The main observation this chapter is that the efficient representation of sawtooth functions is intimately linked to the approximation of the square function and hence allows very efficient approximations of polynomial functions. This observation was first made by Dmitry Yarotsky [245] in 2016, and the present chapter is primarily based on this paper.

First, in Section 7.1, we will give an efficient neural network approximation of the squaring function. Second, in Section 7.2, we will demonstrate how the squaring neural network can be modified to yield a neural network that approximates the function that multiplies its inputs. Using these two tools, we conclude in Section 7.3 that deep ReLU neural networks can efficiently approximate k-times continuously differentiable functions with Hölder continuous derivatives.

7.1 The square function

In this section, we will show that the square function $x \mapsto x^2$ can be approximated very efficiently by a deep neural network.

Proposition 7.1. *Let $n \in \mathbb{N}$. Then*

$$s_n(x) := x - \sum_{j=1}^{n} \frac{h_j(x)}{2^{2j}}$$

is a piecewise linear function on $[0,1]$ with break points $x_{n,j} = j2^{-n}$, $j = 0, \dots, 2^n$. Moreover, $s_n(x_{n,k}) = x_{n,k}^2$ for all $k = 0, \dots, 2^n$, i.e. s_n is the piecewise linear interpolant of x^2 on $[0,1]$.

Proof. The statement holds for $n = 1$. We proceed by induction. Assume the statement holds for s_n and let $k \in \{0, \dots, 2^{n+1}\}$. By Lemma 6.5, $h_{n+1}(x_{n+1,k}) = 0$ whenever k is even. Hence for even $k \in \{0, \dots, 2^{n+1}\}$

$$\begin{aligned} s_{n+1}(x_{n+1,k}) &= x_{n+1,k} - \sum_{j=1}^{n+1} \frac{h_j(x_{n+1,k})}{2^{2j}} \\ &= s_n(x_{n+1,k}) - \frac{h_{n+1}(x_{n+1,k})}{2^{2(n+1)}} = s_n(x_{n+1,k}) = x_{n+1,k}^2, \end{aligned}$$

where we used the induction assumption $s_n(x_{n+1,k}) = x_{n+1,k}^2$ for $x_{n+1,k} = k2^{-(n+1)} = \frac{k}{2}2^{-n} = x_{n,k/2}$.

Now let $k \in \{1, \dots, 2^{n+1} - 1\}$ be odd. Then by Lemma 6.5, $h_{n+1}(x_{n+1,k}) = 1$. Moreover, since s_n is linear on $[x_{n,(k-1)/2}, x_{n,(k+1)/2}] = [x_{n+1,k-1}, x_{n+1,k+1}]$ and $x_{n+1,k}$ is the midpoint of this interval,

$$\begin{aligned} s_{n+1}(x_{n+1,k}) &= s_n(x_{n+1,k}) - \frac{h_{n+1}(x_{n+1,k})}{2^{2(n+1)}} \\ &= \frac{1}{2}(x_{n+1,k-1}^2 + x_{n+1,k+1}^2) - \frac{1}{2^{2(n+1)}} \\ &= \frac{(k-1)^2}{2^{2(n+1)+1}} + \frac{(k+1)^2}{2^{2(n+1)+1}} - \frac{2}{2^{2(n+1)+1}} \\ &= \frac{1}{2}\frac{2k^2}{2^{2(n+1)}} = \frac{k^2}{2^{2(n+1)}} = x_{n+1,k}^2. \end{aligned}$$

This completes the proof. □

Lemma 7.2. *For $n \in \mathbb{N}$, it holds*

$$\sup_{x \in [0,1]} |x^2 - s_n(x)| \leq 2^{-2n-1}.$$

Moreover $s_n \in \mathcal{N}_1^1(\sigma_{\mathrm{ReLU}}; n, 3)$, and $\mathrm{size}(s_n) \leq 7n$ and $\mathrm{depth}(s_n) = n$.

Proof. Set $e_n(x) := x^2 - s_n(x)$. Let x be in the interval $[x_{n,k}, x_{n,k+1}] = [k2^{-n}, (k+1)2^{-n}]$ of length 2^{-n}. Since s_n is the linear interpolant of x^2 on this interval, we have

$$|e_n'(x)| = \left| 2x - \frac{x_{n,k+1}^2 - x_{n,k}^2}{2^{-n}} \right| = \left| 2x - \frac{2k+1}{2^n} \right| \leq \frac{1}{2^n}.$$

Thus $e_n : [0,1] \to \mathbb{R}$ has Lipschitz constant 2^{-n}. Since $e_n(x_{n,k}) = 0$ for all $k = 0, \dots, 2^n$, and the length of the interval $[x_{n,k}, x_{n,k+1}]$ equals 2^{-n} we get

$$\sup_{x \in [0,1]} |e_n(x)| \leq \frac{1}{2} 2^{-n} 2^{-n} = 2^{-2n-1}.$$

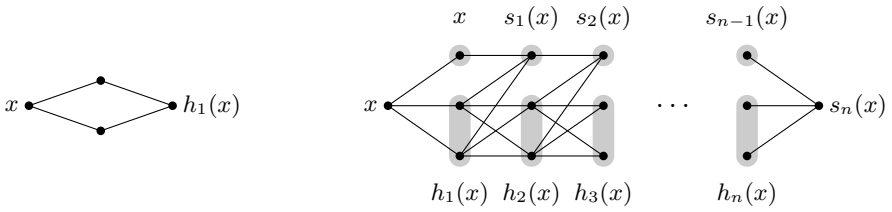


Figure 7.1: The neural networks $h_1(x) = \sigma_{\mathrm{ReLU}}(2x) - \sigma_{\mathrm{ReLU}}(4x-2)$ and $s_n(x) = \sigma_{\mathrm{ReLU}}(s_{n-1}(x)) - h_n(x)/2^{2n}$ where $h_n = h_1 \circ h_{n-1}$.

Finally, to see that s_n can be represented by a neural network of the claimed architecture, note that for $n \geq 2$

$$s_n(x) = x - \sum_{j=1}^{n} \frac{h_j(x)}{2^{2j}} = s_{n-1}(x) - \frac{h_n(x)}{2^{2n}} = \sigma_{\mathrm{ReLU}} \circ s_{n-1}(x) - \frac{h_1 \circ h_{n-1}(x)}{2^{2n}}.$$

Here we used that s_{n-1} is the piecewise linear interpolant of x^2, so that $s_{n-1}(x) \geq 0$ and thus $s_{n-1}(x) = \sigma_{\mathrm{ReLU}}(s_{n-1}(x))$ for all $x \in [0,1]$. Hence s_n is of depth n and width 3, see Figure 7.1. □

In conclusion, we have shown that $s_n : [0,1] \to [0,1]$ approximates the square function uniformly on $[0,1]$ with exponentially decreasing error in the neural network size. Note that due to Theorem 6.4, this would not be possible with a shallow neural network, which can at best interpolate x^2 on a partition of $[0,1]$ with polynomially many (w.r.t. the neural network size) pieces.

7.2 Multiplication

According to Lemma 7.2, depth can help in the approximation of $x \mapsto x^2$, which, on first sight, seems like a rather specific example. However, as we shall discuss in the following, this opens up a path towards fast approximation of functions with high regularity, e.g., $C^k([0,1]^d)$ for some $k > 1$. The crucial observation is that, via the polarization identity we can write the product of two numbers as a sum of squares

$$x \cdot y = \frac{(x+y)^2 - (x-y)^2}{4} \tag{7.2.1}$$

for all x, $y \in \mathbb{R}$. Efficient approximation of the operation of multiplication allows efficient approximation of polynomials. Those in turn are well-known to be good approximators for functions exhibiting $k \in \mathbb{N}$ derivatives. Before exploring this idea further in the next section, we first make precise the observation that neural networks can efficiently approximate the multiplication of real numbers.

We start with the multiplication of two numbers, in which case neural networks of logarithmic size in the desired accuracy are sufficient.

Lemma 7.3. *For every $\varepsilon > 0$ there exists a ReLU neural network $\Phi_\varepsilon^\times : [-1,1]^2 \to [-1,1]$ such that*

$$\sup_{x,y\in[-1,1]} |x \cdot y - \Phi_\varepsilon^\times(x,y)| \leq \varepsilon,$$

and it holds $\operatorname{size}(\Phi_\varepsilon^\times) \leq C \cdot (1 + |\log(\varepsilon)|)$ *and* $\operatorname{depth}(\Phi_\varepsilon^\times) \leq C \cdot (1 + |\log(\varepsilon)|)$ *for a constant $C > 0$ independent of ε. Moreover, $\Phi_\varepsilon^\times(x,y) = 0$ if $x = 0$ or $y = 0$.*

Proof. With $n = \lceil |\log_4(\varepsilon)| \rceil$, define the neural network

$$\begin{aligned}\Phi_\varepsilon^\times(x,y) :=& s_n\left(\frac{\sigma_{\mathrm{ReLU}}(x+y) + \sigma_{\mathrm{ReLU}}(-x-y)}{2}\right) \\ &- s_n\left(\frac{\sigma_{\mathrm{ReLU}}(x-y) + \sigma_{\mathrm{ReLU}}(y-x)}{2}\right).\end{aligned} \tag{7.2.2}$$

Since $|a| = \sigma_{\mathrm{ReLU}}(a) + \sigma_{\mathrm{ReLU}}(-a)$, by (7.2.1) we have for all x, $y \in [-1,1]$

$$\begin{aligned}\left|x \cdot y - \Phi_\varepsilon^\times(x,y)\right| &= \left|\frac{(x+y)^2 - (x-y)^2}{4} - \left(s_n\left(\frac{|x+y|}{2}\right) - s_n\left(\frac{|x-y|}{2}\right)\right)\right| \\ &= \left|\frac{4(\frac{x+y}{2})^2 - 4(\frac{x-y}{2})^2}{4} - \frac{4s_n(\frac{|x+y|}{2}) - 4s_n(\frac{|x-y|}{2})}{4}\right| \\ &\leq \frac{4(2^{-2n-1} + 2^{-2n-1})}{4} = 4^{-n} \leq \varepsilon,\end{aligned}$$

where we used $|x+y|/2$, $|x-y|/2 \in [0,1]$. We have $\operatorname{depth}(\Phi_\varepsilon^\times) = 1 + \operatorname{depth}(s_n) = 1 + n \leq 1 + \lceil \log_4(\varepsilon) \rceil$ and $\operatorname{size}(\Phi_\varepsilon^\times) \leq C + 2\operatorname{size}(s_n) \leq Cn \leq C \cdot (1 - \log(\varepsilon))$ for some constant $C > 0$.

The fact that $\Phi_\varepsilon^\times$ maps from $[-1,1]^2 \to [-1,1]$ follows by (7.2.2) and because $s_n : [0,1] \to [0,1]$. Finally, if $x = 0$, then $\Phi_\varepsilon^\times(x,y) = s_n(|x+y|) - s_n(|x-y|) = s_n(|y|) - s_n(|y|) = 0$. If $y = 0$ the same argument can be made. □

In a similar way as in Proposition 4.8 and Lemma 5.11, we can apply operations with two inputs in the form of a binary tree to extend them to an operation on arbitrary many inputs.

Proposition 7.4. *For every $n \geq 2$ and $\varepsilon > 0$ there exists a ReLU neural network $\Phi_{n,\varepsilon}^\times : [-1,1]^n \to [-1,1]$ such that*

$$\sup_{x_j\in[-1,1]} \left|\prod_{j=1}^n x_j - \Phi_{n,\varepsilon}^\times(x_1,\ldots,x_n)\right| \leq \varepsilon,$$

and it holds $\operatorname{size}(\Phi_{n,\varepsilon}^\times) \leq Cn \cdot (1 + |\log(\varepsilon/n)|)$ *and* $\operatorname{depth}(\Phi_{n,\varepsilon}^\times) \leq C\log(n)(1 + |\log(\varepsilon/n)|)$ *for a constant $C > 0$ independent of ε and n.*

Proof. We begin with the case $n = 2^k$. For $k = 1$ let $\tilde{\Phi}^{\times}_{2,\delta} := \Phi^{\times}_{\delta}$. If $k \geq 2$ let

$$\tilde{\Phi}^{\times}_{2^k,\delta} := \Phi^{\times}_{\delta} \circ \left(\tilde{\Phi}^{\times}_{2^{k-1},\delta}, \tilde{\Phi}^{\times}_{2^{k-1},\delta}\right).$$

Using Lemma 7.3, we find that this neural network has depth bounded by

$$\operatorname{depth}\left(\tilde{\Phi}^{\times}_{2^k,\delta}\right) \leq k\operatorname{depth}(\Phi^{\times}_{\delta}) \leq Ck \cdot (1 + |\log(\delta)|) \leq C\log(n)(1 + |\log(\delta)|).$$

Observing that the number of occurences of $\Phi^{\times}_{\delta}$ equals $\sum_{j=0}^{k-1} 2^j \leq n$, the size of $\tilde{\Phi}^{\times}_{2^k,\delta}$ can bounded by $Cn\operatorname{size}(\Phi^{\times}_{\delta}) \leq Cn \cdot (1 + |\log(\delta)|)$.

To estimate the approximation error, denote with $\boldsymbol{x} = (x_j)_{j=1}^{2^k}$

$$e_k := \sup_{x_j \in [-1,1]} \left| \prod_{j \leq 2^k} x_j - \tilde{\Phi}^{\times}_{2^k,\delta}(\boldsymbol{x}) \right|.$$

Then, using short notation of the type $\boldsymbol{x}_{\leq 2^{k-1}} := (x_1, \dots, x_{2^{k-1}})$,

$$\begin{aligned}
e_k &= \sup_{x_j \in [-1,1]} \left| \prod_{j=1}^{2^k} x_j - \Phi^{\times}_{\delta}\left(\tilde{\Phi}^{\times}_{2^{k-1},\delta}(\boldsymbol{x}_{\leq 2^{k-1}}), \tilde{\Phi}^{\times}_{2^{k-1},\delta}(\boldsymbol{x}_{>2^{k-1}})\right) \right| \\
&\leq \delta + \sup_{x_j \in [-1,1]} \left(\left| \prod_{j \leq 2^{k-1}} x_j \right| e_{k-1} + \left| \tilde{\Phi}^{\times}_{2^{k-1},\delta}(\boldsymbol{x}_{>2^{k-1}}) \right| e_{k-1} \right) \\
&\leq \delta + 2e_{k-1} \leq \delta + 2(\delta + 2e_{k-2}) \leq \cdots \leq \delta \sum_{j=0}^{k-2} 2^j + 2^{k-1} e_1 \\
&\leq 2^k \delta = n\delta = \varepsilon.
\end{aligned}$$

Here we used $e_1 \leq \delta$, and that $\tilde{\Phi}^{\times}_{2^k,\delta}$ maps $[-1,1]^{2^{k-1}}$ to $[-1,1]$, which is a consequence of Lemma 7.3.

The case for general $n \geq 2$ (not necessarily $n = 2^k$) is treated similar as in Lemma 5.11, by replacing some $\Phi^{\times}_{\delta}$ neural networks with identity neural networks.

Finally, setting $\delta := \varepsilon/n$ and $\Phi^{\times}_{n,\varepsilon} := \tilde{\Phi}^{\times}_{n,\delta}$ concludes the proof. □

7.3 $C^{k,s}$ functions

We will now discuss the implications of our observations in the previous sections for the approximation of functions in the class $C^{k,s}$.

Definition 7.5. Let $k \in \mathbb{N}_0$, $s \in [0,1]$ and $\Omega \subseteq \mathbb{R}^d$. Then

$$\begin{aligned}
\|f\|_{C^{k,s}(\Omega)} := &\sup_{\boldsymbol{x} \in \Omega} \max_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}| \leq k\}} |D^{\boldsymbol{\alpha}} f(\boldsymbol{x})| \\
&+ \sup_{\boldsymbol{x} \neq \boldsymbol{y} \in \Omega} \max_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}| = k\}} \frac{|D^{\boldsymbol{\alpha}} f(\boldsymbol{x}) - D^{\boldsymbol{\alpha}} f(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_2^s},
\end{aligned} \tag{7.3.1}$$

and we denote by $C^{k,s}(\Omega)$ the set of functions $f \in C^k(\Omega)$ for which $\|f\|_{C^{k,s}(\Omega)} < \infty$.

Note that these spaces are ordered according to

$$C^k(\Omega) \supseteq C^{k,s}(\Omega) \supseteq C^{k,t}(\Omega) \supseteq C^{k+1}(\Omega)$$

for all $0 < s \leq t \leq 1$.

In order to state our main result, we first recall a version of Taylor's remainder formula for $C^{k,s}(\Omega)$ functions.

Lemma 7.6. *Let $d \in \mathbb{N}$, $k \in \mathbb{N}$, $s \in [0,1]$, $\Omega = [0,1]^d$ and $f \in C^{k,s}(\Omega)$. Then for all $\boldsymbol{a}$, $\boldsymbol{x} \in \Omega$*

$$f(\boldsymbol{x}) = \sum_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, 0 \leq |\boldsymbol{\alpha}| \leq k\}} \frac{D^{\boldsymbol{\alpha}} f(\boldsymbol{a})}{\boldsymbol{\alpha}!} (\boldsymbol{x} - \boldsymbol{a})^{\boldsymbol{\alpha}} + R_k(\boldsymbol{x}) \tag{7.3.2}$$

where with $h := \max_{i \leq d} |a_i - x_i|$ we have $|R_k(\boldsymbol{x})| \leq h^{k+s} \frac{d^{k+1/2}}{k!} \|f\|_{C^{k,s}(\Omega)}$.

Proof. First, for a function $g \in C^k(\mathbb{R})$ and a, $t \in \mathbb{R}$

$$\begin{aligned} g(t) &= \sum_{j=0}^{k-1} \frac{g^{(j)}(a)}{j!} (t-a)^j + \frac{g^{(k)}(\xi)}{k!} (t-a)^k \\ &= \sum_{j=0}^{k} \frac{g^{(j)}(a)}{j!} (t-a)^j + \frac{g^{(k)}(\xi) - g^{(k)}(a)}{k!} (t-a)^k, \end{aligned}$$

for some ξ between a and t. Now let $f \in C^{k,s}(\mathbb{R}^d)$ and $\boldsymbol{a}$, $\boldsymbol{x} \in \mathbb{R}^d$. Thus with $g(t) := f(\boldsymbol{a} + t \cdot (\boldsymbol{x} - \boldsymbol{a}))$ holds for $f(\boldsymbol{x}) = g(1)$

$$f(\boldsymbol{x}) = \sum_{j=0}^{k-1} \frac{g^{(j)}(0)}{j!} + \frac{g^{(k)}(\xi)}{k!}.$$

By the chain rule

$$g^{(j)}(t) = \sum_{\{\boldsymbol{\alpha} \in \mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}| = j\}} \binom{j}{\boldsymbol{\alpha}} D^{\boldsymbol{\alpha}} f(\boldsymbol{a} + t \cdot (\boldsymbol{x} - \boldsymbol{a}))(\boldsymbol{x} - \boldsymbol{a})^{\boldsymbol{\alpha}},$$

where we use the multivariate notations $\binom{j}{\boldsymbol{\alpha}} = \frac{j!}{\boldsymbol{\alpha}!} = \frac{j!}{\prod_{j=1}^d \alpha_j!}$ and $(\boldsymbol{x} - \boldsymbol{a})^{\boldsymbol{\alpha}} = \prod_{j=1}^d (x_j - a_j)^{\alpha_j}$.

Hence

$$f(\boldsymbol{x}) = \underbrace{\sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, 0\leq|\boldsymbol{\alpha}|\leq k\}} \frac{D^{\boldsymbol{\alpha}} f(\boldsymbol{a})}{\boldsymbol{\alpha}!}(\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}}}_{\in\mathbb{P}^k} + \underbrace{\sum_{|\boldsymbol{\alpha}|=k} \frac{D^{\boldsymbol{\alpha}} f(\boldsymbol{a}+\xi\cdot(\boldsymbol{x}-\boldsymbol{a})) - D^{\boldsymbol{\alpha}} f(\boldsymbol{a})}{\boldsymbol{\alpha}!}(\boldsymbol{x}-\boldsymbol{a})^{\boldsymbol{\alpha}}}_{=:R_k},$$

for some $\xi \in [0,1]$. Using the definition of h, the remainder term can be bounded by

$$\begin{aligned}|R_k| &\leq h^k \max_{|\boldsymbol{\alpha}|=k} \sup_{\substack{\boldsymbol{x}\in\Omega \\ t\in[0,1]}} |D^{\boldsymbol{\alpha}} f(\boldsymbol{a}+t\cdot(\boldsymbol{x}-\boldsymbol{a})) - D^{\boldsymbol{\alpha}} f(\boldsymbol{a})| \frac{1}{k!} \sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|=k\}} \binom{k}{\boldsymbol{\alpha}} \\ &\leq h^{k+s} \frac{d^{k+\frac{1}{2}}}{k!} \|f\|_{C^{k,s}(\Omega)},\end{aligned}$$

where we used (7.3.1), $\|\boldsymbol{x}-\boldsymbol{a}\|_2 \leq \sqrt{d}h$ and $\sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|=k\}} \binom{k}{\boldsymbol{\alpha}} = (1+\cdots+1)^k = d^k$ by the multinomial formula. □

We now come to the main statement of this section. Up to logarithmic terms, it shows the convergence rate $(k+s)/d$ for approximating functions in $C^{k,s}([0,1]^d)$.

Theorem 7.7. *Let $d \in \mathbb{N}$, $k \in \mathbb{N}_0$, $s \in [0,1]$, and $\Omega = [0,1]^d$. Then, there exists a constant $C > 0$ such that for every $f \in C^{k,s}(\Omega)$ and every $N \geq 2$ there exists a ReLU neural network Φ_N^f such that*

$$\sup_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - \Phi_N^f(\boldsymbol{x})| \leq C N^{-\frac{k+s}{d}} \|f\|_{C^{k,s}(\Omega)}, \tag{7.3.3}$$

$\mathrm{size}(\Phi_N^f) \leq CN\log(N)$ and $\mathrm{depth}(\Phi_N^f) \leq C\log(N)$.

Proof. The idea of the proof is to use the so-called "partition of unity method": First we will construct a partition of unity $(\varphi_{\boldsymbol{\nu}})_{\boldsymbol{\nu}}$, such that for an appropriately chosen $M \in \mathbb{N}$ each $\varphi_{\boldsymbol{\nu}}$ has support on a $O(1/M)$ neighborhood of a point $\boldsymbol{\eta} \in \Omega$. On each of these neighborhoods we will use the local Taylor polynomial $p_{\boldsymbol{\nu}}$ of f around $\boldsymbol{\eta}$ to approximate the function. Then $\sum_{\boldsymbol{\nu}} \varphi_{\boldsymbol{\nu}} p_{\boldsymbol{\nu}}$ gives an approximation to f on Ω. This approximation can be emulated by a neural network of the type $\sum_{\boldsymbol{\nu}} \Phi_\varepsilon^\times(\varphi_{\boldsymbol{\nu}}, \hat{p}_{\boldsymbol{\nu}})$, where $\hat{p}_{\boldsymbol{\nu}}$ is an neural network approximation to the polynomial $p_{\boldsymbol{\nu}}$.

It suffices to show the theorem in the case where

$$\max\left\{\frac{d^{k+1/2}}{k!}, \exp(d)\right\} \|f\|_{C^{k,s}(\Omega)} \leq 1.$$

The general case can then be immediately deduced by a scaling argument.

Step 1. We construct the neural network. Define

$$M := \lceil N^{1/d} \rceil \qquad \text{and} \qquad \varepsilon := N^{-\frac{k+s}{d}}. \tag{7.3.4}$$

Consider a uniform simplicial mesh with nodes $\{\boldsymbol{\nu}/M \,|\, \boldsymbol{\nu} \leq M\}$ where $\boldsymbol{\nu}/M := (\nu_1/M, \dots, \nu_d/M)$, and where "$\boldsymbol{\nu} \leq M$" is short for $\{\boldsymbol{\nu} \in \mathbb{N}_0^d \,|\, \nu_i \leq M \text{ for all } i \leq d\}$. We denote by $\varphi_{\boldsymbol{\nu}}$ the cpwl basis function on this mesh such that $\varphi_{\boldsymbol{\nu}}(\boldsymbol{\nu}/M) = 1$ and $\varphi_{\boldsymbol{\nu}}(\boldsymbol{\mu}/M) = 0$ whenever $\boldsymbol{\mu} \neq \boldsymbol{\nu}$. As shown in Chapter 5, $\varphi_{\boldsymbol{\nu}}$ is a neural network of size $O(1)$. Then

$$\sum_{\boldsymbol{\nu} \leq M} \varphi_{\boldsymbol{\nu}} \equiv 1 \qquad \text{on } \Omega, \tag{7.3.5}$$

is a partition of unity. Moreover, observe that

$$\operatorname{supp}(\varphi_{\boldsymbol{\nu}}) \subseteq \left\{ \boldsymbol{x} \in \Omega \,\middle|\, \left\| \boldsymbol{x} - \frac{\boldsymbol{\nu}}{M} \right\|_\infty \leq \frac{1}{M} \right\}, \tag{7.3.6}$$

where $\|\boldsymbol{x}\|_\infty = \max_{i \leq d} |x_i|$.

For each $\boldsymbol{\nu} \leq M$ define the multivariate polynomial

$$p_{\boldsymbol{\nu}}(\boldsymbol{x}) := \sum_{|\boldsymbol{\alpha}| \leq k} \frac{D^{\boldsymbol{\alpha}} f\left(\frac{\boldsymbol{\nu}}{M}\right)}{\boldsymbol{\alpha}!} \left(\boldsymbol{x} - \frac{\boldsymbol{\nu}}{M}\right)^{\boldsymbol{\alpha}} \in \mathbb{P}^k,$$

and the approximation

$$\hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x}) := \sum_{|\boldsymbol{\alpha}| \leq k} \frac{D^{\boldsymbol{\alpha}} f\left(\frac{\boldsymbol{\nu}}{M}\right)}{\boldsymbol{\alpha}!} \Phi^\times_{|\boldsymbol{\alpha}|,\varepsilon}\left(x_{i_{\boldsymbol{\alpha},1}} - \frac{\nu_{i_{\boldsymbol{\alpha},1}}}{M}, \dots, x_{i_{\boldsymbol{\alpha},k}} - \frac{\nu_{i_{\boldsymbol{\alpha},k}}}{M}\right),$$

where $(i_{\boldsymbol{\alpha},1}, \dots, i_{\boldsymbol{\alpha},k}) \in \{0, \dots, d\}^k$ is arbitrary but fixed such that $|\{j \,|\, i_{\boldsymbol{\alpha},j} = r\}| = \alpha_r$ for all $r = 1, \dots, d$. Finally, define

$$\Phi_N^f := \sum_{\boldsymbol{\nu} \leq M} \Phi^\times_\varepsilon(\varphi_{\boldsymbol{\nu}}, \hat{p}_{\boldsymbol{\nu}}). \tag{7.3.7}$$

Step 2. We bound the approximation error. First, for each $\boldsymbol{x} \in \Omega$, using (7.3.5) and (7.3.6)

$$\begin{aligned}\left| f(\boldsymbol{x}) - \sum_{\boldsymbol{\nu} \leq M} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) \right| &\leq \sum_{\boldsymbol{\nu} \leq M} |\varphi_{\boldsymbol{\nu}}(\boldsymbol{x})| |p_{\boldsymbol{\nu}}(\boldsymbol{x}) - f(\boldsymbol{x})| \\ &\leq \max_{\boldsymbol{\nu} \leq M} \sup_{\{\boldsymbol{y} \in \Omega \,|\, \|\frac{\boldsymbol{\nu}}{M} - \boldsymbol{y}\|_\infty \leq \frac{1}{M}\}} |f(\boldsymbol{y}) - p_{\boldsymbol{\nu}}(\boldsymbol{y})|.\end{aligned}$$

By Lemma 7.6 we obtain

$$\sup_{\boldsymbol{x} \in \Omega} \left| f(\boldsymbol{x}) - \sum_{\boldsymbol{\nu} \leq M} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) \right| \leq M^{-(k+s)} \frac{d^{k+\frac{1}{2}}}{k!} \|f\|_{C^{k,s}(\Omega)} \leq M^{-(k+s)}. \tag{7.3.8}$$

Next, fix $\boldsymbol{\nu} \leq M$ and $\boldsymbol{y} \in \Omega$ such that $\|\boldsymbol{\nu}/M - \boldsymbol{y}\|_\infty \leq 1/M \leq 1$. Then by Proposition 7.4

$$\begin{aligned}|p_{\boldsymbol{\nu}}(\boldsymbol{y}) - \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{y})| &\leq \sum_{|\boldsymbol{\alpha}|\leq k} \frac{D^{\boldsymbol{\alpha}} f\left(\frac{\boldsymbol{\nu}}{M}\right)}{\boldsymbol{\alpha}!} \left| \prod_{j=1}^{k} \left(y_{i_{\boldsymbol{\alpha},j}} - \frac{\nu_{i_{\boldsymbol{\alpha},j}}}{M}\right) \right. \\ &\qquad \left. - \Phi^{\times}_{|\boldsymbol{\alpha}|,\varepsilon}\left(y_{i_{\boldsymbol{\alpha},1}} - \frac{\nu_{i_{\boldsymbol{\alpha},1}}}{M}, \ldots, y_{i_{\boldsymbol{\alpha},k}} - \frac{i_{\boldsymbol{\alpha},k}}{M}\right)\right| \\ &\leq \varepsilon \sum_{|\boldsymbol{\alpha}|\leq k} \frac{D^{\boldsymbol{\alpha}} f(\frac{\boldsymbol{\nu}}{M})}{\boldsymbol{\alpha}!} \leq \varepsilon \exp(d) \|f\|_{C^{k,s}(\Omega)} \leq \varepsilon, \end{aligned} \tag{7.3.9}$$

where we used $|D^{\boldsymbol{\alpha}} f(\boldsymbol{\nu}/M)| \leq \|f\|_{C^{k,s}(\Omega)}$ and

$$\sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|\leq k\}} \frac{1}{\boldsymbol{\alpha}!} = \sum_{j=0}^{k} \frac{1}{j!} \sum_{\{\boldsymbol{\alpha}\in\mathbb{N}_0^d \,|\, |\boldsymbol{\alpha}|=j\}} \frac{j!}{\boldsymbol{\alpha}!} = \sum_{j=0}^{k} \frac{d^j}{j!} \leq \sum_{j=0}^{\infty} \frac{d^j}{j!} = \exp(d).$$

Similarly, one shows that

$$|\hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x})| \leq \exp(d)\|f\|_{C^{k,s}(\Omega)} \leq 1 \qquad \text{for all } \boldsymbol{x} \in \Omega.$$

Fix $\boldsymbol{x} \in \Omega$. Then $\boldsymbol{x}$ belongs to a simplex of the mesh, and thus $\boldsymbol{x}$ can be in the support of at most $d+1$ (the number of nodes of a simplex) functions $\varphi_{\boldsymbol{\nu}}$. Moreover, Lemma 7.3 implies that $\operatorname{supp} \Phi^{\times}_{\varepsilon}(\varphi_{\boldsymbol{\nu}}(\cdot), \hat{p}_{\boldsymbol{\nu}}(\cdot)) \subseteq \operatorname{supp} \varphi_{\boldsymbol{\nu}}$. Hence, using Lemma 7.3 and (7.3.9)

$$\begin{aligned}&\left| \sum_{\boldsymbol{\nu}\leq M} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) - \sum_{\boldsymbol{\nu}\leq M} \Phi^{\times}_{\varepsilon}(\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}), \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x})) \right| \\ &\quad \leq \sum_{\{\boldsymbol{\nu}\leq M \,|\, \boldsymbol{x}\in\operatorname{supp}\varphi_{\boldsymbol{\nu}}\}} \left(|\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) p_{\boldsymbol{\nu}}(\boldsymbol{x}) - \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x})| \right. \\ &\qquad \left. + |\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x}) - \Phi^{\times}_{\varepsilon}(\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}), \hat{p}_{\boldsymbol{\nu}}(\boldsymbol{x}))| \right) \\ &\quad \leq \varepsilon + (d+1)\varepsilon = (d+2)\varepsilon. \end{aligned}$$

In total, together with (7.3.8)

$$\sup_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - \Phi^f_N(\boldsymbol{x})| \leq M^{-(k+s)} + \varepsilon \cdot (d+2).$$

With our choices in (7.3.4) this yields the error bound (7.3.3).

Step 3. It remains to bound the size and depth of the neural network in (7.3.7).

By Lemma 5.17, for each $0 \leq \boldsymbol{\nu} \leq M$ we have

$$\operatorname{size}(\varphi_{\boldsymbol{\nu}}) \leq C \cdot (1 + k_{\mathcal{T}}), \qquad \operatorname{depth}(\varphi_{\boldsymbol{\nu}}) \leq C \cdot (1 + \log(k_{\mathcal{T}})), \tag{7.3.10}$$

where $k_{\mathcal{T}}$ is the maximal number of simplices attached to a node in the mesh. Note that $k_{\mathcal{T}}$ is independent of M, so that the size and depth of $\varphi_{\boldsymbol{\nu}}$ are bounded by a constant C_φ independent of M.

Lemma 7.3 and Proposition 7.4 thus imply with our choice of $\varepsilon = N^{-(k+s)/d}$

$$\begin{aligned}\operatorname{depth}(\Phi_N^f) &= \operatorname{depth}(\Phi_\varepsilon^\times) + \max_{\boldsymbol{\nu}\leq M} \operatorname{depth}(\varphi_{\boldsymbol{\eta}}) + \max_{\boldsymbol{\nu}\leq M} \operatorname{depth}(\hat{p}_{\boldsymbol{\nu}}) \\ &\leq C\cdot(1+|\log(\varepsilon)|+C_\varphi) + \operatorname{depth}(\Phi_{k,\varepsilon}^\times) \\ &\leq C\cdot(1+|\log(\varepsilon)|+C_\varphi) \\ &\leq C\cdot(1+\log(N))\end{aligned}$$

for some constant $C>0$ depending on k and d (we use "C" to denote a generic constant that can change its value in each line).

To bound the size, we first observe with Lemma 5.4 that

$$\operatorname{size}(\hat{p}_{\boldsymbol{\nu}}) \leq C\cdot\left(1+\sum_{|\boldsymbol{\alpha}|\leq k}\operatorname{size}\left(\Phi_{|\boldsymbol{\alpha}|,\varepsilon}^\times\right)\right) \leq C\cdot(1+|\log(\varepsilon)|)$$

for some C depending on k. Thus, for the size of Φ_N^f we obtain with $M=\lceil N^{1/d}\rceil$

$$\begin{aligned}\operatorname{size}(\Phi_N^f) &\leq C\cdot\left(1+\sum_{\boldsymbol{\nu}\leq M}\left(\operatorname{size}(\Phi_\varepsilon^\times)+\operatorname{size}(\varphi_{\boldsymbol{\nu}})+\operatorname{size}(\hat{p}_{\boldsymbol{\nu}})\right)\right) \\ &\leq C\cdot(1+M)^d(1+|\log(\varepsilon)|+C_\varphi) \\ &\leq C\cdot(1+N^{1/d})^d(1+C_\varphi+\log(N)) \\ &\leq CN\log(N),\end{aligned}$$

which concludes the proof. □

Theorem 7.7 shows the convergence rate $(k+s)/d$ for approximating a $C^{k,s}$-function $f:[0,1]^d\to\mathbb{R}$. As long as k is large, in principle we can achieve arbitrarily large (and d-independent if $k\geq d$) convergence rates. Crucially, and in contrast to Theorem 5.22, achieving error $N^{-\frac{k+s}{d}}$ requires the neural networks to be of size $O(N\log(N))$ and depth $O(\log(N))$, i.e. to get more and more accurate approximations, the neural network depth is required to increase.

Remark 7.8. Under the stronger assumption that f is an analytic function (in particular such an f is in C^∞), one can show exponential convergence rates for ReLU networks of the type $\exp(-\beta N^{1/(d+1)})$ for some fixed $\beta>0$ and where N corresponds again to the neural network size (up to logarithmic terms), see [58, 166].

Remark 7.9. Let $L:\boldsymbol{x}\mapsto \boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}:\mathbb{R}^d\to\mathbb{R}^d$ be a bijective affine transformation and set $\Omega := L([0,1]^d)\subseteq\mathbb{R}^d$. Then for a function $f\in C^{k,s}(\Omega)$, by Theorem 7.7 there exists a neural network Φ_N^f such that

$$\begin{aligned}\sup_{\boldsymbol{x}\in\Omega}|f(\boldsymbol{x})-\Phi_N^f(L^{-1}(\boldsymbol{x}))| &= \sup_{\boldsymbol{x}\in[0,1]^d}|f(L(\boldsymbol{x}))-\Phi_N^f(\boldsymbol{x})| \\ &\leq C\|f\circ L\|_{C^{k,s}([0,1]^d)}N^{-\frac{k+s}{d}}.\end{aligned}$$

Since for $\boldsymbol{x}\in[0,1]^d$ holds $|f(L(\boldsymbol{x}))|\leq\sup_{\boldsymbol{y}\in\Omega}|f(\boldsymbol{y})|$ and if $\boldsymbol{0}\neq\boldsymbol{\alpha}\in\mathbb{N}_0^d$ is a multiindex $|D^{\boldsymbol{\alpha}}(f(L(\boldsymbol{x}))|\leq \|A\|_2^{|\boldsymbol{\alpha}|}\sup_{\boldsymbol{y}\in\Omega}|D^{\boldsymbol{\alpha}}f(\boldsymbol{y})|$, we have $\|f\circ L\|_{C^{k,s}([0,1]^d)}\leq(1+\|A\|_2^{k+s})\|f\|_{C^{k,s}(\Omega)}$. Thus the convergence rate $N^{-\frac{k+s}{d}}$ is achieved on every set of the type $L([0,1]^d)$ for an affine map L, and in particular on every hypercube $\times_{j=1}^d[a_j,b_j]$.

Bibliography and further reading

This chapter is based on the seminal 2017 paper by Yarotsky [245], where the construction of approximating the square function, the multiplication, and polynomials (discussed in Sections 7.1 and 7.2) was first introduced and analyzed. The construction relies on the sawtooth function discussed in Section 6.2 and originally introduced by Telgarsky in [227]. Yarotsky's work has since sparked a large body of research, as it allows to lift polynomial approximation theory to neural network classes. Convergence results based on this type of argument include for example [174, 59, 150, 58, 166].

The approximation result derived in Section 7.3 for Hölder continuous functions follows by standard approximation theory for piecewise polynomial functions. We point out that similar results for the approximation of functions in C^k or functions that are analytic can also be shown for other activation function than ReLU; see in particular the works of Mhaskar [144, 145] and Section 6 in Pinkus' Acta Numerica article [176] for sigmoidal and smooth activations. Additionally, the more recent paper [48] specifically addresses the hyperbolic tangent activation. Finally, [81] studies general activation functions that allow for the construction of approximate partitions of unity.

Chapter 8

High-dimensional approximation

In the previous chapters we established convergence rates for the approximation of a function $f : [0,1]^d \to \mathbb{R}$ by a neural network. For example, Theorem 7.7 provides the error bound $\mathcal{O}(N^{-(k+s)/d})$ in terms of the network size N (up to logarithmic terms), where k and s describe the smoothness of f. Achieving an accuracy of $\varepsilon > 0$, therefore, necessitates a network size $N = O(\varepsilon^{-d/(k+s)})$ (according to this bound). Hence, the size of the network needs to increase exponentially in d. This exponential dependence on the dimension d is referred to as the **curse of dimensionality** [16]. For classical smoothness spaces, such exponential d dependence cannot be avoided [16, 52, 164]. However, functions f that are of interest in practice may have additional properties, which allow for better convergence rates.

In this chapter, we discuss three scenarios under which the curse of dimensionality can be mitigated. First, we examine an assumption limiting the behavior of functions in their Fourier domain. This assumption allows for slow but dimension independent approximation rates. Second, we consider functions with a specific compositional structure. Concretely, these functions are constructed by compositions and linear combinations of simple low-dimensional subfunctions. In this case, the curse of dimension is present but only through the input dimension of the subfunctions. Finally, we study the situation, where we still approximate high-dimensional functions, but only care about the approximation accuracy on a lower dimensional submanifold. Here, the approximation rate is goverened by the smoothness and the dimension of the manifold.

8.1 The Barron class

In [10], Barron introduced a set of functions that can be approximated by neural networks without a curse of dimensionality. This set, known as the **Barron class**, is characterized by a specific type of bounded variation. To define it, for $f \in L^1(\mathbb{R}^d)$ denote by

$$\hat{f}(\boldsymbol{w}) \coloneqq \int_{\mathbb{R}^d} f(\boldsymbol{x}) e^{-2\pi \mathrm{i}\, \boldsymbol{w}^\top \boldsymbol{x}} \,\mathrm{d}\boldsymbol{x}$$

its Fourier transform. Then, for $C > 0$ the Barron class is defined as

$$\Gamma_C \coloneqq \left\{ f \in L^1(\mathbb{R}^d) \,\middle|\, \|\hat{f}\|_{L^1(\mathbb{R}^d)} < \infty,\ \int_{\mathbb{R}^d} |2\pi\boldsymbol{\xi}||\hat{f}(\boldsymbol{\xi})| \,\mathrm{d}\boldsymbol{\xi} < C \right\}.$$

We point out that the definition of Γ_C in [10] is more general, but our assumption will simplify some of the arguments. Nonetheless, the following proof is very close to the original result, and the presentation is similar to [175, Section 5]. Theorem 1 in [10] reads as follows.

Theorem 8.1. *Let $\sigma : \mathbb{R} \to \mathbb{R}$ be sigmoidal (see Definition 3.11) and let $f \in \Gamma_C$ for some $C > 0$. Denote by $B_1^d := \{\boldsymbol{x} \in \mathbb{R}^d \,|\, \|\boldsymbol{x}\| \leq 1\}$ the unit ball. Then, for every $c > 4C^2$ and every $N \in \mathbb{N}$ there exists a neural network Φ^f with architecture $(\sigma; d, N, 1)$ such that*

$$\frac{1}{|B_1^d|} \int_{B_1^d} \left| f(\boldsymbol{x}) - \Phi^f(\boldsymbol{x}) \right|^2 \mathrm{d}\boldsymbol{x} \leq \frac{c}{N}, \tag{8.1.1}$$

where $|B_1^d|$ is the Lebesgue measure of B_1^d.

Remark 8.2. The approximation rate on (8.1.1) can be slightly improved under some assumptions on the activation function such as powers of the ReLU, [213].

Importantly, the dimension d does not enter on the right-hand side of (8.1.1), in particular the convergence rate is not directly affected by the dimension, which is in stark contrast to the results of the previous chapters. However, it should be noted, that the constant C_f may still have some inherent d-dependence, see Exercise 8.10.

The proof of Theorem 8.1 is based on a peculiar property of high-dimensional convex sets, which is described by the (approximate) Caratheodory theorem, the original version of which was given in [31]. The more general version stated in the following lemma follows [236, Theorem 0.0.2] and [10, 177]. For its statement recall that $\overline{\mathrm{co}}(G)$ denotes the the closure of the convex hull of G.

Lemma 8.3. *Let H be a Hilbert space, and let $G \subseteq H$ be such that for some $B > 0$ it holds that $\|g\|_H \leq B$ for all $g \in G$. Let $f \in \overline{\mathrm{co}}(G)$. Then, for every $N \in \mathbb{N}$ and every $c > B^2$ there exist $(g_i)_{i=1}^N \subseteq G$ such that*

$$\left\| f - \frac{1}{N} \sum_{i=1}^N g_i \right\|_H^2 \leq \frac{c}{N}. \tag{8.1.2}$$

Proof. Fix $\varepsilon > 0$ and $N \in \mathbb{N}$. Since $f \in \overline{\mathrm{co}}(G)$, there exist coefficients $\alpha_1, \dots, \alpha_m \in [0, 1]$ summing to 1, and linearly independent elements $h_1, \dots, h_m \in G$ such that

$$f^* := \sum_{j=1}^m \alpha_j h_j$$

satisfies $\|f - f^*\|_H < \varepsilon$. We claim that there exists $g_1, \dots, g_N$, each in $\{h_1, \dots, h_m\}$, such that

$$\left\| f^* - \frac{1}{N} \sum_{j=1}^N g_j \right\|_H^2 \leq \frac{B^2}{N}. \tag{8.1.3}$$

Since $\varepsilon > 0$ was arbitrary, this then concludes the proof. Since there exists an isometric isomorphism from $\operatorname{span}\{h_1, \dots, h_m\}$ to $\mathbb{R}^m$, there is no loss of generality in assuming $H = \mathbb{R}^m$ in the following.

Let X_i, $i = 1, \dots, N$, be i.i.d. $\mathbb{R}^m$-valued random variables with

$$\mathbb{P}[X_i = h_j] = \alpha_j \qquad \text{for all } i = 1, \dots, m.$$

In particular $\mathbb{E}[X_i] = \sum_{j=1}^m \alpha_j h_j = f^*$ for each i. Moreover,

$$\begin{aligned}
\mathbb{E}\left[\left\| f^* - \frac{1}{N}\sum_{j=1}^N X_j \right\|^2\right] &= \mathbb{E}\left[\left\| \frac{1}{N}\sum_{j=1}^N (f^* - X_j) \right\|^2\right] \\
&= \frac{1}{N^2}\left[\sum_{j=1}^N \|f^* - X_j\|^2 + \sum_{i\neq j} \langle f^* - X_i, f^* - X_j\rangle\right] \\
&= \frac{1}{N}\mathbb{E}[\|f^* - X_1\|^2] \\
&= \frac{1}{N}\mathbb{E}[\|f^*\| - 2\langle f^*, X_1\rangle + \|X_1\|^2] \\
&= \frac{1}{N}\mathbb{E}[\|X_1\|^2 - \|f^*\|^2] \leq \frac{B^2}{N}. \qquad (8.1.4)
\end{aligned}$$

Here we used that the $(X_i)_{i=1}^N$ are independent, the fact that $\mathbb{E}[X_i] = f^*$, as well as $\mathbb{E}\langle f^* - X_i, f^* - X_j\rangle = 0$ if $i \neq j$. Since the expectation in (8.1.4) is bounded by B^2/N, there must exist at least one realization of the random variables $X_i \in \{h_1, \dots, h_m\}$, denoted as g_i, for which (8.1.3) holds. □

Lemma 8.3 provides a powerful tool: If we want to approximate a function f with a superposition of N elements in a set G, then it is sufficient to show that f can be represented as an arbitrary (infinite) convex combination of elements of G.

Lemma 8.3 suggests that we can prove Theorem 8.1 by showing that each function in Γ_C belongs to the convex hull of neural networks with just a single neuron. We make a small detour before proving this result. We first show that each function $f \in \Gamma_C$ is in the convex hull of affine transforms of Heaviside functions. We define the *set of affine transforms of Heaviside functions* G_C as

$$G_C \coloneqq \left\{ B_1^d \ni \boldsymbol{x} \mapsto \gamma \cdot \mathbb{1}_{\mathbb{R}_+}(\langle \boldsymbol{a}, \boldsymbol{x}\rangle + b) \,\middle|\, \boldsymbol{a} \in \mathbb{R}^d, b \in \mathbb{R}, |\gamma| \leq 2C \right\}.$$

The following lemma, corresponding to [175, Lemma 5.12], provides a link between Γ_C and G_C.

Lemma 8.4. *Let $d \in \mathbb{N}$, $C > 0$ and $f \in \Gamma_C$. Then $f|_{B_1^d} - f(0) \in \overline{\operatorname{co}}(G_C)$, where the closure is taken with respect to the norm*

$$\|g\|_{L^{2,\diamond}(B_1^d)} \coloneqq \left(\frac{1}{|B_1^d|} \int_{B_1^d} |g(\boldsymbol{x})|^2 \, \mathrm{d}\boldsymbol{x} \right)^{1/2}.$$

Proof. Since $f \in \Gamma_C$, we have that $f,\ \hat{f} \in L^1(\mathbb{R}^d)$. Hence, we can apply the inverse Fourier transform and get the following computation:

$$\begin{aligned} f(\boldsymbol{x}) - f(0) &= \int_{\mathbb{R}^d} \hat{f}(\boldsymbol{\xi}) \left(e^{2\pi i \langle \boldsymbol{x}, \boldsymbol{\xi} \rangle} - 1 \right) \mathrm{d}\boldsymbol{\xi} \\ &= \int_{\mathbb{R}^d} \left| \hat{f}(\boldsymbol{\xi}) \right| \left(e^{2\pi i \langle \boldsymbol{x}, \boldsymbol{\xi} \rangle + i\kappa(\boldsymbol{\xi})} - e^{i\kappa(\boldsymbol{\xi})} \right) \mathrm{d}\boldsymbol{\xi} \\ &= \int_{\mathbb{R}^d} \left| \hat{f}(\boldsymbol{\xi}) \right| \left(\cos(2\pi \langle \boldsymbol{x}, \boldsymbol{\xi} \rangle + \kappa(\boldsymbol{\xi})) - \cos(\kappa(\boldsymbol{\xi})) \right) \mathrm{d}\boldsymbol{\xi}, \end{aligned}$$

where $\kappa(\boldsymbol{\xi})$ is the phase of $\hat{f}(\boldsymbol{\xi})$ and the last inequality follows since f is real-valued.

To use the fact that f has a bounded Fourier moment, we reformulate the integral as

$$\begin{aligned} &\int_{\mathbb{R}^d} \left| \hat{f}(\boldsymbol{\xi}) \right| \left(\cos(2\pi \langle x, \boldsymbol{\xi} \rangle + \kappa(\boldsymbol{\xi})) - \cos(\kappa(\boldsymbol{\xi})) \right) \mathrm{d}\boldsymbol{\xi} \\ &\quad = \int_{\mathbb{R}^d} \frac{\left(\cos(2\pi \langle x, \boldsymbol{\xi} \rangle + \kappa(\boldsymbol{\xi})) - \cos(\kappa(\boldsymbol{\xi})) \right)}{|2\pi\boldsymbol{\xi}|} |2\pi\boldsymbol{\xi}| \left| \hat{f}(\boldsymbol{\xi}) \right| \mathrm{d}\boldsymbol{\xi}. \end{aligned}$$

We define a new measure Λ with density

$$\mathrm{d}\Lambda(\boldsymbol{\xi}) \coloneqq \frac{1}{C} |2\pi\boldsymbol{\xi}| |\hat{f}(\boldsymbol{\xi})| \, \mathrm{d}\boldsymbol{\xi}.$$

Since $f \in \Gamma_C$, it follows that Λ is a probability measure on $\mathbb{R}^d$. Now we have that

$$f(\boldsymbol{x}) - f(0) = C \int_{\mathbb{R}^d} \frac{\left(\cos(2\pi \langle \boldsymbol{x}, \boldsymbol{\xi} \rangle + \kappa(\boldsymbol{\xi})) - \cos(\kappa(\boldsymbol{\xi})) \right)}{|2\pi\boldsymbol{\xi}|} \, \mathrm{d}\Lambda(\boldsymbol{\xi}). \tag{8.1.5}$$

Next, we would like to replace the integral of (8.1.5) by an appropriate finite sum.

The cosine function is 1-Lipschitz. Hence, we note that $\boldsymbol{\xi} \mapsto q_{\boldsymbol{x}}(\boldsymbol{\xi}) \coloneqq (\cos(2\pi \langle \boldsymbol{x}, \boldsymbol{\xi} \rangle + \kappa(\boldsymbol{\xi})) - \cos(\kappa(\boldsymbol{\xi})))/|2\pi\boldsymbol{\xi}|$ is bounded by 1. In addition, it is easy to see that $q_{\boldsymbol{x}}$ is well-defined and continuous even in the origin.

Therefore, the integral (8.1.5) can be approximated by a Riemann sum, i.e.,

$$\left| C \int_{\mathbb{R}^d} q_{\boldsymbol{x}}(\boldsymbol{\xi}) \, \mathrm{d}\Lambda(\boldsymbol{\xi}) - C \sum_{\theta \in \frac{1}{n}\mathbb{Z}^d} q_{\boldsymbol{x}}(\theta) \cdot \Lambda(I_\theta) \right| \to 0, \tag{8.1.6}$$

where $I_\theta \coloneqq [0, 1/n)^d + \theta$.

Since $f(\boldsymbol{x}) - f(0)$ is continuous and thus bounded on B_1^d, we have by the dominated convergence theorem that

$$\frac{1}{|B_1^d|} \int_{B_1^d} \left| f(\boldsymbol{x}) - f(0) - C \sum_{\theta \in \frac{1}{n}\mathbb{Z}^d} q_{\boldsymbol{x}}(\theta) \cdot \Lambda(I_\theta) \right|^2 \mathrm{d}\boldsymbol{x} \to 0. \tag{8.1.7}$$

Since $\sum_{\theta \in \frac{1}{n}\mathbb{Z}^d} \Lambda(I_\theta) = \Lambda(\mathbb{R}^d) = 1$, we conclude that $f(\boldsymbol{x}) - f(0)$ is in the $L^{2,\diamond}(B_1^d)$ closure of convex combinations of functions of the form

$$\boldsymbol{x} \mapsto g_\theta(\boldsymbol{x}) \coloneqq \alpha_\theta q_{\boldsymbol{x}}(\theta),$$

for $\theta \in \mathbb{R}^d$ and $0 \leq \alpha_\theta \leq C$.

Now we only need to prove that each g_θ is in $\overline{\mathrm{co}}(G_C)$. By setting $z = \langle \boldsymbol{x}, \theta/|\theta| \rangle$, we observe that the result follows if the map

$$[-1,1] \ni z \mapsto \alpha_\theta \frac{\cos(2\pi|\theta|z + \kappa(\theta)) - \cos(\kappa(\theta))}{|2\pi\theta|} =: \tilde{g}_\theta(z),$$

can be approximated arbitrarily well by convex combinations of functions of the form

$$[-1,1] \ni z \mapsto \gamma \mathbb{1}_{\mathbb{R}_+}\left(a'z + b'\right), \tag{8.1.8}$$

where a', $b' \in \mathbb{R}$ and $|\gamma| \leq 2C$.

We define, for $T \in \mathbb{N}$,

$$g_{T,+} \coloneqq \sum_{i=1}^{T} \frac{\left|\tilde{g}_\theta\left(\frac{i}{T}\right) - \tilde{g}_\theta\left(\frac{i-1}{T}\right)\right|}{2C} \left(2C\mathrm{sign}\left(\tilde{g}_\theta\left(\frac{i}{T}\right) - \tilde{g}_\theta\left(\frac{i-1}{T}\right)\right) \mathbb{1}_{\mathbb{R}_+}\left(x - \frac{i}{T}\right)\right),$$

$$g_{T,-} \coloneqq \sum_{i=1}^{T} \frac{\left|\tilde{g}_\theta\left(-\frac{i}{T}\right) - \tilde{g}_\theta\left(\frac{1-i}{T}\right)\right|}{2C} \left(2C\mathrm{sign}\left(\tilde{g}_\theta\left(-\frac{i}{T}\right) - \tilde{g}_\theta\left(\frac{1-i}{T}\right)\right) \mathbb{1}_{\mathbb{R}_+}\left(-x + \frac{i}{T}\right)\right).$$

Per construction, $g_{T,-} + g_{T,+}$ converges to $\tilde{g}_\theta$ for $T \to \infty$. Moreover, $\|\tilde{g}_\theta'\|_{L^\infty(\mathbb{R})} \leq C$ and hence

$$\begin{aligned}
&\sum_{i=1}^{T} \frac{|\tilde{g}_\theta(i/T) - \tilde{g}_\theta((i-1)/T)|}{2C} + \sum_{i=1}^{T} \frac{|\tilde{g}_\theta(-i/T) - \tilde{g}_\theta((1-i)/T)|}{2C} \\
&\leq \frac{2}{2CT} \sum_{i=1}^{T} \|\tilde{g}_\theta'\|_{L^\infty(\mathbb{R})} \leq 1.
\end{aligned}$$

We conclude that $g_{T,-} + g_{T,+}$ is a convex combination of functions of the form (8.1.8). Hence, $\tilde{g}_\theta$ can be arbitrarily well approximated by convex combinations of the form (8.1.8). Therefore $g_\theta \in \overline{\mathrm{co}}(G_C)$. Finally, (8.1.7) yields that $f - f(0) \in \overline{\mathrm{co}}(G_C)$. □

We now have all tools to complete the proof of Theorem 8.1.

of Theorem 8.1. Let $f \in \Gamma_C$. By Lemma 8.4

$$f|_{B_1^d} - f(0) \in \overline{\mathrm{co}}(G_C).$$

It is not hard to see that for every $g \in G_C$ holds $\|g\|_{L^{2,\diamond}(B_1^d)} \leq 2C$. Applying Lemma 8.3 with the Hilbert space $L^{2,\diamond}(B_1^d)$, we get that for every $N \in \mathbb{N}$ there exist $|\gamma_i| \leq 2C$, $\boldsymbol{a}_i \in \mathbb{R}^d$, $b_i \in \mathbb{R}$, for $i = 1, \ldots, N$, so that

$$\frac{1}{|B_1^d|} \int_{B_1^d} \left| f(\boldsymbol{x}) - f(0) - \sum_{i=1}^{N} \gamma_i \mathbb{1}_{\mathbb{R}_+}(\langle \boldsymbol{a}_i, x \rangle + b_i) \right|^2 \mathrm{d}\boldsymbol{x} \leq \frac{4C^2}{N}.$$

By Exercise 3.24, it holds that $\sigma(\lambda \cdot) \to \mathbb{1}_{\mathbb{R}_+}$ for $\lambda \to \infty$ almost everywhere. Thus, for every $\delta > 0$ there exist $\tilde{\boldsymbol{a}}_i$, $\tilde{b}_i$, $i = 1, \ldots, N$, so that

$$\frac{1}{|B_1^d|}\int_{B_1^d}\left|f(\boldsymbol{x})-f(0)-\sum_{i=1}^{N}\gamma_i\sigma\left(\langle\tilde{\boldsymbol{a}}_i,\boldsymbol{x}\rangle+\tilde{b}_i\right)\right|^2\,\mathrm{d}\boldsymbol{x}\leq\frac{4C^2}{N}+\delta.$$

The result follows by observing that

$$\sum_{i=1}^{N}\gamma_i\sigma\left(\langle\tilde{\boldsymbol{a}}_i,\boldsymbol{x}\rangle+\tilde{b}_i\right)+f(0)$$

is a neural network with architecture $(\sigma;d,N,1)$. □

The dimension-independent approximation rate of Theorem 8.1 may seem surprising, especially in comparison to the results in Chapters 4 and 5. However, this can be explained by recognizing that the assumption of a finite Fourier moment is effectively a *dimension-dependent regularity assumption*. Indeed, the condition becomes more restrictive in higher dimensions and hence the complexity of Γ_C does not grow with the dimension.

To further explain this, let us relate the Barron class to classical function spaces. In [10, Section II] it was observed that a sufficient condition is that all derivatives of order up to $\lfloor d/2\rfloor+2$ are square-integrable. In other words, if f belongs to the Sobolev space $H^{\lfloor d/2\rfloor+2}(\mathbb{R}^d)$, then f is a Barron function. Importantly, the functions must become smoother, as the dimension increases. This assumption would also imply an approximation rate of $N^{-1/2}$ in the L^2 norm by sums of at most N B-splines, see [168, 52]. However, in such estimates some constants may still depend exponentially on d, whereas all constants in Theorem 8.1 are controlled independently of d.

Another notable aspect of the approximation of Barron functions is that the absolut values of the weights other than the output weights are not bounded by a constant. To see this, we refer to (8.1.6), where arbitrarily large θ need to be used. While Γ_C is a compact set, the set of neural networks of the specified architecture for a fixed $N\in\mathbb{N}$ is not parameterized with a compact parameter set. In a certain sense, this is reminiscent of Proposition 3.19 and Theorem 3.20, where arbitrarily strong approximation rates where achieved by using a very complex activation function and a non-compact parameter space.

8.2 Functions with compositionality structure

As a next instance of types of functions for which the curse of dimensionality can be overcome, we study functions with compositional structure. In words, this means that we study high-dimensional functions that are constructed by composing many low-dimensional functions. This point of view was proposed in [178]. Note that this can be a realistic assumption in many cases, such as for sensor networks, where local information is first aggregated in smaller clusters of sensors before some information is sent to a processing unit for further evaluation.

We introduce a model for compositional functions next. Consider a directed acyclic graph $\mathcal{G}$ with M vertices $\eta_1,\dots,\eta_M$ such that

- exactly d vertices, $\eta_1,\dots,\eta_d$, have no ingoing edge,
- each vertex has at most $m\in\mathbb{N}$ ingoing edges,
- exactly one vertex, η_M, has no outgoing edge.

With each vertex η_j for $j > d$ we associate a function $f_j : \mathbb{R}^{d_j} \to \mathbb{R}$. Here d_j denotes the cardinality of the set S_j, which is defined as the set of indices i corresponding to vertices η_i for which we have an edge from η_i to η_j. Without loss of generality, we assume that $m \geq d_j = |S_j| \geq 1$ for all $j > d$. Finally, we let

$$F_j := x_j \quad \text{for all} \quad j \leq d \tag{8.2.1a}$$

and[1]

$$F_j := f_j((F_i)_{i \in S_j}) \quad \text{for all} \quad j > d. \tag{8.2.1b}$$

Then $F_M(x_1, \dots, x_d)$ is a function from $\mathbb{R}^d \to \mathbb{R}$. Assuming

$$\|f_j\|_{C^{k,s}(\mathbb{R}^{d_j})} \leq 1 \quad \text{for all} \quad j = d+1, \dots, M, \tag{8.2.2}$$

we denote the set of all functions of the type F_M by $\mathcal{F}^{k,s}(m, d, M)$. Figure 8.1 shows possible graphs of such functions.

Clearly, for $s = 0$, $\mathcal{F}^{k,0}(m, d, M) \subseteq C^k(\mathbb{R}^d)$ since the composition of functions in C^k belongs again to C^k. A direct application of Theorem 7.7 allows to approximate $F_M \in \mathcal{F}^k(m, d, M)$ with a neural network of size $O(N \log(N))$ and error $O(N^{-\frac{k}{d}})$. Since each f_j depends only on m variables, intuitively we expect an error convergence of type $O(N^{-\frac{k}{m}})$ with the constant somehow depending on the number M of vertices. To show that this is actually possible, in the following we associate with each node η_j a depth $l_j \geq 0$, such that l_j is the maximum number of edges connecting η_j to one of the nodes $\{\eta_1, \dots, \eta_d\}$.

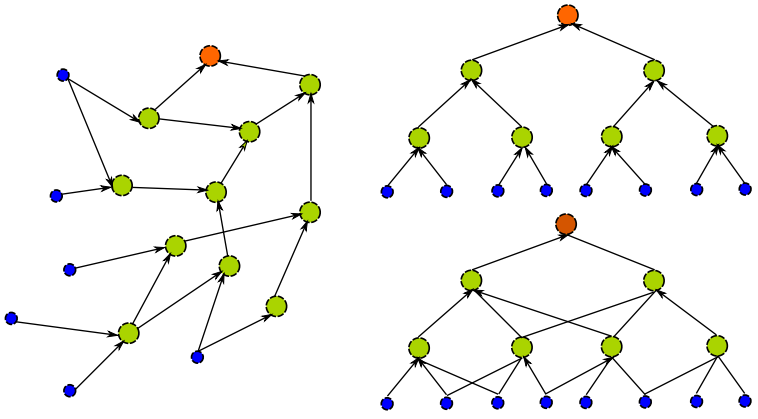


Figure 8.1: Three types of graphs that could be the basis of compositional functions. The associated functions are composed of two or three-dimensional functions only.

Proposition 8.5. *Let k, m, d, $M \in \mathbb{N}$ and $s > 0$. Let $F_M \in \mathcal{F}^{k,s}(m, d, M)$. Then there exists a constant $C = C(m, k+s, M)$ such that for every $N \in \mathbb{N}$ there exists a ReLU neural network Φ^{F_M}*

[1]The ordering of the inputs $(F_i)_{i \in S_j}$ in (8.2.1b) is arbitrary but considered fixed throughout.

such that

$$\operatorname{size}(\Phi^{F_M}) \leq CN\log(N), \qquad \operatorname{depth}(\Phi^{F_M}) \leq C\log(N)$$

and

$$\sup_{\boldsymbol{x}\in[0,1]^d} |F_M(\boldsymbol{x}) - \Phi^{F_M}(\boldsymbol{x})| \leq N^{-\frac{k+s}{m}}.$$

Proof. Throughout this proof we assume without loss of generality that the indices follow a topological ordering, i.e., they are ordered such that $S_j \subseteq \{1,\dots,j-1\}$ for all j (i.e. the inputs of vertex η_j can only be vertices η_i with $i<j$).

Step 1. First assume that $\hat{f}_j$ are functions such that

$$|f_j(\boldsymbol{x}) - \hat{f}_j(\boldsymbol{x})| \leq \delta_j := \varepsilon\cdot(2m)^{-(M+1-j)} \quad \text{for all} \quad \boldsymbol{x}\in[-2,2]^{d_j}. \tag{8.2.3}$$

Let $\hat{F}_j$ be defined as in (8.2.1), but with all f_j in (8.2.1b) replaced by $\hat{f}_j$. We now check the error of the approximation $\hat{F}_M$ to F_M. To do so we proceed by induction over j and show that for all $\boldsymbol{x}\in[-1,1]^d$

$$|F_j(\boldsymbol{x}) - \hat{F}_j(\boldsymbol{x})| \leq (2m)^{-(M-j)}\varepsilon. \tag{8.2.4}$$

Note that due to $\|f_j\|_{C^k} \leq 1$ we have $|F_j(\boldsymbol{x})| \leq 1$ and thus (8.2.4) implies in particular that $\hat{F}_j(\boldsymbol{x}) \in [-2,2]$.

For $j=1$ it holds $F_1(x_1) = \hat{F}_1(x_1) = x_1$, and thus (8.2.4) is valid for all $x_1\in[-1,1]$. For the induction step, for all $\boldsymbol{x}\in[-1,1]^d$ by (8.2.3) and the induction hypothesis

$$\begin{aligned}
|F_j(\boldsymbol{x}) - \hat{F}_j(\boldsymbol{x})| &= |f_j((F_i)_{i\in S_j}) - \hat{f}_j((\hat{F}_i)_{i\in S_j})| \\
&= |f_j((F_i)_{i\in S_j}) - f_j((\hat{F}_i)_{i\in S_j})| + |f_j((\hat{F}_i)_{i\in S_j}) - \hat{f}_j((\hat{F}_i)_{i\in S_j})| \\
&\leq \sum_{i\in S_j} |F_i - \hat{F}_i| + \delta_j \\
&\leq m\cdot(2m)^{-(M-(j-1))}\varepsilon + (2m)^{-(M+1-j)}\varepsilon \\
&\leq (2m)^{-(M-j)}\varepsilon.
\end{aligned}$$

Here we used that $|\frac{d}{dx_r} f_j((x_i)_{i\in S_j})| \leq 1$ for all $r\in S_j$ so that

$$\begin{aligned}
|f_j((x_i)_{i\in S_j}) - f_j((y_i)_{i\in S_j})| &\leq \sum_{r\in S_j} |f((x_i)_{\substack{i\in S_j\\ i\leq r}}, (y_i)_{\substack{i\in S_j\\ i>r}}) - f((x_i)_{\substack{i\in S_j\\ i<r}}, (y_i)_{\substack{i\in S_j\\ i\geq r}})| \\
&\leq \sum_{r\in S_j} |x_r - y_r|.
\end{aligned}$$

This shows that (8.2.4) holds, and thus for all $\boldsymbol{x}\in[-1,1]^d$

$$|F_M(\boldsymbol{x}) - \hat{F}_M(\boldsymbol{x})| \leq \varepsilon.$$

Step 2. We sketch a construction, of how to write $\hat{F}_M$ from Step 1 as a neural network Φ^{F_M} of the claimed size and depth bounds. Fix $N \in \mathbb{N}$ and let

$$N_j := \lceil N(2m)^{\frac{m}{k+s}(M+1-j)} \rceil.$$

By Theorem 7.7, since $d_j \leq m$, we can find a neural network Φ^{f_j} satisfying

$$\sup_{\boldsymbol{x} \in [-2,2]^{d_j}} |f_j(\boldsymbol{x}) - \Phi^{f_j}(\boldsymbol{x})| \leq N_j^{-\frac{k+s}{m}} \leq N^{-\frac{k+s}{m}} (2m)^{-(M+1-j)} \tag{8.2.5}$$

and

$$\mathrm{size}(\Phi^{f_j}) \leq C N_j \log(N_j) \leq C N (2m)^{\frac{m(M+1-j)}{k+s}} \left(\log(N) + \log(2m) \frac{m(M+1-j)}{k+s} \right)$$

as well as

$$\mathrm{depth}(\Phi^{f_j}) \leq C \cdot \left(\log(N) + \log(2m) \frac{m(M+1-j)}{k+s} \right).$$

Then

$$\begin{aligned} \sum_{j=1}^{n} \mathrm{size}(\Phi^{f_j}) \leq 2CN \log(N) \sum_{j=1}^{M} (2m)^{\frac{m(M+1-j)}{k+s}} &\leq 2CN \log(N) \sum_{j=1}^{M} \left((2m)^{\frac{m}{k+s}} \right)^j \\ &\leq 2CN \log(N) (2m)^{\frac{m(M+1)}{k+s}}. \end{aligned}$$

Here we used $\sum_{j=1}^{M} a^j \leq \int_1^{M+1} \exp(\log(a)x) \, \mathrm{d}x \leq \frac{1}{\log(a)} a^{M+1}$.

The function $\hat{F}_M$ from Step 1 then will yield error $N^{-\frac{k+s}{m}}$ by (8.2.3) and (8.2.5). We observe that $\hat{F}_M$ can be constructed inductively as a neural network Φ^{F_M} by propagating all values $\Phi^{F_1}, \ldots, \hat{\Phi}^{F_j}$ to all consecutive layers using identity neural networks and then using the outputs of $(\Phi^{F_i})_{i \in S_{j+1}}$ as input to $\Phi^{f_{j+1}}$. The depth of this neural network is bounded by

$$\sum_{j=1}^{M} \mathrm{depth}(\Phi^{f_j}) = O(M \log(N)).$$

We have at most $\sum_{j=1}^{M} |S_j| \leq mM$ values which need to be propagated through these $O(M \log(N))$ layers, amounting to an overhead $O(mM^2 \log(N)) = O(\log(N))$ for the identity neural networks. In all the neural network size is thus $O(N \log(N))$. □

Remark 8.6. From the proof we observe that the constant C in Proposition 8.5 behaves like $O((2m)^{\frac{m(M+1)}{k+s}})$.

8.3 Functions on manifolds

Another instance in which the curse of dimension can be mitigated, is if the input to the network belongs to $\mathbb{R}^d$, but stems from an m-dimensional manifold $\mathcal{M} \subseteq \mathbb{R}^d$. If we only measure the

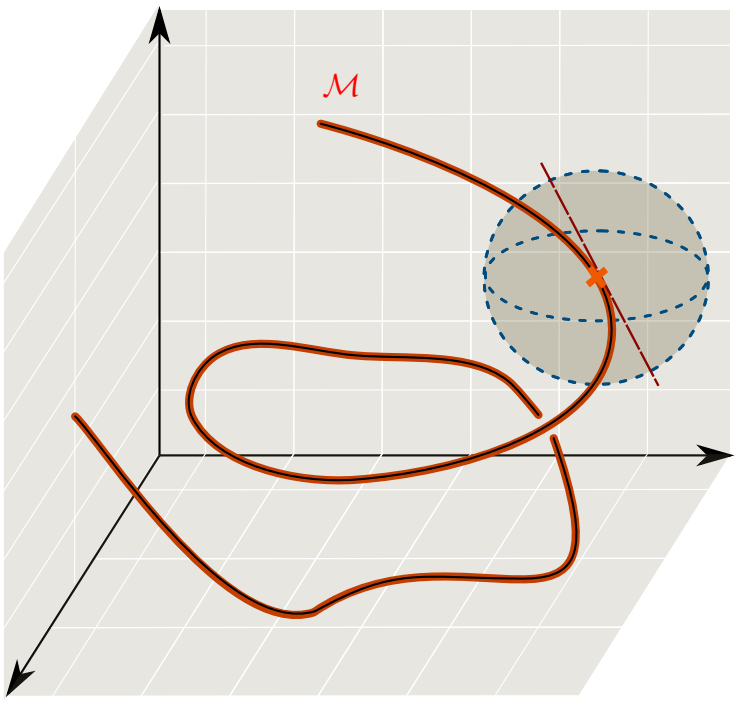


Figure 8.2: One-dimensional sub-manifold of three-dimensional space. At the orange point, we depict a ball and the tangent space of the manifold.

approximation error on $\mathcal{M}$, then we can again show that it is m rather than d that determines the rate of convergence.

To explain the idea, we assume in the following that $\mathcal{M}$ is a smooth, compact m-dimensional manifold in $\mathbb{R}^d$. Moreover, we suppose that there exists $\delta > 0$ and finitely many points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_M \in \mathcal{M}$ such that the δ-balls $B_{\delta/2}(\boldsymbol{x}_i) := \{\boldsymbol{y} \in \mathbb{R}^d \,|\, \|\boldsymbol{y} - \boldsymbol{x}\|_2 < \delta/2\}$ for $j = 1, \dots, M$ cover $\mathcal{M}$ (for every $\delta > 0$ such $\boldsymbol{x}_i$ exist since $\mathcal{M}$ is compact). Moreover, denoting by $T_{\boldsymbol{x}}\mathcal{M} \simeq \mathbb{R}^m$ the tangential space of $\mathcal{M}$ at $\boldsymbol{x}$, we assume $\delta > 0$ to be so small that the orthogonal projection

$$\pi_j : B_\delta(\boldsymbol{x}_j) \cap \mathcal{M} \to T_{\boldsymbol{x}_j}\mathcal{M} \tag{8.3.1}$$

is injective, the set $\pi_j(B_\delta(\boldsymbol{x}_j) \cap \mathcal{M}) \subseteq T_{\boldsymbol{x}_j}\mathcal{M}$ has C^∞ boundary, and the inverse projection

$$\pi_j^{-1} : \pi_j(B_\delta(\boldsymbol{x}_j) \cap \mathcal{M}) \to \mathcal{M} \tag{8.3.2}$$

is C^∞ (this is possible because $\mathcal{M}$ is a smooth manifold). A visualization of this assumption is shown in Figure 8.2.

Note that π_j in (8.3.1) is a linear map, whereas π_j^{-1} in (8.3.2) is in general non-linear.

For a function $f : \mathcal{M} \to \mathbb{R}$ and $\boldsymbol{x} \in B_\delta(\boldsymbol{x}_j) \cap \mathcal{M}$ we can then write

$$f(\boldsymbol{x}) = f(\pi_j^{-1}(\pi_j(\boldsymbol{x}))) = f_j(\pi_j(\boldsymbol{x}))$$

where

$$f_j := f \circ \pi_j^{-1} : \pi_j(B_\delta(\boldsymbol{x}_j) \cap \mathcal{M}) \to \mathbb{R}.$$

In the following, for $f : \mathcal{M} \to \mathbb{R}$, $k \in \mathbb{N}_0$, and $s \in [0, 1)$ we let

$$\|f\|_{C^{k,s}(\mathcal{M})} := \sup_{j=1,\dots,M} \|f_j\|_{C^{k,s}(\pi_j(B_\delta(\boldsymbol{x}_j)\cap\mathcal{M}))}.$$

We now state the main result of this section.

Proposition 8.7. *Let $d, k \in \mathbb{N}$, $s \geq 0$, and let $\mathcal{M}$ be a smooth, compact m-dimensional manifold in $\mathbb{R}^d$. Then there exists a constant $C > 0$ such that for all $f \in C^{k,s}(\mathcal{M})$ and every $N \in \mathbb{N}$ there exists a ReLU neural network Φ_N^f such that $\operatorname{size}(\Phi_N^f) \leq CN\log(N)$, $\operatorname{depth}(\Phi_N^f) \leq C\log(N)$ and*

$$\sup_{\boldsymbol{x}\in\mathcal{M}} |f(\boldsymbol{x}) - \Phi_N^f(\boldsymbol{x})| \leq C\|f\|_{C^{k,s}(\mathcal{M})} N^{-\frac{k+s}{m}}.$$

Proof. Since $\mathcal{M}$ is compact there exists $A > 0$ such that $\mathcal{M} \subseteq [-A, A]^d$. Similar as in the proof of Theorem 7.7, we consider a uniform mesh with nodes $\{-A + 2A\frac{\boldsymbol{\nu}}{n} \,|\, \boldsymbol{\nu} \leq n\}$, and the corresponding piecewise linear basis functions forming the partition of unity $\sum_{\boldsymbol{\nu}\leq} \varphi_{\boldsymbol{\nu}} \equiv 1$ on $[-A, A]^d$ where $\operatorname{supp}\varphi_{\boldsymbol{\nu}} \leq \{\boldsymbol{y} \in \mathbb{R}^d \,|\, \|\frac{\boldsymbol{\nu}}{n} - \boldsymbol{y}\|_\infty \leq \frac{A}{n}\}$. Let $\delta > 0$ be such as in the beginning of this section. Since $\mathcal{M}$ is covered by the balls $(B_{\delta/2}(\boldsymbol{x}_j))_{j=1}^M$, fixing $n \in \mathbb{N}$ large enough, for each $\boldsymbol{\nu}$ such that $\operatorname{supp}\varphi_{\boldsymbol{\nu}} \cap \mathcal{M} \neq \emptyset$ there exists $j(\boldsymbol{\nu}) \in \{1, \dots, M\}$ such that $\operatorname{supp}\varphi_{\boldsymbol{\nu}} \subseteq B_\delta(\boldsymbol{x}_{j(\boldsymbol{\nu})})$ and we set $I_j := \{\boldsymbol{\nu} \leq M \,|\, j = j(\boldsymbol{\nu})\}$. Then we have for all $\boldsymbol{x} \in \mathcal{M}$

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{\nu}\leq n} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) f_j(\pi_j(\boldsymbol{x})) = \sum_{j=1}^M \sum_{\boldsymbol{\nu}\in I_j} \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) f_j(\pi_j(\boldsymbol{x})). \tag{8.3.3}$$

Next, we approximate the functions f_j. Let C_j be the smallest (m-dimensional) cube in $T_{\boldsymbol{x}_j}\mathcal{M} \simeq \mathbb{R}^m$ such that $\pi_j(B_\delta(\boldsymbol{x}_j) \cap \mathcal{M}) \subseteq C_j$. The function $\hat{f}_j$ can be extended to a function on C_j (we will use the same notation for this extension) such that

$$\|f\|_{C^{k,s}(C_j)} \leq C\|f\|_{C^{k,s}(\pi_j(B_\delta(\boldsymbol{x}_j)\cap\mathcal{M}))},$$

for some constant depending on $\pi_j(B_\delta(\boldsymbol{x}_j) \cap \mathcal{M})$ but independent of f. Such an extension result can, for example, be found in [216, Chapter VI]. By Theorem 7.7 (also see Remark 7.9), there exists a neural network $\hat{f}_j : C_j \to \mathbb{R}$ such that

$$\sup_{\boldsymbol{x}\in C_j} |f_j(\boldsymbol{x}) - \hat{f}_j(\boldsymbol{x})| \leq CN^{-\frac{k+s}{m}} \tag{8.3.4}$$

and

$$\operatorname{size}(\hat{f}_j) \leq CN\log(N), \qquad \operatorname{depth}(\hat{f}_j) \leq C\log(N).$$

To approximate f in (8.3.3) we now let with $\varepsilon := N^{-\frac{k+s}{d}}$

$$\Phi_N := \sum_{j=1}^M \sum_{\boldsymbol{\nu}\in I_j} \Phi_\varepsilon^\times(\varphi_{\boldsymbol{\nu}}, \hat{f}_i \circ \pi_j),$$

where we note that π_j is linear and thus $\hat{f}_j \circ \pi_j$ can be expressed by a neural network. First let us estimate the error of this approximation. For $\boldsymbol{x} \in \mathcal{M}$

$$\begin{aligned}
|f(\boldsymbol{x}) - \Phi_N(\boldsymbol{x})| &\leq \sum_{j=1}^{M} \sum_{\boldsymbol{\nu} \in I_j} |\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) f_j(\pi_j(\boldsymbol{x})) - \Phi_\varepsilon^\times(\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}), \hat{f}_j(\pi_j(\boldsymbol{x})))| \\
&\leq \sum_{j=1}^{M} \sum_{\boldsymbol{\nu} \in I_j} \big(|\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) f_j(\pi_j(\boldsymbol{x})) - \varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) f_j(\pi_j(\boldsymbol{x}))| \\
&\qquad + |\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}) f_j(\pi_j(\boldsymbol{x})) - \Phi_\varepsilon^\times(\varphi_{\boldsymbol{\nu}}(\boldsymbol{x}), \hat{f}_j(\pi_j(\boldsymbol{x})))|\big) \\
&\leq \sup_{i \leq M} \|f_i - \hat{f}_i\|_{L^\infty(C_i)} \sum_{j=1}^{M} \sum_{\boldsymbol{\nu} \in I_j} |\varphi_{\boldsymbol{\nu}}(\boldsymbol{x})| + \sum_{j=1}^{M} \sum_{\{\boldsymbol{\nu} \in I_j \,|\, \boldsymbol{x} \in \operatorname{supp} \varphi_{\boldsymbol{\nu}}\}} \varepsilon \\
&\leq C N^{-\frac{k+s}{m}} + d\varepsilon \leq C N^{-\frac{k+s}{m}},
\end{aligned}$$

where we used that $\boldsymbol{x}$ can be in the support of at most d of the $\varphi_{\boldsymbol{\nu}}$, and where C is a constant depending on d and $\mathcal{M}$.

Finally, let us bound the size and depth of this approximation. Using $\operatorname{size}(\varphi_{\boldsymbol{\nu}}) \leq C$, $\operatorname{depth}(\varphi_{\boldsymbol{\nu}}) \leq C$ (see (5.3.12)) and $\operatorname{size}(\Phi_\varepsilon^\times) \leq C \log(\varepsilon) \leq C \log(N)$ and $\operatorname{depth}(\Phi_\varepsilon^\times) \leq C \operatorname{depth}(\varepsilon) \leq C \log(N)$ (see Lemma 7.3) we find

$$\begin{aligned}
\sum_{j=1}^{M} \sum_{\boldsymbol{\nu} \in I_j} \Big(\operatorname{size}(\Phi_\varepsilon^\times) + \operatorname{size}(\varphi_{\boldsymbol{\nu}}) + \operatorname{size}(\hat{f}_i \circ \pi_j)\Big) &\leq \sum_{j=1}^{M} \sum_{\boldsymbol{\nu} \in I_j} C \log(N) + C + C N \log(N) \\
&= O(N \log(N)),
\end{aligned}$$

which implies the bound on $\operatorname{size}(\Phi_N)$. Moreover,

$$\begin{aligned}
\operatorname{depth}(\Phi_N) &\leq \operatorname{depth}(\Phi_\varepsilon^\times) + \max\Big\{\operatorname{depth}(\varphi_{\boldsymbol{\nu}}, \hat{f}_j)\Big\} \\
&\leq C \log(N) + \log(N) = O(\log(N)).
\end{aligned}$$

This completes the proof. □

Bibliography and further reading

The ideas of Section 8.1 were originally developed in [10], with an extension to L^∞ approximation provided in [9]. These arguments can be extended to yield dimension-independent approximation rates for high-dimensional discontinuous functions, provided the discontinuity follows a Barron function, as shown in [175]. The Barron class has been generalized in various ways, as discussed in [138, 137, 239, 240, 11].

The compositionality assumption of Section 8.2 was discussed in the form presented in [178]. An alternative approach, known as the hierarchical composition/interaction model, was studied in [119].

The manifold assumption discussed in Section 8.3 is frequently found in the literature, with notable examples including [211, 40, 35, 203, 156, 118].

Another prominent direction, omitted in this chapter, pertains to scientific machine learning. High-dimensional functions often arise from (parametric) PDEs, which have a rich literature describing their properties and structure. Various results have shown that neural networks can leverage the inherent low-dimensionality known to exist in such problems. Efficient approximation of certain classes of high-dimensional (or even infinite-dimensional) analytic functions, ubiquitous in parametric PDEs, has been verified in [208, 209]. Further general analyses for high-dimensional parametric problems can be found in [167, 122], and results exploiting specific structural conditions of the underlying PDEs, e.g., in [125, 198]. Additionally, [58, 150, 166] provide results regarding fast convergence for certain smooth functions in potentially high but finite dimensions.

For high-dimensional PDEs, elliptic problems have been addressed in [78], linear and semilinear parabolic evolution equations have been explored in [79, 71, 100], and stochastic differential equations in [109, 80].

Exercises

Exercise 8.8. Let $C > 0$ and $d \in \mathbb{N}$. Show that, if $g \in \Gamma_C$, then

$$a^{-d} g\left(a(\cdot - \boldsymbol{b})\right) \in \Gamma_C,$$

for every $a \in \mathbb{R}_+$, $\boldsymbol{b} \in \mathbb{R}^d$.

Exercise 8.9. Let $C > 0$ and $d \in \mathbb{N}$. Show that, for $g_i \in \Gamma_C$, $i = 1, \dots, m$ and $c = (c_i)_{i=1}^m$ it holds that

$$\sum_{i=1}^m c_i g_i \in \Gamma_{\|c\|_1 C}.$$

Exercise 8.10. For every $d \in \mathbb{N}$ the function $f(\boldsymbol{x}) := \exp(-\|\boldsymbol{x}\|_2^2/2)$, $\boldsymbol{x} \in \mathbb{R}^d$, belongs to Γ_d. It holds $C_f = O(\sqrt{d})$, for $d \to \infty$.

Exercise 8.11. Let $d \in \mathbb{N}$, and let $f(\boldsymbol{x}) = \sum_{i=1}^\infty c_i \sigma_{\mathrm{ReLU}}(\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle + b_i)$ for $\boldsymbol{x} \in \mathbb{R}^d$ with $\|\boldsymbol{a}_i\| = 1, |b_i| \leq 1$ for all $i \in \mathbb{N}$. Show that for every $N \in \mathbb{N}$, there exists a ReLU neural network with N neurons and one layer such that

$$\|f - f_N\|_{L^2(B_1^d)} \leq \frac{3\|c\|_1}{\sqrt{N}}.$$

Hence, every infinite ReLU neural network can be approximated at a rate $O(N^{1/2})$ by finite ReLU neural networks of width N.

Exercise 8.12. Let $C > 0$ prove that every $f \in \Gamma_C$ is continuously differentiable.

Chapter 9

Interpolation

The learning problem associated to minimizing the empirical risk of (1.2.3) is based on minimizing an error that results from evaluating a neural network on a *finite* set of (training) points. In contrast, all previous approximation results focused on achieving uniformly small errors across the entire domain. Finding neural networks that achieve a small training error appears to be much simpler, since, instead of $\|f - \Phi_n\|_\infty \to 0$ for a sequence of neural networks Φ_n, it suffices to have $\Phi_n(\boldsymbol{x}_i) \to f(\boldsymbol{x}_i)$ for all $\boldsymbol{x}_i$ in the training set.

In this chapter, we study the extreme case of the aforementioned approximation problem. We analyze under which conditions it is possible to find a neural network that coincides with the target function f at all training points. This is referred to as *interpolation*. To make this notion more precise, we state the following definition.

Definition 9.1 (Interpolation)**.** Let d, $m \in \mathbb{N}$, and let $\Omega \subseteq \mathbb{R}^d$. We say that a set of functions $\mathcal{H} \subseteq \{h \colon \Omega \to \mathbb{R}\}$ **interpolates** m **points in** Ω, if for every $S = (\boldsymbol{x}_i, y_i)_{i=1}^m \subseteq \Omega \times \mathbb{R}$, such that $\boldsymbol{x}_i \neq \boldsymbol{x}_j$ for $i \neq j$, there exists a function $h \in \mathcal{H}$ such that $h(\boldsymbol{x}_i) = y_i$ for all $i = 1, \dots, m$.

Knowing the interpolation properties of an architecture represents extremely valuable information for two reasons:

- Consider an architecture that interpolates m points and let the number of training samples be bounded by m. Then (1.2.3) always has a solution.
- Consider again an architecture that interpolates m points and assume that the number of training samples is *less* than m. Then for every point $\tilde{\boldsymbol{x}}$ not in the training set and every $y \in \mathbb{R}$ there exists a minimizer h of (1.2.3) that satisfies $h(\tilde{\boldsymbol{x}}) = y$. As a consequence, without further restrictions (many of which we will discuss below), such an architecture cannot generalize to unseen data.

The existence of solutions to the interpolation problem does not follow trivially from the approximation results provided in the previous chapters (even though we will later see that there is a close connection). We also remark that the question of how many points neural networks with a given architecture can interpolate is closely related to the so-called VC dimension, which we will study in Chapter 14.

We start our analysis of the interpolation properties of neural networks by presenting a result similar to the universal approximation theorem but for interpolation in the following section. In the subsequent section, we then look at interpolation with desirable properties.

9.1 Universal interpolation

Under what conditions on the activation function and architecture can a set of neural networks interpolate $m \in \mathbb{N}$ points? According to Chapter 3, particularly Theorem 3.8, we know that shallow neural networks can approximate every continuous function with arbitrary accuracy, provided the neural network width is large enough. As the neural network's width and/or depth increases, the architectures become increasingly powerful, leading us to expect that at some point, they should be able to interpolate m points. However, this intuition may not be correct:

Example 9.2. Let $\mathcal{H} := \{f \in C^0([0,1]) \,|\, f(0) \in \mathbb{Q}\}$. Then $\mathcal{H}$ is dense in $C^0([0,1])$, but $\mathcal{H}$ does not even interpolate one point in $[0,1]$.

Moreover, Theorem 3.8 is an asymptotic result that only states that a given function can be approximated for sufficiently large neural network architectures, but it does not state how large the architecture needs to be.

Surprisingly, Theorem 3.8 can nonetheless be used to give a guarantee that a fixed-size architecture yields sets of neural networks that allow the interpolation of m points. This result is due to [176]; for a more detailed discussion of previous results see the bibliography section. Due to its similarity to the universal approximation theorem and the fact that it uses the same assumptions, we call the following theorem the "Universal Interpolation Theorem". For its statement recall the definition of the set of allowed activation functions $\mathcal{M}$ in (3.1.1) and the class $\mathcal{N}_d^1(\sigma, 1, n)$ of shallow neural networks of width n introduced in Definition 3.6.

Theorem 9.3 (Universal Interpolation Theorem)**.** *Let d, $n \in \mathbb{N}$ and let $\sigma \in \mathcal{M}$ not be a polynomial. Then $\mathcal{N}_d^1(\sigma, 1, n)$ interpolates $n+1$ points in $\mathbb{R}^d$.*

Proof. Fix $(\boldsymbol{x}_i)_{i=1}^{n+1} \subseteq \mathbb{R}^d$ arbitrary. We will show that for any $(y_i)_{i=1}^{n+1} \subseteq \mathbb{R}$ there exist weights and biases $(\boldsymbol{w}_j)_{j=1}^n \subseteq \mathbb{R}^d$, $(b_j)_{j=1}^n$, $(v_j)_{j=1}^n \subseteq \mathbb{R}$, $c \in \mathbb{R}$ such that

$$\Phi(\boldsymbol{x}_i) := \sum_{j=1}^{n} v_j \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i + b_j) + c = y_i \quad \text{for all} \quad i = 1, \dots, n+1. \tag{9.1.1}$$

Since $\Phi \in \mathcal{N}_d^1(\sigma, 1, n)$ this then concludes the proof.

Denote

$$\boldsymbol{A} := \begin{pmatrix} 1 & \sigma(\boldsymbol{w}_1^\top \boldsymbol{x}_1 + b_1) & \cdots & \sigma(\boldsymbol{w}_n^\top \boldsymbol{x}_1 + b_n) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \sigma(\boldsymbol{w}_1^\top \boldsymbol{x}_{n+1} + b_1) & \cdots & \sigma(\boldsymbol{w}_n^\top \boldsymbol{x}_{n+1} + b_n) \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}. \tag{9.1.2}$$

Then $\boldsymbol{A}$ being regular implies that for each $(y_i)_{i=1}^{n+1}$ exist c and $(v_j)_{j=1}^n$ such that (9.1.1) holds. Hence, it suffices to find $(\boldsymbol{w}_j)_{j=1}^n$ and $(b_j)_{j=1}^n$ such that $\boldsymbol{A}$ is regular.

To do so, we proceed by induction over $k = 0, \dots, n$, to show that there exist $(\boldsymbol{w}_j)_{j=1}^k$ and $(b_j)_{j=1}^k$ such that the first $k+1$ columns of $\boldsymbol{A}$ are linearly independent. The case $k = 0$ is trivial. Next let $0 < k < n$ and assume that the first k columns of $\boldsymbol{A}$ are linearly independent. We wish to find $\boldsymbol{w}_k$, b_k such that the first $k+1$ columns are linearly independent. Suppose such $\boldsymbol{w}_k$, b_k do not exist and denote by $Y_k \subseteq \mathbb{R}^{n+1}$ the space spanned by the first k columns of $\boldsymbol{A}$. Then for all $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$ the vector $(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b))_{i=1}^{n+1} \in \mathbb{R}^{n+1}$ must belong to Y_k. Fix $\boldsymbol{y} = (y_i)_{i=1}^{n+1} \in \mathbb{R}^{n+1} \backslash Y_k$. Then

$$\begin{aligned}\inf_{\tilde{\Phi} \in \mathcal{N}_d^1(\sigma,1)} \|(\tilde{\Phi}(\boldsymbol{x}_i))_{i=1}^{n+1} - \boldsymbol{y}\|_2^2 &= \inf_{N, \boldsymbol{w}_j, b_j, v_j, c} \sum_{i=1}^{n+1} \Big(\sum_{j=1}^{N} v_j \sigma(\boldsymbol{w}_j^\top \boldsymbol{x}_i + b_j) + c - y_i \Big)^2 \\ &\geq \inf_{\tilde{\boldsymbol{y}} \in Y_k} \|\tilde{\boldsymbol{y}} - \boldsymbol{y}\|_2^2 > 0.\end{aligned}$$

Since we can find a continuous function $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(\boldsymbol{x}_i) = y_i$ for all $i = 1, \dots, n+1$, this contradicts Theorem 3.8. □

9.2 Optimal interpolation and reconstruction

Consider a bounded domain $\Omega \subseteq \mathbb{R}^d$, a function $f : \Omega \to \mathbb{R}$, distinct points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m \in \Omega$, and corresponding function values $y_i := f(\boldsymbol{x}_i)$. Our objective is to approximate f based solely on the data pairs $(\boldsymbol{x}_i, y_i)$, $i = 1, \dots, m$. In this section, we will show that, under certain assumptions on f, ReLU neural networks can express an "optimal" reconstruction which also turns out to be an interpolant of the data.

9.2.1 Motivation

In the previous section, we observed that neural networks with $m - 1 \in \mathbb{N}$ hidden neurons can interpolate m points for every reasonable activation function. However, not all interpolants are equally suitable for a given application. For instance, consider Figure 9.1 for a comparison between polynomial and piecewise affine interpolation on the unit interval.

The two interpolants exhibit rather different behaviors. In general, there is no way of determining which constitutes a better approximation to f. In particular, given our limited information about f, we cannot accurately reconstruct any additional features that may exist between interpolation points $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$. In accordance with Occam's razor, it thus seems reasonable to assume that f does not exhibit extreme oscillations or behave erratically between interpolation points. As such, the piecewise interpolant appears preferable in this scenario. One way to formalize the assumption that f does not "exhibit extreme oscillations" is to *assume* that the Lipschitz constant

$$\mathrm{Lip}(f) := \sup_{\boldsymbol{x} \neq \boldsymbol{y}} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \tag{9.2.1}$$

of f is bounded by a fixed value $M \in \mathbb{R}$. Here $\|\cdot\|$ denotes an arbitrary fixed norm on $\mathbb{R}^d$.

How should we choose M? For every function $f : \Omega \to \mathbb{R}$ satisfying

$$f(\boldsymbol{x}_i) = y_i \quad \text{for all} \quad i = 1, \dots, m, \tag{9.2.2}$$

we have

$$\mathrm{Lip}(f) = \sup_{\boldsymbol{x} \neq \boldsymbol{y} \in \Omega} \frac{|f(\boldsymbol{x}) - f(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \geq \sup_{i \neq j} \frac{|y_i - y_j|}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|} =: \tilde{M}. \tag{9.2.3}$$

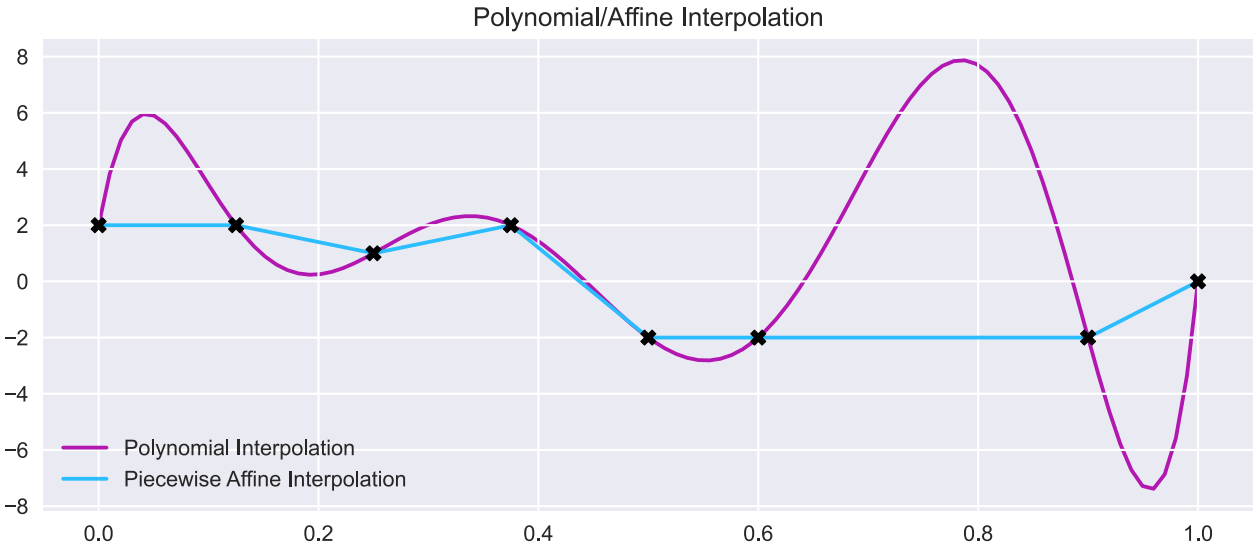


Figure 9.1: Interpolation of eight points by a polynomial of degree seven and by a piecewise affine spline. The polynomial interpolation has a significantly larger derivative or Lipschitz constant than the piecewise affine interpolator.

Because of this, we fix M as a real number greater than or equal to $\tilde{M}$ for the remainder of our analysis.

9.2.2 Optimal reconstruction for Lipschitz continuous functions

The above considerations raise the following question: *Given only the information that the function has Lipschitz constant at most M, what is the best reconstruction of f based on the data?* We consider here the "best reconstruction" to be a function that minimizes the L^∞-error in the worst case. Specifically, with

$$\mathrm{Lip}_M(\Omega) := \{f : \Omega \to \mathbb{R} \,|\, \mathrm{Lip}(f) \leq M\}, \tag{9.2.4}$$

denoting the set of all functions with Lipschitz constant at most M, we want to solve the following problem:

Problem 9.4. We wish to find an element

$$\Phi \in \mathrm{argmin}_{h:\Omega\to\mathbb{R}} \sup_{\substack{f\in\mathrm{Lip}_M(\Omega)\\ f \text{ satisfies (9.2.2)}}} \sup_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - h(\boldsymbol{x})|. \tag{9.2.5}$$

The next theorem shows that a function Φ as in (9.2.5) indeed exists. This Φ not only allows for an explicit formula, it also belongs to $\mathrm{Lip}_M(\Omega)$ and additionally interpolates the data. Hence, it is not just an optimal reconstruction, it is also an optimal interpolant. This theorem goes back to [13], which, in turn, is based on [219].

Theorem 9.5. *Let m, $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$, $f : \Omega \to \mathbb{R}$, and let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m \in \Omega$, $y_1, \dots, y_m \in \mathbb{R}$ satisfy* (9.2.2) *and* (9.2.3) *with $\tilde{M} > 0$. Further, let $M \geq \tilde{M}$.*

Then, Problem 9.4 has at least one solution given by

$$\Phi(\boldsymbol{x}) := \frac{1}{2}(f_{\text{upper}}(\boldsymbol{x}) + f_{\text{lower}}(\boldsymbol{x})) \qquad \text{for } \boldsymbol{x} \in \Omega, \tag{9.2.6}$$

where

$$f_{\text{upper}}(\boldsymbol{x}) := \min_{k=1,\dots,m} (y_k + M\|\boldsymbol{x} - \boldsymbol{x}_k\|)$$
$$f_{\text{lower}}(\boldsymbol{x}) := \max_{k=1,\dots,m} (y_k - M\|\boldsymbol{x} - \boldsymbol{x}_k\|).$$

Moreover, $\Phi \in \text{Lip}_M(\Omega)$ and Φ interpolates the data (i.e. satisfies (9.2.2)*).*

Proof. First we claim that for all h_1, $h_2 \in \text{Lip}_M(\Omega)$ holds $\max\{h_1, h_2\} \in \text{Lip}_M(\Omega)$ as well as $\min\{h_1, h_2\} \in \text{Lip}_M(\Omega)$. Since $\min\{h_1, h_2\} = -\max\{-h_1, -h_2\}$, it suffices to show the claim for the maximum. We need to check that

$$\frac{|\max\{h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\} - \max\{h_1(\boldsymbol{y}), h_2(\boldsymbol{y})\}|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \leq M \tag{9.2.7}$$

for all $\boldsymbol{x} \neq \boldsymbol{y} \in \Omega$. Fix $\boldsymbol{x} \neq \boldsymbol{y}$. Without loss of generality we assume that

$$\max\{h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\} \geq \max\{h_1(\boldsymbol{y}), h_2(\boldsymbol{y})\} \quad \text{and} \quad \max\{h_1(\boldsymbol{x}), h_2(\boldsymbol{x})\} = h_1(\boldsymbol{x}).$$

If $\max\{h_1(\boldsymbol{y}), h_2(\boldsymbol{y})\} = h_1(\boldsymbol{y})$ then the numerator in (9.2.7) equals $h_1(\boldsymbol{x}) - h_1(\boldsymbol{y})$ which is bounded by $M\|\boldsymbol{x} - \boldsymbol{y}\|$. If $\max\{h_1(\boldsymbol{y}), h_2(\boldsymbol{y})\} = h_2(\boldsymbol{y})$, then the numerator equals $h_1(\boldsymbol{x}) - h_2(\boldsymbol{y})$ which is bounded by $h_1(\boldsymbol{x}) - h_1(\boldsymbol{y}) \leq M\|\boldsymbol{x} - \boldsymbol{y}\|$. In either case (9.2.7) holds.

Clearly, $\boldsymbol{x} \mapsto y_k - M\|\boldsymbol{x} - \boldsymbol{x}_k\| \in \text{Lip}_M(\Omega)$ for each $k = 1, \dots, m$ and thus f_{upper}, $f_{\text{lower}} \in \text{Lip}_M(\Omega)$ as well as $\Phi \in \text{Lip}_M(\Omega)$.

Next we claim that for all $f \in \text{Lip}_M(\Omega)$ satisfying (9.2.2) holds

$$f_{\text{lower}}(\boldsymbol{x}) \leq f(\boldsymbol{x}) \leq f_{\text{upper}}(\boldsymbol{x}) \quad \text{for all} \quad \boldsymbol{x} \in \Omega. \tag{9.2.8}$$

This is true since for every $k \in \{1, \dots, m\}$ and $\boldsymbol{x} \in \Omega$

$$|y_k - f(\boldsymbol{x})| = |f(\boldsymbol{x}_k) - f(\boldsymbol{x})| \leq M\|\boldsymbol{x} - \boldsymbol{x}_k\|$$

so that for all $\boldsymbol{x} \in \Omega$

$$f(\boldsymbol{x}) \leq \min_{k=1,\dots,m} (y_k + M\|\boldsymbol{x} - \boldsymbol{x}_k\|), \qquad f(\boldsymbol{x}) \geq \max_{k=1,\dots,m} (y_k - M\|\boldsymbol{x} - \boldsymbol{x}_k\|).$$

Since f_{upper}, $f_{\text{lower}} \in \text{Lip}_M(\Omega)$ satisfy (9.2.2), we conclude that for every $h : \Omega \to \mathbb{R}$ holds

$$\begin{aligned} \sup_{\substack{f \in \text{Lip}_M(\Omega) \\ f \text{ satisfies (9.2.2)}}} \sup_{\boldsymbol{x} \in \Omega} |f(\boldsymbol{x}) - h(\boldsymbol{x})| &\geq \sup_{\boldsymbol{x} \in \Omega} \max\{|f_{\text{lower}}(\boldsymbol{x}) - h(\boldsymbol{x})|, |f_{\text{upper}}(\boldsymbol{x}) - h(\boldsymbol{x})|\} \\ &\geq \sup_{\boldsymbol{x} \in \Omega} \frac{|f_{\text{lower}}(\boldsymbol{x}) - f_{\text{upper}}(\boldsymbol{x})|}{2}. \end{aligned} \tag{9.2.9}$$

Moreover, using (9.2.8),

$$\sup_{\substack{f\in \mathrm{Lip}_M(\Omega)\\ f \text{ satisfies (9.2.2)}}} \sup_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - \Phi(\boldsymbol{x})| \leq \sup_{\boldsymbol{x}\in\Omega} \max\{|f_{\mathrm{lower}}(\boldsymbol{x}) - \Phi(\boldsymbol{x})|, |f_{\mathrm{upper}}(\boldsymbol{x}) - \Phi(\boldsymbol{x})|\}$$

$$= \sup_{\boldsymbol{x}\in\Omega} \frac{|f_{\mathrm{lower}}(\boldsymbol{x}) - f_{\mathrm{upper}}(\boldsymbol{x})|}{2}. \tag{9.2.10}$$

Finally, (9.2.9) and (9.2.10) imply that Φ is a solution of Problem 9.4. □

Figure 9.2 depicts f_{upper}, f_{lower}, and Φ for the interpolation problem shown in Figure 9.1, while Figure 9.3 provides a two-dimensional example.

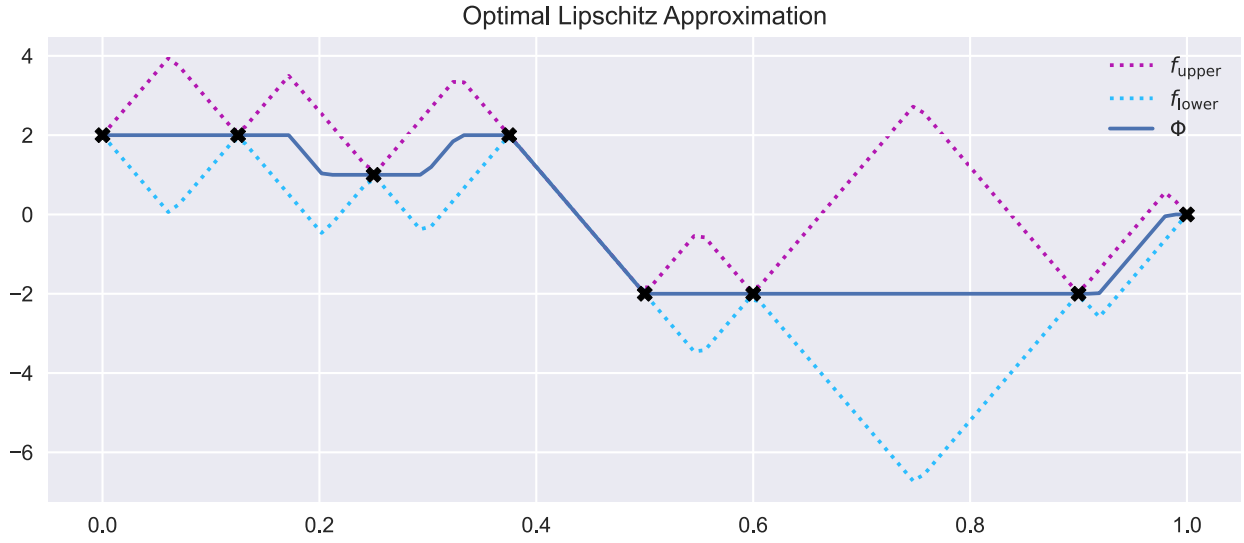


Figure 9.2: Interpolation of the points from Figure 9.1 with the optimal Lipschitz interpolant.

9.2.3 Optimal ReLU reconstructions

So far everything was valid with an arbitrary norm on $\mathbb{R}^d$. For the next theorem, we will restrict ourselves to the 1-norm $\|\boldsymbol{x}\|_1 = \sum_{j=1}^d |x_j|$. Using the explicit formula of Theorem 9.5, we will now show the remarkable result that ReLU neural networks can exactly express an optimal reconstruction (in the sense of Problem 9.4) with a neural network whose size scales linearly in the product of the dimension d and the number of data points m. Additionally, the proof is constructive, thus allowing in principle for an explicit construction on the neural network without the need for training.

Theorem 9.6 (Optimal Lipschitz Reconstruction)**.** *Let m, $d \in \mathbb{N}$, $\Omega \subseteq \mathbb{R}^d$, $f : \Omega \to \mathbb{R}$, and let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \Omega$, $y_1, \ldots, y_m \in \mathbb{R}$ satisfy* (9.2.2) *and* (9.2.3) *with $\tilde{M} > 0$. Further, let $M \geq \tilde{M}$ and let $\|\cdot\| = \|\cdot\|_1$ in* (9.2.3) *and* (9.2.4)*.*

Then, there exists a ReLU neural network $\Phi \in \mathrm{Lip}_M(\Omega)$ *that interpolates the data (i.e. satisfies* (9.2.2)*) and satisfies*

$$\Phi \in \operatorname{argmin}_{h:\Omega\to\mathbb{R}} \sup_{\substack{f\in\mathrm{Lip}_M(\Omega) \\ f \text{ satisfies } (9.2.2)}} \sup_{\boldsymbol{x}\in\Omega} |f(\boldsymbol{x}) - h(\boldsymbol{x})|.$$

Moreover, $\mathrm{depth}(\Phi) = O(\log(m))$*,* $\mathrm{width}(\Phi) = O(dm)$ *and all weights of* Φ *are bounded in absolute value by* $\max\{M, \|\boldsymbol{y}\|_\infty\}$*.*

Proof. To prove the result, we simply need to show that the function in (9.2.6) can be expressed as a ReLU neural network with the size bounds described in the theorem. First we notice, that there is a simple ReLU neural network that implements the 1-norm. It holds for all $\boldsymbol{x} \in \mathbb{R}^d$ that

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{d} \left(\sigma(x_i) + \sigma(-x_i)\right).$$

Thus, there exists a ReLU neural network $\Phi^{\|\cdot\|_1}$ such that for all $\boldsymbol{x} \in \mathbb{R}^d$

$$\mathrm{width}(\Phi^{\|\cdot\|_1}) = 2d, \qquad \mathrm{depth}(\Phi^{\|\cdot\|_1}) = 1, \qquad \Phi^{\|\cdot\|_1}(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$$

As a result, there exist ReLU neural networks $\Phi_k : \mathbb{R}^d \to \mathbb{R}$, $k = 1, \dots, m$, such that

$$\mathrm{width}(\Phi_k) = 2d, \qquad \mathrm{depth}(\Phi_k) = 1, \qquad \Phi_k(\boldsymbol{x}) = y_k + M\|\boldsymbol{x} - \boldsymbol{x}_k\|_1$$

for all $\boldsymbol{x} \in \mathbb{R}^d$. Using the parallelization of neural networks introduced in Section 5.1.3, there exists a ReLU neural network $\Phi_{\text{all}} := (\Phi_1, \dots, \Phi_m) \colon \mathbb{R}^d \to \mathbb{R}^m$ such that

$$\mathrm{width}(\Phi_{\text{all}}) = 4md, \qquad \mathrm{depth}(\Phi_{\text{all}}) = 1$$

and

$$\Phi_{\text{all}}(\boldsymbol{x}) = (y_k + M\|\boldsymbol{x} - \boldsymbol{x}_k\|_1)_{k=1}^m \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^d.$$

Using Lemma 5.11, we can now find a ReLU neural network Φ_{upper} such that $\Phi_{\text{upper}} = f_{\text{upper}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$, $\mathrm{width}(\Phi_{\text{upper}}) \leq \max\{16m, 4md\}$, and $\mathrm{depth}(\Phi_{\text{upper}}) \leq 1 + \log(m)$.

Essentially the same construction yields a ReLU neural network Φ_{lower} with the respective properties. Lemma 5.4 then completes the proof. □

Bibliography and further reading

The universal interpolation theorem stated in this chapter is due to [176, Theorem 5.1]. There exist several earlier interpolation results, which were shown under stronger assumptions: In [200], the interpolation property is already linked with a rank condition on the matrix (9.1.2). However, no general conditions on the activation functions were formulated. In [105], the interpolation theorem is established under the assumption that the activation function σ is continuous and nondecreasing,

$\lim_{x\to-\infty}\sigma(x) = 0$, and $\lim_{x\to\infty}\sigma(x) = 1$. This result was improved in [97], which dropped the nondecreasing assumption on σ.

The main idea of the optimal Lipschitz interpolation theorem in Section 9.2 is due to [13]. A neural network construction of Lipschitz interpolants, which however is not the optimal interpolant in the sense of Problem 9.4, is given in [108, Theorem 2.27].

Exercises

Exercise 9.7. Under the assumptions of Theorem 9.5, we define for $x \in \Omega$ the set of nearest neighbors by

$$I_x := \operatorname{argmin}_{i=1,\dots,m} \|x_i - x\|.$$

The one-nearest-neighbor classifier $f_{1\mathrm{NN}}$ is defined by

$$f_{1\mathrm{NN}}(x) = \frac{1}{2}(\min_{i \in I_x} y_i + \max_{i \in I_x} y_i).$$

Let Φ_M be the function in (9.2.6). Show that for all $x \in \Omega$

$$\Phi_M(x) \to f_{1\mathrm{NN}}(x) \qquad \text{as } M \to \infty.$$

Exercise 9.8. Extend Theorem 9.6 to the $\|\cdot\|_\infty$-norm. *Hint:* The resulting neural network will need to be deeper than the one of Theorem 9.6.

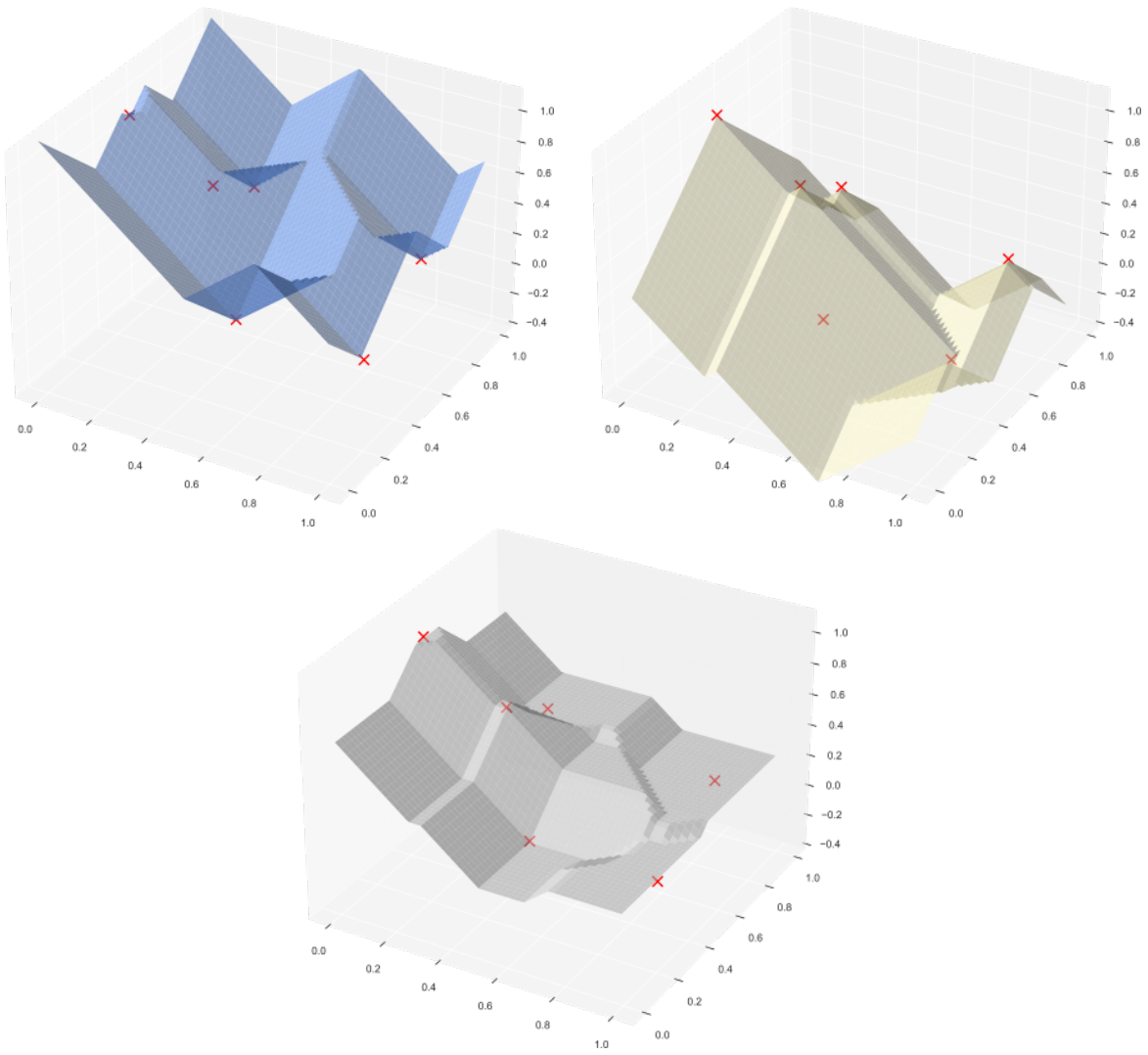


Figure 9.3: Two-dimensional example of the interpolation method of (9.2.6). From top left to bottom we see f_{upper}, f_{lower}, and Φ. The interpolation points $(\boldsymbol{x}_i, y_i)_{i=1}^{6}$ are marked with red crosses.

Chapter 10

Training of neural networks

Up to this point, we have discussed the representation and approximation of certain function classes using neural networks. The second pillar of deep learning concerns the question of how to fit a neural network to given data, i.e., having fixed an architecture, how to find suitable weights and biases. This task amounts to minimizing a so-called **objective function** such as the empirical risk $\hat{\mathcal{R}}_S$ in (1.2.3). Throughout this chapter we denote the objective function by

$$f : \mathbb{R}^n \to \mathbb{R},$$

and interpret it as a function of all neural network weights and biases collected in a vector in $\mathbb{R}^n$. The goal is to (approximately) determine a **minimizer**, i.e., some $\boldsymbol{w}_* \in \mathbb{R}^n$ satisfying

$$f(\boldsymbol{w}_*) \leq f(\boldsymbol{w}) \qquad \text{for all } \boldsymbol{w} \in \mathbb{R}^n.$$

Standard approaches include, in particular, variants of (stochastic) gradient descent. These are the topic of this chapter, in which we present basic ideas and results in convex optimization using gradient-based methods.

10.1 Gradient descent

The general idea of gradient descent is to start with some $\boldsymbol{w}_0 \in \mathbb{R}^n$, and then apply sequential updates by moving in the direction of *steepest descent* of the objective function. Assume for the moment that $f \in C^2(\mathbb{R}^n)$, and denote the kth iterate by $\boldsymbol{w}_k$. Then

$$f(\boldsymbol{w}_k + \boldsymbol{v}) = f(\boldsymbol{w}_k) + \boldsymbol{v}^\top \nabla f(\boldsymbol{w}_k) + O(\|\boldsymbol{v}\|^2) \qquad \text{for } \|\boldsymbol{v}\|^2 \to 0. \tag{10.1.1}$$

This shows that the change in f around $\boldsymbol{w}_k$ is locally described by the gradient $\nabla f(\boldsymbol{w}_k)$. For small $\boldsymbol{v}$ the contribution of the second order term is negligible, and the direction $\boldsymbol{v}$ along which the decrease of the risk is maximized equals the negative gradient $-\nabla f(\boldsymbol{w}_k)$. Thus, $-\nabla f(\boldsymbol{w}_k)$ is also called the direction of steepest descent. This leads to an update of the form

$$\boldsymbol{w}_{k+1} := \boldsymbol{w}_k - h_k \nabla f(\boldsymbol{w}_k), \tag{10.1.2}$$

where $h_k > 0$ is referred to as the **step size** or **learning rate**. We refer to this iterative algorithm as **gradient descent**.

In practice tuning the learning rate can be a subtle issue as it should strike a balance between the following dissenting requirements:

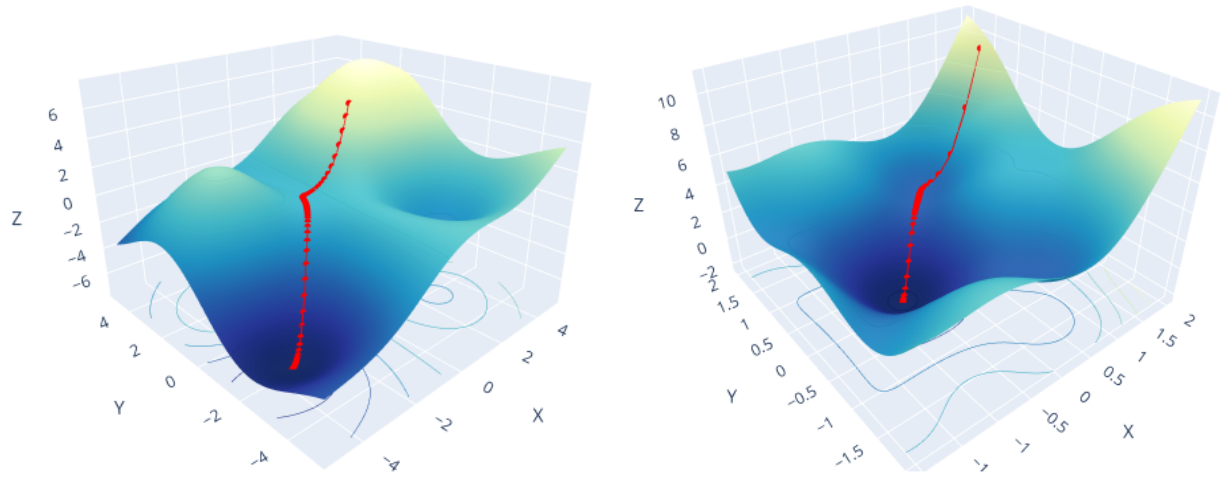


Figure 10.1: Two examples of gradient descent as defined in (10.1.2). The red points represent the $\boldsymbol{w}_k$.

(i) h_k needs to be sufficiently small so that with $\boldsymbol{v} = -h_k \nabla f(\boldsymbol{w}_k)$, the second-order term in (10.1.1) is not dominating. This ensures that the update (10.1.2) decreases the objective function.

(ii) h_k should be large enough to ensure significant decrease of the objective function, which facilitates faster convergence of the algorithm.

A learning rate that is too high might overshoot the minimum, while a rate that is too low results in slow convergence. Common strategies include, in particular, constant learning rates ($h_k = h$ for all $k \in \mathbb{N}_0$), learning rate schedules such as decaying learning rates ($h_k \searrow 0$ as $k \to \infty$), and adaptive methods. For adaptive methods the algorithm dynamically adjust h_k based on the values of $f(\boldsymbol{w}_j)$ or $\nabla f(\boldsymbol{w}_j)$ for $j \leq k$.

Remark 10.1. It is instructive to interpret (10.1.2) as an Euler discretization of the "gradient flow"

$$\boldsymbol{w}(0) = \boldsymbol{w}_0, \qquad \boldsymbol{w}'(t) = -\nabla f(\boldsymbol{w}(t)) \quad \text{for } t \in [0, \infty). \tag{10.1.3}$$

This ODE describes the movement of a particle $\boldsymbol{w}(t)$, whose velocity at time $t \geq 0$ equals $-\nabla f(\boldsymbol{w}(t))$—the vector of steepest descent. Note that

$$\frac{\mathrm{d} f(\boldsymbol{w}(t))}{\mathrm{d}t} = \big\langle \nabla f(\boldsymbol{w}(t)), \boldsymbol{w}'(t) \big\rangle = -\|\nabla f(\boldsymbol{w}(t))\|^2,$$

and thus the dynamics (10.1.3) necessarily decreases the value of the objective function along its path as long as $\nabla f(\boldsymbol{w}(t)) \neq 0$.

Throughout the rest of Section 10.1 we assume that $\boldsymbol{w}_0 \in \mathbb{R}^n$ is arbitrary, and the sequence $(\boldsymbol{w}_k)_{k \in \mathbb{N}_0}$ is generated by (10.1.2). We will analyze the convergence of this algorithm under suitable assumptions on f and the h_k. The proofs primarily follow the arguments in [159, Chapter 2]. We also refer to that book for a much more detailed discussion of gradient descent, and further reading on convex optimization.

10.1.1 L-smoothness

A key assumption to analyze convergence of (10.1.2) is Lipschitz continuity of ∇f.

Definition 10.2. Let $n \in \mathbb{N}$, and $L > 0$. The function $f : \mathbb{R}^n \to \mathbb{R}$ is called **L-smooth** if $f \in C^1(\mathbb{R}^n)$ and

$$\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{v})\| \leq L\|\boldsymbol{w} - \boldsymbol{v}\| \qquad \text{for all } \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n.$$

For fixed $\boldsymbol{w}$, L-smoothness implies the linear growth bound

$$\|\nabla f(\boldsymbol{w} + \boldsymbol{v})\| \leq \|\nabla f(\boldsymbol{w})\| + L\|\boldsymbol{v}\|$$

for ∇f. Integrating the gradient along lines in $\mathbb{R}^n$ then shows that f is bounded from above by a quadratic function touching the graph of f at $\boldsymbol{w}$, as stated in the next lemma; also see Figure 10.2.

Lemma 10.3. *Let $n \in \mathbb{N}$ and $L > 0$. Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth. Then*

$$f(\boldsymbol{v}) \leq f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle + \frac{L}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 \qquad \textit{for all } \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n. \tag{10.1.4}$$

Proof. We have for all $\boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n$

$$\begin{aligned} f(\boldsymbol{v}) &= f(\boldsymbol{w}) + \int_0^1 \langle \nabla f(\boldsymbol{w} + t(\boldsymbol{v} - \boldsymbol{w})), \boldsymbol{v} - \boldsymbol{w} \rangle \, \mathrm{d}t \\ &= f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle + \int_0^1 \langle \nabla f(\boldsymbol{w} + t(\boldsymbol{v} - \boldsymbol{w})) - \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle \, \mathrm{d}t. \end{aligned}$$

Thus

$$f(\boldsymbol{v}) - f(\boldsymbol{w}) - \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle \leq \int_0^1 L\|t(\boldsymbol{v} - \boldsymbol{w})\| \|\boldsymbol{v} - \boldsymbol{w}\| \, \mathrm{d}t = \frac{L}{2}\|\boldsymbol{v} - \boldsymbol{w}\|^2,$$

which shows (10.1.4). □

Remark 10.4. The argument in the proof of Lemma 10.3 also gives the lower bound

$$f(\boldsymbol{v}) \geq f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle - \frac{L}{2}\|\boldsymbol{w} - \boldsymbol{v}\|^2 \qquad \text{for all } \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n. \tag{10.1.5}$$

The previous lemma allows us to show a decay property for the gradient descent iterates. Specifically, the values of f necessarily decrease in each iteration as long as the step size h_k is small enough, and $\nabla f(\boldsymbol{w}_k) \neq 0$.

Lemma 10.5. *Let $n \in \mathbb{N}$ and $L > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth. Further, let $(h_k)_{k=1}^{\infty}$ be positive numbers and let $(\boldsymbol{w}_k)_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be defined by* (10.1.2).

Then, for all $k \in \mathbb{N}$

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) - \Big(h_k - \frac{Lh_k^2}{2}\Big)\|\nabla f(\boldsymbol{w}_k)\|^2. \tag{10.1.6}$$

Proof. Lemma 10.3 with $\boldsymbol{v} = \boldsymbol{w}_{k+1}$ and $\boldsymbol{w} = \boldsymbol{w}_k$ gives

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) + \langle \nabla f(\boldsymbol{w}_k), -h_k \nabla f(\boldsymbol{w}_k)\rangle + \frac{L}{2}\|h_k \nabla f(\boldsymbol{w}_k)\|^2,$$

which corresponds to (10.1.6). □

Remark 10.6. The right-hand side in (10.1.6) is minimized for step size $h_k = 1/L$, in which case (10.1.6) reads

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2.$$

Next, let us discuss the behavior of the gradients for constant step sizes.

Proposition 10.7. *Let $n \in \mathbb{N}$ and $L > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth. Further, let $h_k = h \in (0, 2/L)$ for all $k \in \mathbb{N}$, and $(\boldsymbol{w}_k)_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be defined by* (10.1.2).

Then, for all $k \in \mathbb{N}$

$$\frac{1}{k+1}\sum_{j=0}^{k} \|\nabla f(\boldsymbol{w}_j)\|^2 \leq \frac{1}{k+1}\frac{2}{2h - Lh^2}(f(\boldsymbol{w}_0) - f(\boldsymbol{w}_{k+1})). \tag{10.1.7}$$

Proof. Set $c := h - (Lh^2)/2 = (2h - Lh^2)/2 > 0$. By (10.1.6) for $j \geq 0$

$$f(\boldsymbol{w}_j) - f(\boldsymbol{w}_{j+1}) \geq c\|\nabla f(\boldsymbol{w}_j)\|^2.$$

Hence

$$\sum_{j=0}^{k} \|\nabla f(\boldsymbol{w}_j)\|^2 \leq \frac{1}{c}\sum_{j=0}^{k} f(\boldsymbol{w}_j) - f(\boldsymbol{w}_{j+1}) = \frac{1}{c}\left(f(\boldsymbol{w}_0) - f(\boldsymbol{w}_{k+1})\right).$$

Dividing by $k+1$ concludes the proof. □

Suppose that f is bounded from below, i.e. $\inf_{\boldsymbol{w}\in\mathbb{R}^n} f(\boldsymbol{w}) > -\infty$. In this case, the right-hand side in (10.1.7) behaves like $O(k^{-1})$ as $k \to \infty$, and (10.1.7) implies

$$\min_{j=1,\dots,k} \|\nabla f(\boldsymbol{w}_j)\| = O(k^{-1/2}).$$

Thus, lower boundedness of the objective function together with L-smoothness already suffice to obtain some form of convergence of the gradients to 0. We emphasize that this does *not imply* convergence of $\boldsymbol{w}_k$ towards some $\boldsymbol{w}_*$ with $\nabla f(\boldsymbol{w}_*) = 0$ as the example $f(w) = \arctan(w)$, $w \in \mathbb{R}$, shows.

10.1.2 Convexity

While L-smoothness entails some interesting properties of gradient descent, it does not have any direct implications on the existence or uniqueness of minimizers. To show convergence of $f(\boldsymbol{w}_k)$ towards $\min_{\boldsymbol{w}} f(\boldsymbol{w})$ for $k \to \infty$ (assuming this minimum exists), we will assume that f is a convex function.

Definition 10.8. Let $n \in \mathbb{N}$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is called **convex** if and only if

$$f(\lambda \boldsymbol{w} + (1-\lambda)\boldsymbol{v}) \leq \lambda f(\boldsymbol{w}) + (1-\lambda) f(\boldsymbol{v}), \tag{10.1.8}$$

for all $\boldsymbol{w}$, $\boldsymbol{v} \in \mathbb{R}^n$, $\lambda \in (0,1)$.

Let $n \in \mathbb{N}$. If $f \in C^1(\mathbb{R}^n)$, then f is convex if and only if

$$f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle \leq f(\boldsymbol{v}) \qquad \text{for all } \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n, \tag{10.1.9}$$

as shown in Exercise 10.27. Thus, $f \in C^1(\mathbb{R}^n)$ is convex if and only if the graph of f lies above each of its tangents, see Figure 10.2.

For convex f, a minimizer neither needs to exist (e.g., $f(w) = w$ for $w \in \mathbb{R}$) nor be unique (e.g., $f(\boldsymbol{w}) = 0$ for $\boldsymbol{w} \in \mathbb{R}^n$). However, if $\boldsymbol{w}_*$ and $\boldsymbol{v}_*$ are two minimizers, then every convex combination $\lambda \boldsymbol{w}_* + (1-\lambda)\boldsymbol{v}_*$, $\lambda \in [0,1]$, is also a minimizer due to (10.1.8). Thus, the set of all minimizers is convex. In particular, a convex objective function has either zero, one, or infinitely many minimizers. Moreover, if $f \in C^1(\mathbb{R}^n)$ then $\nabla f(\boldsymbol{w}) = 0$ implies

$$f(\boldsymbol{w}) = f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle \leq f(\boldsymbol{v}) \qquad \text{for all } \boldsymbol{v} \in \mathbb{R}^n.$$

Thus, $\boldsymbol{w}$ is a minimizer of f if and only if $\nabla f(\boldsymbol{w}) = 0$.

By Lemma 10.5, smallness of the step sizes and L-smoothness suffice to show a decay property for the objective function f. Under the additional assumption of convexity, we also get a decay property for the distance of $\boldsymbol{w}_k$ to *any* minimizer $\boldsymbol{w}_*$.

Lemma 10.9. *Let $n \in \mathbb{N}$ and $L > 0$. Let $f \colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth and convex. Further, let $h_k \in (0, 2/L)$ for all $k \in \mathbb{N}_0$, and $(\boldsymbol{w}_k)_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be defined by (10.1.2). Suppose that $\boldsymbol{w}_*$ is a minimizer of f.*

Then, for all $k \in \mathbb{N}_0$

$$\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_*\|^2 \leq \|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 - h_k \cdot \Big(\frac{2}{L} - h_k\Big) \|\nabla f(\boldsymbol{w}_k)\|^2.$$

To prove the lemma, we will require the following inequality [159, Theorem 2.1.5].

Lemma 10.10. *Let $n \in \mathbb{N}$ and $L > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth and convex.*
Then,

$$\frac{1}{L}\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{v})\|^2 \leq \langle \nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{v}), \boldsymbol{w} - \boldsymbol{v}\rangle \qquad \text{for all } \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n.$$

Proof. Fix $\boldsymbol{w} \in \mathbb{R}^n$ and set $\Psi(\boldsymbol{u}) := f(\boldsymbol{u}) - \langle \nabla f(\boldsymbol{w}), \boldsymbol{u}\rangle$ for all $\boldsymbol{u} \in \mathbb{R}^n$. Then $\nabla\Psi(\boldsymbol{u}) = \nabla f(\boldsymbol{u}) - \nabla f(\boldsymbol{w})$ and thus Ψ is L-smooth. Moreover, convexity of f, specifically (10.1.9), yields $\Psi(\boldsymbol{u}) \geq f(\boldsymbol{w}) - \langle \nabla f(\boldsymbol{w}), \boldsymbol{w}\rangle = \Psi(\boldsymbol{w})$ for all $\boldsymbol{u} \in \mathbb{R}^n$, and thus $\boldsymbol{w}$ is a minimizer of Ψ. Using (10.1.4) on Ψ we get for every $\boldsymbol{v} \in \mathbb{R}^n$

$$\begin{aligned}
\Psi(\boldsymbol{w}) &= \min_{\boldsymbol{u}\in\mathbb{R}^n} \Psi(\boldsymbol{u}) \leq \min_{\boldsymbol{u}\in\mathbb{R}^n} \left(\Psi(\boldsymbol{v}) + \langle \nabla\Psi(\boldsymbol{v}), \boldsymbol{u} - \boldsymbol{v}\rangle + \frac{L}{2}\|\boldsymbol{u} - \boldsymbol{v}\|^2\right) \\
&= \min_{t\geq 0} \Psi(\boldsymbol{v}) - t\|\nabla\Psi(\boldsymbol{v})\|^2 + t^2\frac{L}{2}\|\nabla\Psi(\boldsymbol{v})\|^2 \\
&= \Psi(\boldsymbol{v}) - \frac{1}{2L}\|\nabla\Psi(\boldsymbol{v})\|^2
\end{aligned}$$

since the minimum of $t \mapsto t^2 L/2 - t$ is attained at $t = L^{-1}$. This implies

$$f(\boldsymbol{w}) - f(\boldsymbol{v}) + \frac{1}{2L}\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{v})\|^2 \leq \langle \nabla f(\boldsymbol{w}), \boldsymbol{w} - \boldsymbol{v}\rangle .$$

Adding the same inequality with the roles of $\boldsymbol{w}$ and $\boldsymbol{v}$ switched gives the result. □

of Lemma 10.9. It holds

$$\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_*\|^2 = \|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 - 2h_k \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{w}_k - \boldsymbol{w}_*\rangle + h_k^2\|\nabla f(\boldsymbol{w}_k)\|^2.$$

Since $\nabla f(\boldsymbol{w}_*) = 0$, Lemma 10.10 gives

$$-\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{w}_k - \boldsymbol{w}_*\rangle \leq -\frac{1}{L}\|\nabla f(\boldsymbol{w}_k)\|^2$$

which implies the claim. □

These preparations allow us to show that for constant step size $h < 2/L$, we obtain convergence of $f(\boldsymbol{w}_k)$ towards $f(\boldsymbol{w}_*)$ with rate $O(k^{-1})$, as stated in the next theorem which corresponds to [159, Theorem 2.1.14].

Theorem 10.11. *Let $n \in \mathbb{N}$ and $L > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth and convex. Further, let $h_k = h \in (0, 2/L)$ for all $k \in \mathbb{N}_0$, and let $(\boldsymbol{w}_k)_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be defined by* (10.1.2)*. Suppose that $\boldsymbol{w}_*$ is a minimizer of f.*
Then, $f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*) = O(k^{-1})$ for $k \to \infty$, and for the specific choice $h = 1/L$

$$f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*) \leq \frac{2L}{4+k}\|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2 \qquad \text{for all } k \in \mathbb{N}_0. \tag{10.1.10}$$

Proof. The case $\boldsymbol{w}_0 = \boldsymbol{w}_*$ is trivial and throughout we assume $\boldsymbol{w}_0 \neq \boldsymbol{w}_*$.

Step 1. Let $j \in \mathbb{N}_0$. Using convexity (10.1.9)

$$f(\boldsymbol{w}_j) - f(\boldsymbol{w}_*) \leq -\langle \nabla f(\boldsymbol{w}_j), \boldsymbol{w}_* - \boldsymbol{w}_j \rangle \leq \|\nabla f(\boldsymbol{w}_j)\| \|\boldsymbol{w}_* - \boldsymbol{w}_j\|. \tag{10.1.11}$$

By Lemma 10.9 and since $\boldsymbol{w}_0 \neq \boldsymbol{w}_*$ it holds $\|\boldsymbol{w}_* - \boldsymbol{w}_j\| \leq \|\boldsymbol{w}_* - \boldsymbol{w}_0\| \neq 0$, so that we obtain a lower bound on the gradient

$$\|\nabla f(\boldsymbol{w}_j)\|^2 \geq \frac{(f(\boldsymbol{w}_j) - f(\boldsymbol{w}_*))^2}{\|\boldsymbol{w}_* - \boldsymbol{w}_0\|^2}.$$

Lemma 10.5 then yields

$$\begin{aligned} f(\boldsymbol{w}_{j+1}) - f(\boldsymbol{w}_*) &\leq f(\boldsymbol{w}_j) - f(\boldsymbol{w}_*) - \Big(h - \frac{Lh^2}{2}\Big)\|\nabla f(\boldsymbol{w}_j)\|^2 \\ &\leq f(\boldsymbol{w}_j) - f(\boldsymbol{w}_*) - \Big(h - \frac{Lh^2}{2}\Big)\frac{(f(\boldsymbol{w}_j) - f(\boldsymbol{w}_*))^2}{\|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2}. \end{aligned}$$

With $e_j := f(\boldsymbol{w}_j) - f(\boldsymbol{w}_*)$ and $\omega := (h - Lh^2/2)/\|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2$ this reads

$$e_{j+1} \leq e_j - \omega e_j^2 = e_j \cdot (1 - \omega e_j), \tag{10.1.12}$$

which is valid for all $j \in \mathbb{N}_0$.

Step 2. By L-smoothness (10.1.4) and $\nabla f(\boldsymbol{w}_*) = 0$ it holds

$$f(\boldsymbol{w}_0) - f(\boldsymbol{w}_*) \leq \frac{L}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2, \tag{10.1.13}$$

which implies (10.1.10) for $k = 0$. It remains to show the bound for $k \in \mathbb{N}$.

Fix $k \in \mathbb{N}$. We may assume $e_k > 0$, since otherwise (10.1.10) is trivial. Then $e_j > 0$ for all $j = 0, \dots, k-1$ since $e_j = 0$ implies $e_i = 0$ for all $i > j$, contradicting $e_k > 0$. Moreover, $\omega e_j < 1$ for all $j = 0, \dots, k-1$, since $\omega e_j \geq 1$ implies $e_{j+1} \leq 0$ by (10.1.12), contradicting $e_{j+1} > 0$.

Using that $1/(1-x) \geq 1 + x$ for all $x \in [0, 1)$, (10.1.12) thus gives

$$\frac{1}{e_{j+1}} \geq \frac{1}{e_j}(1 + \omega e_j) = \frac{1}{e_j} + \omega \qquad \text{for all } j = 0, \dots, k-1.$$

Hence

$$\frac{1}{e_k} - \frac{1}{e_0} = \sum_{j=0}^{k-1} \Big(\frac{1}{e_{j+1}} - \frac{1}{e_j}\Big) \geq k\omega$$

and

$$f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*) = e_k \leq \frac{1}{\frac{1}{e_0} + k\omega} = \frac{1}{\frac{1}{f(\boldsymbol{w}_0) - f(\boldsymbol{w}_*)} + k\frac{(h - Lh^2/2)}{\|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2}}.$$

Using (10.1.13) we get

$$f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*) \leq \frac{\|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2}{\frac{2}{L} + kh \cdot (1 - \frac{Lh}{2})} = O(k^{-1}). \tag{10.1.14}$$

Finally, (10.1.10) follows by plugging in $h = 1/L$. □

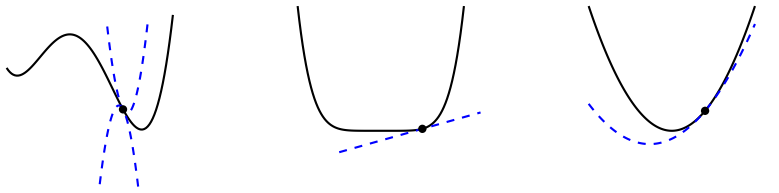


Figure 10.2: The graph of L-smooth functions lies between two quadratic functions at each point, see (10.1.4) and (10.1.5), the graph of convex functions lies above the tangent at each point, see (10.1.9), and the graph of μ-strongly convex functions lies above a quadratic function at each point, see (10.1.15).

Remark 10.12. The step size $h = 1/L$ is again such that the upper bound in (10.1.14) is minimized.

We emphasize, that while under the assumptions of Theorem 10.11 it holds $f(\boldsymbol{w}_k) \to f(\boldsymbol{w}_*)$, in general it is not true that $\boldsymbol{w}_k \to \boldsymbol{w}_*$ as $k \to \infty$. To show the convergence of the $\boldsymbol{w}_k$, we need to introduce stronger assumptions that guarantee the existence of a unique minimizer.

10.1.3 Strong convexity

To obtain faster convergence and guarantee the existence of unique minimizers, we next introduce the notion of strong convexity. As the terminology suggests, strong convexity implies convexity; specifically, while convexity requires f to be lower bounded by the linearization around each point, strongly convex functions are lower bounded by the linearization plus a positive quadratic term.

Definition 10.13. Let $n \in \mathbb{N}$ and $\mu > 0$. A function $f \in C^1(\mathbb{R}^n)$ is called **μ-strongly convex** if

$$f(\boldsymbol{v}) \geq f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle + \frac{\mu}{2}\|\boldsymbol{v} - \boldsymbol{w}\|^2 \qquad \text{for all } \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n. \tag{10.1.15}$$

Note that (10.1.15) is the opposite of the bound (10.1.4) implied by L-smoothness. We depict the three notions of L-smoothness, convexity, and μ-strong convexity in Figure 10.2.

Every μ-strongly convex function has a unique minimizer. To see this note first that (10.1.15) implies f to be lower bounded by a convex quadratic function, so that there exists at least one minimizer $\boldsymbol{w}_*$, and $\nabla f(\boldsymbol{w}_*) = 0$. By (10.1.15) we then have $f(\boldsymbol{v}) > f(\boldsymbol{w}_*)$ for every $\boldsymbol{v} \neq \boldsymbol{w}_*$.

The next theorem shows that the gradient descent iterates converge linearly towards the unique minimizer for L-smooth and μ-strongly convex functions. Recall that a sequence e_k is said to **converge linearly** to 0, if and only if there exist constants $C > 0$ and $c \in [0, 1)$ such that

$$e_k \leq Cc^k \qquad \text{for all } k \in \mathbb{N}_0.$$

The constant c is also referred to as the **rate of convergence**. Before giving the statement, we first note that comparing (10.1.4) and (10.1.15) it necessarily holds $L \geq \mu$ and therefore $\kappa := L/\mu \geq 1$. This term is known as the **condition number** of f. It crucially influences the rate of convergence.

Theorem 10.14. *Let $n \in \mathbb{N}$ and $L \geq \mu > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth and μ-strongly convex. Further, let $h_k = h \in (0, 1/L]$ for all $k \in \mathbb{N}_0$, let $(\boldsymbol{w}_k)_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be defined by (10.1.2), and let $\boldsymbol{w}_*$ be the unique minimizer of f.*

Then, $f(\boldsymbol{w}_k) \to f(\boldsymbol{w}_*)$ and $\boldsymbol{w}_k \to \boldsymbol{w}_*$ converge linearly for $k \to \infty$. For the specific choice $h = 1/L$

$$\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 \leq \Big(1 - \frac{\mu}{L}\Big)^k \|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2 \tag{10.1.16a}$$

$$f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*) \leq \frac{L}{2}\Big(1 - \frac{\mu}{L}\Big)^k \|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2. \tag{10.1.16b}$$

Proof. It suffices to show (10.1.16a) since (10.1.16b) follows directly by Lemma 10.3 and because $\nabla f(\boldsymbol{w}_*) = 0$. The case $k = 0$ is trivial, so let $k \in \mathbb{N}$.

Expanding $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - h\nabla f(\boldsymbol{w}_{k-1})$ and using μ-strong convexity (10.1.15)

$$\begin{aligned}\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 &= \|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2 - 2h\,\langle \nabla f(\boldsymbol{w}_{k-1}), \boldsymbol{w}_{k-1} - \boldsymbol{w}_*\rangle + h^2\|\nabla f(\boldsymbol{w}_{k-1})\|^2 \\ &\leq (1 - \mu h)\|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2 - 2h \cdot (f(\boldsymbol{w}_{k-1}) - f(\boldsymbol{w}_*)) + h^2\|\nabla f(\boldsymbol{w}_{k-1})\|^2.\end{aligned}$$

Moreover, the descent property in Lemma 10.5 gives

$$\begin{aligned}&- 2h \cdot (f(\boldsymbol{w}_{k-1}) - f(\boldsymbol{w}_*)) + h^2\|\nabla f(\boldsymbol{w}_{k-1})\|^2 \\ &\quad\leq -2h \cdot (f(\boldsymbol{w}_{k-1}) - f(\boldsymbol{w}_*)) + \frac{h^2}{h \cdot (1 - Lh/2)}(f(\boldsymbol{w}_{k-1}) - f(\boldsymbol{w}_k)).\end{aligned} \tag{10.1.17}$$

The descent property also implies $f(\boldsymbol{w}_{k-1}) - f(\boldsymbol{w}_*) \geq f(\boldsymbol{w}_{k-1}) - f(\boldsymbol{w}_k)$. Thus the right-hand side of (10.1.17) is less or equal to zero as long as $2h \geq h/(1 - Lh/2)$, which is equivalent to $h \leq 1/L$. Hence

$$\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 \leq (1 - \mu h)\|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2 \leq \cdots \leq (1 - \mu h)^k \|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2.$$

This concludes the proof. □

Remark 10.15. With a more refined argument, see [159, Theorem 2.1.15], the constraint on the step size can be relaxed to $h \leq 2/(\mu + L)$. For $h = 2/(\mu + L)$ one then obtains (10.1.16) with $1 - \mu/L = 1 - \kappa^{-1}$ replaced by

$$\Big(\frac{L/\mu - 1}{L/\mu + 1}\Big)^2 = \Big(\frac{\kappa - 1}{\kappa + 1}\Big)^2 \in [0, 1). \tag{10.1.18}$$

We have

$$\Big(\frac{\kappa - 1}{\kappa + 1}\Big)^2 = 1 - 4\kappa^{-1} + O(\kappa^{-2})$$

as $\kappa \to \infty$. Thus, (10.1.18) gives a slightly better, but conceptually similar, rate of convergence than $1 - \kappa^{-1}$ shown in Theorem 10.14.

10.1.4 PL-inequality

Linear convergence for gradient descent can also be shown under a weaker assumption known as the Polyak-Łojasiewicz-inequality, or PL-inequality for short.

Lemma 10.16. *Let $n \in \mathbb{N}$ and $\mu > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be μ-strongly convex and denote its unique minimizer by $\boldsymbol{w}_*$. Then f satisfies the* **PL-inequality**

$$\mu \cdot (f(\boldsymbol{w}) - f(\boldsymbol{w}_*)) \leq \frac{1}{2}\|\nabla f(\boldsymbol{w})\|^2 \qquad \text{for all } \boldsymbol{w} \in \mathbb{R}^n. \tag{10.1.19}$$

Proof. By μ-strong convexity we have

$$f(\boldsymbol{v}) \geq f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w}\rangle + \frac{\mu}{2}\|\boldsymbol{v} - \boldsymbol{w}\|^2 \qquad \text{for all } \boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^n. \tag{10.1.20}$$

The gradient of the right-hand side with respect to $\boldsymbol{v}$ equals $\nabla f(\boldsymbol{w}) + \mu \cdot (\boldsymbol{v} - \boldsymbol{w})$. This implies that the minimum of this expression is attained at $\boldsymbol{v} = \boldsymbol{w} - \nabla f(\boldsymbol{w})/\mu$. Minimizing both sides of (10.1.20) in $\boldsymbol{v}$ we thus find

$$f(\boldsymbol{w}_*) \geq f(\boldsymbol{w}) - \frac{1}{\mu}\|\nabla f(\boldsymbol{w})\|^2 + \frac{1}{2\mu}\|\nabla f(\boldsymbol{w})\|^2 = f(\boldsymbol{w}) - \frac{1}{2\mu}\|\nabla f(\boldsymbol{w})\|^2.$$

Rearranging the terms gives (10.1.19). □

As the lemma states, the PL-inequality is implied by strong convexity. Moreover, it is indeed weaker than strong convexity, and does not even imply convexity, see Exercise 10.28. The next theorem, which corresponds to [220, Theorem 1], gives a convergence result for L-smooth functions satisfying the PL-inequality. It therefore does *not require convexity*. The proof is left as an exercise. We only note that the PL-inequality bounds the distance to the minimal value of the objective function by the squared norm of the gradient. It is thus precisely the type of bound required to show convergence of gradient descent.

Theorem 10.17. *Let $n \in \mathbb{N}$ and $L > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth. Further, let $h_k = 1/L$ for all $k \in \mathbb{N}_0$, and let $(\boldsymbol{w}_k)_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be defined by* (10.1.2)*, and let $\boldsymbol{w}_*$ be a (not necessarily unique) minimizer of f, so that the PL-inequality* (10.1.19) *holds.*

Then, it holds for all $k \in \mathbb{N}_0$ that

$$f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*) \leq \Big(1 - \frac{\mu}{L}\Big)^k (f(\boldsymbol{w}_0) - f(\boldsymbol{w}_*)).$$

10.2 Stochastic gradient descent (SGD)

We next discuss a stochastic variant of gradient descent. The idea, which originally goes back to Robbins and Monro [191], is to replace the gradient $\nabla f(\boldsymbol{w}_k)$ in (10.1.2) by a random variable that

we denote by $\boldsymbol{G}_k$. We interpret $\boldsymbol{G}_k$ as an approximation to $\nabla f(\boldsymbol{w}_k)$; specifically, throughout we will assume that (given $\boldsymbol{w}_k$) $\boldsymbol{G}_k$ is an unbiased estimator, i.e.

$$\mathbb{E}[\boldsymbol{G}_k|\boldsymbol{w}_k] = \nabla f(\boldsymbol{w}_k). \tag{10.2.1}$$

After choosing some initial value $\boldsymbol{w}_0 \in \mathbb{R}^n$, the update rule becomes

$$\boldsymbol{w}_{k+1} := \boldsymbol{w}_k - h_k \boldsymbol{G}_k, \tag{10.2.2}$$

where $h_k > 0$ denotes again the step size, and unlike in Section 10.1, we focus here on the case of h_k depending on k. The iteration (10.2.2) creates a Markov chain $(\boldsymbol{w}_0, \boldsymbol{w}_1, \dots)$, meaning that $\boldsymbol{w}_k$ is a random variable, and its state only depends[1] on $\boldsymbol{w}_{k-1}$. The main reason for replacing the actual gradient by an estimator, is not to improve the accuracy or convergence rate, but rather to *decrease the computational cost and storage requirements* of the algorithm. The underlying assumption is that $\boldsymbol{G}_{k-1}$ can be computed at a fraction of the cost required for the computation of $\nabla f(\boldsymbol{w}_{k-1})$. The next example illustrates this in the standard setting.

Example 10.18 (Empirical risk minimization)**.** Suppose we have some data $S := (\boldsymbol{x}_j, y_j)_{j=1}^m$, where $y_j \in \mathbb{R}$ is the label corresponding to the data point $\boldsymbol{x}_j \in \mathbb{R}^d$. Using the square loss, we wish to fit a neural network $\Phi(\cdot, \boldsymbol{w}) : \mathbb{R}^d \to \mathbb{R}$ depending on parameters (i.e. weights and biases) $\boldsymbol{w} \in \mathbb{R}^n$, such that the empirical risk

$$f(\boldsymbol{w}) := \hat{\mathcal{R}}_S(\boldsymbol{w}) = \frac{1}{2m} \sum_{j=1}^m (\Phi(\boldsymbol{x}_j, \boldsymbol{w}) - y_j)^2,$$

is minimized. Performing one step of gradient descent requires the computation of

$$\nabla f(\boldsymbol{w}) = \frac{1}{m} \sum_{j=1}^m (\Phi(\boldsymbol{x}_j, \boldsymbol{w}) - y_j) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_j, \boldsymbol{w}), \tag{10.2.3}$$

and thus the computation of m gradients of the neural network Φ. For large m (in practice m can be in the millions or even larger), this computation might be infeasible. To decrease computational complexity, we replace the full gradient (10.2.3) by

$$\boldsymbol{G} := (\Phi(\boldsymbol{x}_j, \boldsymbol{w}) - y_j) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_j, \boldsymbol{w})$$

where $j \sim \text{uniform}(1, \dots, m)$ is a random variable with uniform distribution on the discrete set $\{1, \dots, m\}$. Then

$$\mathbb{E}[\boldsymbol{G}] = \frac{1}{m} \sum_{j=1}^m (\Phi(\boldsymbol{x}_j, \boldsymbol{w}) - y_j) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_j, \boldsymbol{w}) = \nabla f(\boldsymbol{w}),$$

but an evaluation of $\boldsymbol{G}$ merely requires the computation of a single gradient of the neural network. More general, one can choose a **mini-batch** size m_b (where $m_b \ll m$) and let $\boldsymbol{G} = \frac{1}{m_b} \sum_{j \in J} (\Phi(\boldsymbol{x}_j, \boldsymbol{w}) - y_j) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_j, \boldsymbol{w})$, where J is a random subset of $\{1, \dots, m\}$ of cardinality m_b.

[1] More precisely, given $\boldsymbol{w}_{k-1}$, the state of $\boldsymbol{w}_k$ is conditionally independent of $\boldsymbol{w}_1, \dots, \boldsymbol{w}_{k-2}$. See Appendix A.3.3.

Remark 10.19. In practice, the following variant is also common: Let $m_b k = m$ for m_b, k, $m \in \mathbb{N}$, i.e. the number of data points m is a k-fold multiple of the mini-batch size m_b. In each **epoch**, first a random partition $\dot{\bigcup}_{i=1}^{k} J_i = \{1, \dots, m\}$ is determined. Then for each $i = 1, \dots, k$, the weights are updated with the gradient estimate

$$\frac{1}{m_b} \sum_{j \in J_i} \Phi(\boldsymbol{x}_j - y_j, \boldsymbol{w}) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_j, \boldsymbol{w}).$$

Hence, in one epoch (which corresponds to k updates of the neural network weights), the algorithm sweeps through the whole dataset.

SGD can be analyzed in various settings. To give the general idea, we concentrate on the case of L-smooth and μ-strongly convex objective functions. Let us start by looking at a property akin to the (descent) Lemma 10.5. Using Lemma 10.3

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) - h_k \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{G}_k \rangle + h_k^2 \frac{L}{2} \|\boldsymbol{G}_k\|^2.$$

In contrast to gradient descent, we cannot say anything about the sign of the term in the middle of the right-hand side. Thus, (10.2.2) need not necessarily decrease the value of the objective function in every step. The key insight is that *in expectation* the value is still decreased under certain assumptions, namely

$$\begin{aligned}
\mathbb{E}[f(\boldsymbol{w}_{k+1}) | \boldsymbol{w}_k] &\leq f(\boldsymbol{w}_k) - h_k \mathbb{E}[\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{G}_k \rangle | \boldsymbol{w}_k] + h_k^2 \frac{L}{2} \mathbb{E}\left[\|\boldsymbol{G}_k\|^2 \middle| \boldsymbol{w}_k \right] \\
&= f(\boldsymbol{w}_k) - h_k \|\nabla f(\boldsymbol{w}_k)\|^2 + h_k^2 \frac{L}{2} \mathbb{E}\left[\|\boldsymbol{G}_k\|^2 \middle| \boldsymbol{w}_k \right] \\
&= f(\boldsymbol{w}_k) - h_k \left(\|\nabla f(\boldsymbol{w}_k)\|^2 - h_k \frac{L}{2} \mathbb{E}[\|\boldsymbol{G}_k\|^2 | \boldsymbol{w}_k] \right)
\end{aligned}$$

where we used (10.2.1).

Assuming, for some fixed $\gamma > 0$, the uniform bound

$$\mathbb{E}[\|\boldsymbol{G}_k\|^2 | \boldsymbol{w}_k] \leq \gamma$$

and that $\|\nabla f(\boldsymbol{w}_k)\| > 0$ (which is true unless $\boldsymbol{w}_k$ is the minimizer), upon choosing

$$0 < h_k < \frac{2 \|\nabla f(\boldsymbol{w}_k)\|^2}{L \gamma},$$

the expectation of the objective function decreases. Since $\nabla f(\boldsymbol{w}_k)$ tends to 0 as we approach the minimum, this also indicates that we should choose step sizes h_k that tend to 0 for $k \to \infty$. For our analysis we will work with the specific choice

$$h_k := \frac{1}{\mu} \frac{(k+1)^2 - k^2}{(k+1)^2} \quad \text{for all} \quad k \in \mathbb{N}_0, \tag{10.2.4}$$

as, e.g., in [76]. Note that

$$h_k = \frac{2k+1}{\mu(k+1)^2} = \frac{2}{\mu(k+1)} + O(k^{-2}) = O(k^{-1}).$$

Since $\boldsymbol{w}_k$ is a random variable by construction, a convergence statement can only be stochastic, e.g., in expectation or with high probability. We concentrate here on the former, but emphasize that also the latter can be shown.

Theorem 10.20. *Let $n \in \mathbb{N}$ and L, μ, $\gamma > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth and μ-strongly convex. Let $(h_k)_{k=0}^\infty$ satisfy (10.2.4) and let $(\boldsymbol{G}_k)_{k=0}^\infty$, $(\boldsymbol{w}_k)_{k=0}^\infty$ be sequences of random variables satisfying (10.2.1) and (10.2.2). Assume that $\mathbb{E}[\|\boldsymbol{G}_k\|^2|\boldsymbol{w}_k] \le \gamma$ for all $k \in \mathbb{N}_0$.*

Then

$$\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2] \le \frac{4\gamma}{\mu^2 k} = O(k^{-1}),$$
$$\mathbb{E}[f(\boldsymbol{w}_k)] - f(\boldsymbol{w}_*) \le \frac{4L\gamma}{2\mu^2 k} = O(k^{-1})$$

for $k \to \infty$.

Proof. We proceed similar as in the proof of Theorem 10.14. It holds for $k \ge 1$

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-1}] \\
&\quad = \|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2 - 2h_{k-1}\mathbb{E}[\langle \boldsymbol{G}_{k-1}, \boldsymbol{w}_{k-1} - \boldsymbol{w}_*\rangle\,|\boldsymbol{w}_{k-1}] + h_{k-1}^2\mathbb{E}[\|\boldsymbol{G}_{k-1}\|^2|\boldsymbol{w}_{k-1}] \\
&\quad = \|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2 - 2h_{k-1}\,\langle \nabla f(\boldsymbol{w}_{k-1}), \boldsymbol{w}_{k-1} - \boldsymbol{w}_*\rangle + h_{k-1}^2\mathbb{E}[\|\boldsymbol{G}_{k-1}\|^2|\boldsymbol{w}_{k-1}].
\end{aligned}$$

By μ-strong convexity (10.1.15)

$$\begin{aligned}
-2h_{k-1}\,\langle \nabla f(\boldsymbol{w}_{k-1}), \boldsymbol{w}_{k-1} - \boldsymbol{w}_*\rangle &\le -\mu h_{k-1}\|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2 - 2h_{k-1}\cdot(f(\boldsymbol{w}_{k-1}) - f(\boldsymbol{w}_*)) \\
&\le -\mu h_{k-1}\|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2.
\end{aligned}$$

Thus

$$\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-1}] \le (1 - \mu h_{k-1})\|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2 + h_{k-1}^2\gamma.$$

Using the Markov property, we have

$$\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-1}, \boldsymbol{w}_{k-2}] = \mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-1}]$$

so that

$$\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-1}] \le (1 - \mu h_{k-1})\mathbb{E}[\|\boldsymbol{w}_{k-1} - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-2}] + h_{k-1}^2\gamma.$$

With $e_0 := \|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2$ and $e_k := \mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-1}]$ for $k \ge 1$ we have found

$$\begin{aligned}
e_k &\le (1 - \mu h_{k-1})e_{k-1} + h_{k-1}^2\gamma \\
&\le (1 - \mu h_{k-1})((1 - \mu h_{k-2})e_{k-2} + h_{k-2}^2\gamma) + h_{k-1}^2\gamma \\
&\le \cdots \le e_0\prod_{j=0}^{k-1}(1 - \mu h_j) + \gamma\sum_{j=0}^{k-1} h_j^2\prod_{i=j+1}^{k-1}(1 - \mu h_i).
\end{aligned}$$

By choice of h_i

$$\prod_{i=j}^{k-1}(1-\mu h_i) = \prod_{i=j}^{k-1}\frac{i^2}{(i+1)^2} = \frac{j^2}{k^2}$$

and thus

$$\begin{aligned} e_k &\le \frac{\gamma}{\mu^2}\sum_{j=0}^{k-1}\left(\frac{(j+1)^2-j^2}{(j+1)^2}\right)^2\frac{(j+1)^2}{k^2} \\ &\le \frac{\gamma}{\mu^2}\frac{1}{k^2}\sum_{j=0}^{k-1}\underbrace{\frac{(2j+1)^2}{(j+1)^2}}_{\le 4} \\ &\le \frac{\gamma}{\mu^2}\frac{4k}{k^2} \\ &\le \frac{4\gamma}{\mu^2 k}. \end{aligned}$$

Since $\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2]$ is the expectation of $\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2|\boldsymbol{w}_{k-1}]$ with respect to the random variable $\boldsymbol{w}_{k-1}$, and $e_0/k^2 + 4\gamma/(\mu^2 k)$ is a constant independent of $\boldsymbol{w}_{k-1}$, we obtain

$$\mathbb{E}[\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2] \le \frac{4\gamma}{\mu^2 k}.$$

Finally, using L-smoothness

$$f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*) \le \langle \nabla f(\boldsymbol{w}_*), \boldsymbol{w}_k - \boldsymbol{w}_*\rangle + \frac{L}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 = \frac{L}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2,$$

and taking the expectation concludes the proof. □

The specific choice of h_k in (10.2.4) simplifies the calculations in the proof, but it is not necessary in order for the asymptotic convergence to hold. One can show similar convergence results with $h_k = c_1/(c_2 + k)$ under certain assumptions on c_1, c_2, e.g. [23, Theorem 4.7].

10.3 Backpropagation

We now explain how to apply gradient-based methods to the training of neural networks. Let $\Phi \in \mathcal{N}_{d_0}^{d_{L+1}}(\sigma; L, n)$ (see Definition 3.6) and assume that the activation function satisfies $\sigma \in C^1(\mathbb{R})$. As earlier, we denote the neural network parameters by

$$\boldsymbol{w} = ((\boldsymbol{W}^{(0)}, \boldsymbol{b}^{(0)}), \dots, (\boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)})) \tag{10.3.1}$$

with weight matrices $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}\times d_\ell}$ and bias vectors $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$. Additionally, we fix a differentiable loss function $\mathcal{L} : \mathbb{R}^{d_{L+1}} \times \mathbb{R}^{d_{L+1}} \to \mathbb{R}$, e.g., $\mathcal{L}(\boldsymbol{w}, \tilde{\boldsymbol{w}}) = \|\boldsymbol{w} - \tilde{\boldsymbol{w}}\|^2/2$, and assume given data $(\boldsymbol{x}_j, \boldsymbol{y}_j)_{j=1}^m \subseteq \mathbb{R}^{d_0} \times \mathbb{R}^{d_{L+1}}$. The goal is to minimize an empirical risk of the form

$$f(\boldsymbol{w}) := \frac{1}{m}\sum_{j=1}^m \mathcal{L}(\Phi(\boldsymbol{x}_j, \boldsymbol{w}), \boldsymbol{y}_j)$$

as a function of the neural network parameters $\boldsymbol{w}$. An application of the gradient step (10.1.2) to update the parameters requires the computation of

$$\nabla f(\boldsymbol{w}) = \frac{1}{m}\sum_{j=1}^{m} \nabla_{\boldsymbol{w}} \mathcal{L}(\Phi(\boldsymbol{x}_j, \boldsymbol{w}), \boldsymbol{y}_j).$$

For stochastic methods, as explained in Example 10.18, we only compute the average over a (random) subbatch of the dataset. In either case, we need an algorithm to determine $\nabla_{\boldsymbol{w}} \mathcal{L}(\Phi(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y})$, i.e. the gradients

$$\nabla_{\boldsymbol{b}^{(\ell)}} \mathcal{L}(\Phi(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y}) \in \mathbb{R}^{d_{\ell+1}}, \quad \nabla_{\boldsymbol{W}^{(\ell)}} \mathcal{L}(\Phi(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y}) \in \mathbb{R}^{d_{\ell+1} \times d_\ell} \tag{10.3.2}$$

for all $\ell = 0, \dots, L$.

The backpropagation algorithm [197] provides an *efficient* way to do so. To explain it, for fixed input $\boldsymbol{x} \in \mathbb{R}^{d_0}$ introduce the notation

$$\bar{\boldsymbol{x}}^{(1)} := \boldsymbol{W}^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)} \tag{10.3.3a}$$

$$\bar{\boldsymbol{x}}^{(\ell+1)} := \boldsymbol{W}^{(\ell)}\sigma(\bar{\boldsymbol{x}}^{(\ell)}) + \boldsymbol{b}^{(\ell)} \qquad \text{for } \ell \in \{1, \dots, L\}, \tag{10.3.3b}$$

where the application of $\sigma : \mathbb{R} \to \mathbb{R}$ to a vector is, as always, understood componentwise. With the notation of Definition 2.1, $\boldsymbol{x}^{(\ell)} = \sigma(\bar{\boldsymbol{x}}^{(\ell)}) \in \mathbb{R}^{d_\ell}$ for $\ell = 1, \dots, L$ and $\bar{\boldsymbol{x}}^{(L+1)} = \boldsymbol{x}^{(L+1)} = \Phi(\boldsymbol{x}, \boldsymbol{w}) \in \mathbb{R}^{d_{L+1}}$ is the output of the neural network. Therefore, the $\bar{\boldsymbol{x}}^{(\ell)}$ for $\ell = 1, \dots, L$ are sometimes also referred to as the *preactivations*.

In the following, we additionally fix $\boldsymbol{y} \in \mathbb{R}^{d_{L+1}}$ and write

$$\mathcal{L} := \mathcal{L}(\Phi(\boldsymbol{x}, \boldsymbol{w}), \boldsymbol{y}) = \mathcal{L}(\bar{\boldsymbol{x}}^{(L+1)}, \boldsymbol{y}).$$

Note that $\bar{\boldsymbol{x}}^{(k)}$ depends on $(\boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(\ell)})$ only if $k > \ell$. Since $\bar{\boldsymbol{x}}^{(\ell+1)}$ is a function of $\bar{\boldsymbol{x}}^{(\ell)}$ for each ℓ, by repeated application of the chain rule

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(\ell)}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{x}}^{(L+1)}}}_{\in \mathbb{R}^{1 \times d_{L+1}}} \underbrace{\frac{\partial \bar{\boldsymbol{x}}^{(L+1)}}{\partial \bar{\boldsymbol{x}}^{(L)}}}_{\in \mathbb{R}^{d_{L+1} \times d_L}} \cdots \underbrace{\frac{\partial \bar{\boldsymbol{x}}^{(\ell+2)}}{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}}_{\in \mathbb{R}^{d_{\ell+2} \times d_{\ell+1}}} \underbrace{\frac{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}{\partial W_{ij}^{(\ell)}}}_{\in \mathbb{R}^{d_{\ell+1} \times 1}}. \tag{10.3.4}$$

An analogous calculation holds for $\partial \mathcal{L}/\partial b_j^{(\ell)}$. Since all terms in (10.3.4) are easy to compute (see (10.3.3)), in principle we could use this formula to determine the gradients in (10.3.2). To avoid unnecessary computations, the main idea of backpropagation is to introduce

$$\boldsymbol{\alpha}^{(\ell)} := \nabla_{\bar{\boldsymbol{x}}^{(\ell)}} \mathcal{L} \in \mathbb{R}^{d_\ell} \qquad \text{for all } \ell = 1, \dots, L+1$$

and observe that

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(\ell)}} = (\boldsymbol{\alpha}^{(\ell+1)})^\top \frac{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}{\partial W_{ij}^{(\ell)}}.$$

As the following lemma shows, the $\boldsymbol{\alpha}^{(\ell)}$ can be computed recursively for $\ell = L+1, \dots, 1$. This explains the name "backpropagation". In the following, $\odot$ denotes the componentwise (Hadamard) product, i.e. $\boldsymbol{a} \odot \boldsymbol{b} = (a_i b_i)_{i=1}^d$ for every $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{R}^d$.

Lemma 10.21. *It holds*

$$\boldsymbol{\alpha}^{(L+1)} = \nabla_{\bar{\boldsymbol{x}}^{(L+1)}} \mathcal{L}(\bar{\boldsymbol{x}}^{(L+1)}, \boldsymbol{y}) \tag{10.3.5}$$

and

$$\boldsymbol{\alpha}^{(\ell)} = \sigma'(\bar{\boldsymbol{x}}^{(\ell)}) \odot (\boldsymbol{W}^{(\ell)})^\top \boldsymbol{\alpha}^{(\ell+1)} \qquad \text{for all } \ell = L, \dots, 1.$$

Proof. Equation (10.3.5) holds by definition. For $\ell \in \{1, \dots, L\}$ by the chain rule

$$\boldsymbol{\alpha}^{(\ell)} = \frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{x}}^{(\ell)}} = \underbrace{\Big(\frac{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}{\partial \bar{\boldsymbol{x}}^{(\ell)}}\Big)^\top}_{\in \mathbb{R}^{d_\ell \times d_{\ell+1}}} \underbrace{\frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}}_{\in \mathbb{R}^{d_{\ell+1} \times 1}} = \Big(\frac{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}{\partial \bar{\boldsymbol{x}}^{(\ell)}}\Big)^\top \boldsymbol{\alpha}^{(\ell+1)}.$$

By (10.3.3b) for $i \in \{1, \dots, d_{\ell+1}\}$, $j \in \{1, \dots, d_\ell\}$

$$\Big(\frac{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}{\partial \bar{\boldsymbol{x}}^{(\ell)}}\Big)_{ij} = \frac{\partial \bar{x}_i^{(\ell+1)}}{\partial \bar{x}_j^{(\ell)}} = W_{ij}^{(\ell)} \sigma'(\bar{x}_j^{(\ell)}).$$

Thus the claim follows. □

Putting everything together, we obtain explicit formulas for (10.3.2).

Proposition 10.22. *It holds*

$$\nabla_{\boldsymbol{b}^{(\ell)}} \mathcal{L} = \boldsymbol{\alpha}^{(\ell+1)} \in \mathbb{R}^{d_{\ell+1}} \qquad \text{for } \ell = 0, \dots, L$$

and

$$\nabla_{\boldsymbol{W}^{(0)}} \mathcal{L} = \boldsymbol{\alpha}^{(1)} \boldsymbol{x}^\top \in \mathbb{R}^{d_1 \times d_0}$$

and

$$\nabla_{\boldsymbol{W}^{(\ell)}} \mathcal{L} = \boldsymbol{\alpha}^{(\ell+1)} \sigma(\bar{\boldsymbol{x}}^{(\ell)})^\top \in \mathbb{R}^{d_{\ell+1} \times d_\ell} \qquad \text{for } \ell = 1, \dots, L.$$

Proof. By (10.3.3a) for i, $k \in \{1, \dots, d_1\}$, and $j \in \{1, \dots, d_0\}$

$$\frac{\partial \bar{x}_k^{(1)}}{\partial b_i^{(0)}} = \delta_{ki} \qquad \text{and} \qquad \frac{\partial \bar{x}_k^{(1)}}{\partial W_{ij}^{(0)}} = \delta_{ki} x_j,$$

and by (10.3.3b) for $\ell \in \{1, \dots, L\}$ and i, $k \in \{1, \dots, d_{\ell+1}\}$, and $j \in \{1, \dots, d_\ell\}$

$$\frac{\partial \bar{x}_k^{(\ell+1)}}{\partial b_i^{(\ell)}} = \delta_{ki} \qquad \text{and} \qquad \frac{\partial \bar{x}_k^{(\ell+1)}}{\partial W_{ij}^{(\ell)}} = \delta_{ki} \sigma(\bar{x}_j^{(\ell)}).$$

Thus, with $\boldsymbol{e}_i = (\delta_{ki})_{k=1}^{d_{\ell+1}}$

$$\frac{\partial \mathcal{L}}{\partial b_i^{(\ell)}} = \Big(\frac{\partial \bar{\boldsymbol{x}}^{(\ell+1)}}{\partial b_i^{(\ell)}}\Big)^\top \frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{x}}^{(\ell+1)}} = \boldsymbol{e}_i^\top \boldsymbol{\alpha}^{(\ell+1)} = \alpha_i^{(\ell+1)} \qquad \text{for } \ell \in \{0, \dots, L\}$$

and similarly

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(0)}} = \Big(\frac{\partial \bar{\boldsymbol{x}}^{(1)}}{\partial W_{ij}^{(0)}}\Big)^\top \boldsymbol{\alpha}^{(1)} = \bar{x}_j^{(0)} \boldsymbol{e}_i^\top \boldsymbol{\alpha}^{(1)} = \bar{x}_j^{(0)} \alpha_i^{(1)}$$

and

$$\frac{\partial \mathcal{L}}{\partial W_{ij}^{(\ell)}} = \sigma(\bar{x}_j^{(\ell)}) \alpha_i^{(\ell+1)} \qquad \text{for } \ell \in \{1, \dots, L\}.$$

This concludes the proof. □

Lemma 10.21 and Proposition 10.22 motivate Algorithm 1, in which a forward pass computing $\bar{\boldsymbol{x}}^{(\ell)}$, $\ell = 1, \dots, L+1$, is followed by a backward pass to determine the $\boldsymbol{\alpha}^{(\ell)}$, $\ell = L+1, \dots, 1$, and the gradients of $\mathcal{L}$ with respect to the neural network parameters. This shows how to use gradient-based optimizers from the previous sections for the training of neural networks.

Two important remarks are in order. First, the objective function associated to neural networks is typically not convex as a function of the neural network weights and biases. Thus, the analysis of the previous sections will in general not be directly applicable. It may still give some insight about the convergence behavior locally around the minimizer however. Second, to derive the back-propagation algorithm we assumed the activation function to be continuously differentiable, which does not hold for ReLU. Using the concept of subgradients, gradient-based algorithms and their analysis may be generalized to some extent to also accommodate non-differentiable loss functions, see Exercises 10.31–10.33.

10.4 Acceleration

Acceleration is an important tool for the training of neural networks [221]. The idea was first introduced by Polyak in 1964 under the name "heavy ball method" [180]. It is inspired by the dynamics of a heavy ball rolling down the valley of the loss landscape. Since then other types of acceleration have been proposed and analyzed, with Nesterov acceleration being the most prominent example [160]. In this section, we first give some intuition by discussing the heavy ball method for a simple quadratic loss. Afterwards we turn to Nesterov acceleration and give a convergence proof for L-smooth and μ-strongly convex objective functions that improves upon the bounds obtained for gradient descent.

10.4.1 Heavy ball method

We proceed similar as in [70, 181, 183] to motivate the idea. Consider the quadratic objective function in two dimensions

$$f(\boldsymbol{w}) := \frac{1}{2} \boldsymbol{w}^\top \boldsymbol{D} \boldsymbol{w} \qquad \text{where} \qquad \boldsymbol{D} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \tag{10.4.1}$$

Algorithm 1 Backpropagation

Input: Network input $\boldsymbol{x}$, target output $\boldsymbol{y}$, neural network parameters $((\boldsymbol{W}^{(0)}, \boldsymbol{b}^{(0)}), \dots, (\boldsymbol{W}^{(L)}, \boldsymbol{b}^{(L)}))$
Output: Gradients of the loss function $\mathcal{L}$ with respect to neural network parameters

Forward pass
$\bar{\boldsymbol{x}}^{(1)} \leftarrow \boldsymbol{W}^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)}$
for $\ell = 1, \dots, L$ **do**
  $\bar{\boldsymbol{x}}^{(\ell+1)} \leftarrow \boldsymbol{W}^{(\ell)}\sigma(\bar{\boldsymbol{x}}^{(\ell)}) + \boldsymbol{b}^{(\ell)}$
end for

Backward pass
$\boldsymbol{\alpha}^{(L+1)} \leftarrow \nabla_{\bar{\boldsymbol{x}}^{(L+1)}}\mathcal{L}(\bar{\boldsymbol{x}}^{(L+1)}, \boldsymbol{y})$
for $\ell = L, \dots, 1$ **do**
  $\nabla_{\boldsymbol{b}^{(\ell)}}\mathcal{L} \leftarrow \boldsymbol{\alpha}^{(\ell+1)}$
  $\nabla_{\boldsymbol{W}^{(\ell)}}\mathcal{L} \leftarrow \boldsymbol{\alpha}^{(\ell+1)}\sigma(\bar{\boldsymbol{x}}^{(\ell)})^\top$
  $\boldsymbol{\alpha}^{(\ell)} \leftarrow \sigma'(\bar{\boldsymbol{x}}^{(\ell)}) \odot (\boldsymbol{W}^{(\ell)})^\top\boldsymbol{\alpha}^{(\ell+1)}$
end for
$\nabla_{\boldsymbol{b}^{(0)}}\mathcal{L} \leftarrow \boldsymbol{\alpha}^{(1)}$
$\nabla_{\boldsymbol{W}^{(0)}}\mathcal{L} \leftarrow \boldsymbol{\alpha}^{(1)}\boldsymbol{x}^\top$

with $\lambda_1 \geq \lambda_2 > 0$. Clearly, f has a unique minimizer at $\boldsymbol{w}_* = \boldsymbol{0} \in \mathbb{R}^2$. Starting at some $\boldsymbol{w}_0 \in \mathbb{R}^2$, gradient descent with constant step size $h > 0$ computes the iterates

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - h\boldsymbol{D}\boldsymbol{w}_k = \begin{pmatrix} 1-h\lambda_1 & 0 \\ 0 & 1-h\lambda_2 \end{pmatrix}\boldsymbol{w}_k = \begin{pmatrix} (1-h\lambda_1)^{k+1} & 0 \\ 0 & (1-h\lambda_2)^{k+1} \end{pmatrix}\boldsymbol{w}_0.$$

The method converges for arbitrary initialization $\boldsymbol{w}_0$ if and only if

$$|1-h\lambda_1| < 1 \qquad \text{and} \qquad |1-h\lambda_2| < 1.$$

The optimal step size balancing the speed of convergence in both coordinates is

$$h_* = \operatorname{argmin}_{h>0} \max\{|1-h\lambda_1|, |1-h\lambda_2|\} = \frac{2}{\lambda_1+\lambda_2}. \tag{10.4.2}$$

With $\kappa = \lambda_1/\lambda_2$ we then obtain the convergence rate

$$|1-h_*\lambda_1| = |1-h_*\lambda_2| = \frac{\lambda_1-\lambda_2}{\lambda_1+\lambda_2} = \frac{\kappa-1}{\kappa+1} \in [0,1). \tag{10.4.3}$$

If $\lambda_1 \gg \lambda_2$, this term is close to 1, and thus the convergence will be slow. This is consistent with our analysis for strongly convex objective functions; by Exercise 10.34 the condition number of f equals $\kappa = \lambda_1/\lambda_2 \gg 1$. Hence, the upper bounds in Theorem 10.14 and Remark 10.15 converge only slowly. Similar considerations hold for general quadratic objective functions in $\mathbb{R}^n$ such as

$$\tilde{f}(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^\top\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}^\top\boldsymbol{w} + c \tag{10.4.4}$$

with $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ symmetric positive definite, $\boldsymbol{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$, see Exercise 10.35.

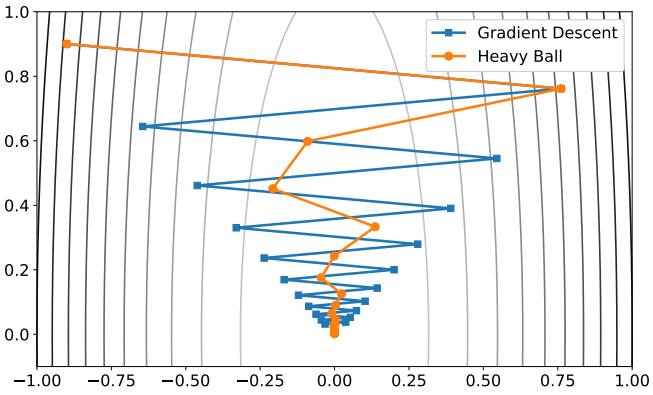


Figure 10.3: 20 steps of gradient descent and the heavy ball method on the objective function (10.4.1) with $\lambda_1 = 12 \gg 1 = \lambda_2$, step size $h = \alpha = h_*$ as in (10.4.2), and $\beta = 1/3$.

Remark 10.23. Interpreting (10.4.4) as a second-order Taylor expansion of some objective function $\tilde{f}$ around its minimizer $\boldsymbol{w}_*$, we note that the described effects also occur for general objective functions with ill-conditioned Hessians at the minimizer.

Figure 10.3 gives further insight into the poor performance of gradient descent for (10.4.1) with $\lambda_1 \gg \lambda_2$. The loss-landscape looks like a ravine (the derivative is much larger in one direction than the other), and away from the floor, ∇f mainly points to the opposite side. Therefore the iterates oscillate back and forth in the first coordinate, and make little progress in the direction of the valley along the second coordinate axis. To address this problem, the heavy ball method introduces a "momentum" term which can mitigate this effect to some extent. The idea is, to choose the update not just according to the gradient at the current location, but to add information from the previous steps. After initializing $\boldsymbol{w}_0$ and, e.g., $\boldsymbol{w}_1 = \boldsymbol{w}_0 - \alpha \nabla f(\boldsymbol{w}_0)$, let for $k \in \mathbb{N}$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha \nabla f(\boldsymbol{w}_k) + \beta(\boldsymbol{w}_k - \boldsymbol{w}_{k-1}). \tag{10.4.5}$$

This is known as Polyak's heavy ball method [180]. Here $\alpha > 0$ and $\beta \in (0, 1)$ are hyperparameters (that could also depend on k) and in practice need to be carefully tuned to balance the strength of the gradient and the momentum term. Iteratively expanding (10.4.5) with the given initialization, observe that for $k \geq 0$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha \left(\sum_{j=0}^{k} \beta^j \nabla f(\boldsymbol{w}_{k-j}) \right). \tag{10.4.6}$$

Thus, $\boldsymbol{w}_k$ is updated using an *exponentially weighted average* of all past gradients. Choosing the momentum parameter β in the interval $(0, 1)$ ensures that the influence of previous gradients on the update decays exponentially. The concrete value of β determines the balance between the impact of recent and past gradients.

Intuitively, this (exponentially weighted) linear combination of the past gradients averages out some of the oscillation observed for gradient descent in Figure 10.3 in the first coordinate, and thus "smoothes" the path. The partial derivative in the second coordinate, along which the objective

function is very flat, does not change much from one iterate to the next. Thus, its proportion in the update is strengthened through the addition of momentum. This is observed in Figure 10.3.

As mentioned earlier, the heavy ball method can be interpreted as a discretization of the dynamics of a ball rolling down the valley of the loss landscape. If the ball has positive mass, i.e. is "heavy", its momentum prevents the ball from bouncing back and forth too strongly. The following remark further elucidates this connection.

Remark 10.24. As pointed out, e.g., in [181, 183], for suitable choices of α and β, (10.4.5) can be interpreted as a discretization of the second-order ODE

$$m\boldsymbol{w}''(t) = -\nabla f(\boldsymbol{w}(t)) - r\boldsymbol{w}'(t). \tag{10.4.7}$$

This equation describes the movement of a point mass m under influence of the force field $-\nabla f(\boldsymbol{w}(t))$; the term $-\boldsymbol{w}'(t)$, which points in the negative direction of the current velocity, corresponds to friction, and $r > 0$ is the friction coefficient. The discretization

$$m\frac{\boldsymbol{w}_{k+1} - 2\boldsymbol{w}_k + \boldsymbol{w}_{k-1}}{h^2} = -\nabla f(\boldsymbol{w}_k) - \frac{\boldsymbol{w}_{k+1} - \boldsymbol{w}_k}{h}$$

then leads to

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \underbrace{\frac{h^2}{m - rh}}_{=\alpha}\nabla f(\boldsymbol{w}_k) + \underbrace{\frac{m}{m - rh}}_{=\beta}(\boldsymbol{w}_k - \boldsymbol{w}_{k-1}), \tag{10.4.8}$$

and thus to (10.4.5), [183].

Letting $m = 0$ in (10.4.8), we recover the gradient descent update (10.1.2). Hence, the positive mass corresponds to the momentum term. Similarly, letting $m = 0$ in the continuous dynamics (10.4.7), we obtain the gradient flow (10.1.3). The key difference between these equations is that $-\nabla f(\boldsymbol{w}(t))$ represents the *velocity* of $\boldsymbol{w}(t)$ in (10.1.3), whereas in (10.4.7), up to the friction term, it corresponds to an *acceleration*.

Let us sketch an argument to show that (10.4.5) improves the convergence over plain gradient descent for the objective function (10.4.1). Denoting $\boldsymbol{w}_k = (w_{k,1}, w_{k,2})^\top \in \mathbb{R}^2$, we obtain from (10.4.5) and the definition of f in (10.4.1)

$$\begin{pmatrix} w_{k+1,j} \\ w_{k,j} \end{pmatrix} = \begin{pmatrix} 1 + \beta - \alpha\lambda_j & -\beta \\ 1 & 0 \end{pmatrix} \begin{pmatrix} w_{k,j} \\ w_{k-1,j} \end{pmatrix} \tag{10.4.9}$$

for $j \in \{1, 2\}$ and $k \geq 1$. The smaller the modulus of the eigenvalues of the matrix in (10.4.9), the faster the convergence towards the minimizer $w_{*,j} = 0 \in \mathbb{R}$ for arbitrary initialization. Hence, the goal is to choose $\alpha > 0$ and $\beta \in (0, 1)$ such that the maximal modulus of the eigenvalues of the matrix for $j \in \{1, 2\}$ is possibly small. We omit the details of this calculation (also see [181, 165, 70]), but mention that this is obtained for

$$\alpha = \Big(\frac{2}{\sqrt{\lambda_1} + \sqrt{\lambda_2}}\Big)^2 \qquad \text{and} \qquad \beta = \Big(\frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1} + \sqrt{\lambda_2}}\Big)^2.$$

With these choices, the modulus of the maximal eigenvalue is bounded by

$$\sqrt{\beta} = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \in [0, 1),$$

where again $\kappa = \lambda_1/\lambda_2$. Due to (10.4.9), this expression gives a rate of convergence for (10.4.5). Contrary to gradient descent, see (10.4.3), for this problem the heavy ball method achieves a convergence rate that only depends on the *square root* of the condition number κ. This explains the improved performance observed in Figure 10.3.

10.4.2 Nesterov acceleration

Nesterov's accelerated gradient method (NAG) [160, 159], is a refinement of the heavy ball method. After initializing $\boldsymbol{v}_0$, $\boldsymbol{w}_0 \in \mathbb{R}^n$, the update is formulated as the two-step process

$$\boldsymbol{v}_{k+1} = \boldsymbol{w}_k - \alpha \nabla f(\boldsymbol{w}_k) \tag{10.4.10a}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{v}_{k+1} + \beta(\boldsymbol{v}_{k+1} - \boldsymbol{v}_k), \tag{10.4.10b}$$

where again $\alpha > 0$ and $\beta \in (0, 1)$ are hyperparameters. Substituting the second line into the first we get

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k - \alpha \nabla f(\boldsymbol{w}_k) + \beta(\boldsymbol{v}_k - \boldsymbol{v}_{k-1}).$$

Comparing with the heavy ball method (10.4.5), the key difference is that the gradient is not evaluated at the current position $\boldsymbol{v}_k$, but instead at the point $\boldsymbol{w}_k = \boldsymbol{v}_k + \beta(\boldsymbol{v}_k - \boldsymbol{v}_{k-1})$, which can be interpreted as an estimate of the position at the next iteration.

We next discuss the convergence for L-smooth and μ-strongly convex objective functions f. It turns out, that these conditions are not sufficient in order for the heavy ball method (10.4.5) to converge, and one can construct counterexamples [133]. This is in contrast to NAG, as the next theorem shows. To give the analysis, it is convenient to first rewrite (10.4.10) as a three sequence update: Let $\tau = \sqrt{\mu/L}$, $\alpha = 1/L$, and $\beta = (1-\tau)/(1+\tau)$. After initializing $\boldsymbol{w}_0$, $\boldsymbol{v}_0 \in \mathbb{R}^n$, (10.4.10) can also be written as $\boldsymbol{u}_0 = ((1+\tau)\boldsymbol{w}_0 - \boldsymbol{v}_0)/\tau$ and for $k \in \mathbb{N}_0$

$$\boldsymbol{w}_k = \frac{\tau}{1+\tau}\boldsymbol{u}_k + \frac{1}{1+\tau}\boldsymbol{v}_k \tag{10.4.11a}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{w}_k - \frac{1}{L}\nabla f(\boldsymbol{w}_k) \tag{10.4.11b}$$

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \tau \cdot (\boldsymbol{w}_k - \boldsymbol{u}_k) - \frac{\tau}{\mu}\nabla f(\boldsymbol{w}_k), \tag{10.4.11c}$$

see Exercise 10.36.

The proof of the following theorem proceeds along the lines of [231, 241].

Theorem 10.25. *Let $n \in \mathbb{N}$ and L, $\mu > 0$. Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be L-smooth and μ-strongly convex. Further, let $\boldsymbol{v}_0$, $\boldsymbol{w}_0 \in \mathbb{R}^n$ and let $\tau = \sqrt{\mu/L}$. Let $(\boldsymbol{w}_k, \boldsymbol{v}_{k+1}, \boldsymbol{u}_{k+1})_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be defined by* (10.4.11a)*, and let $\boldsymbol{w}_*$ be the unique minimizer of f.*

Then, for all $k \in \mathbb{N}_0$, it holds that

$$\|\boldsymbol{u}_k - \boldsymbol{w}_*\|^2 \leq \frac{2}{\mu}\Big(1 - \sqrt{\frac{\mu}{L}}\Big)^k \Big(f(\boldsymbol{v}_0) - f(\boldsymbol{w}_*) + \frac{\mu}{2}\|\boldsymbol{u}_0 - \boldsymbol{w}_*\|^2\Big), \tag{10.4.12a}$$

$$f(\boldsymbol{v}_k) - f(\boldsymbol{w}_*) \leq \Big(1 - \sqrt{\frac{\mu}{L}}\Big)^k \Big(f(\boldsymbol{v}_0) - f(\boldsymbol{w}_*) + \frac{\mu}{2}\|\boldsymbol{u}_0 - \boldsymbol{w}_*\|^2\Big). \tag{10.4.12b}$$

Proof. Define

$$e_k := f(\boldsymbol{v}_k) - f(\boldsymbol{w}_*) + \frac{\mu}{2}\|\boldsymbol{u}_k - \boldsymbol{w}_*\|^2. \tag{10.4.13}$$

To show (10.4.12), it suffices to prove with $c = 1 - \tau$ that $e_{k+1} \leq c e_k$ for all $k \in \mathbb{N}_0$.

We start with the last term in (10.4.13). By (10.4.11c)

$$\begin{aligned}
&\frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{w}_*\|^2 - \frac{\mu}{2}\|\boldsymbol{u}_k - \boldsymbol{w}_*\|^2 \\
&\quad = \frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k + \boldsymbol{u}_k - \boldsymbol{w}_*\|^2 - \frac{\mu}{2}\|\boldsymbol{u}_k - \boldsymbol{w}_*\|^2 \\
&\quad = \frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|^2 + \frac{\mu}{2} \cdot \left(2 \left\langle \tau \cdot (\boldsymbol{w}_k - \boldsymbol{u}_k) - \frac{\tau}{\mu}\nabla f(\boldsymbol{w}_k), \boldsymbol{u}_k - \boldsymbol{w}_* \right\rangle \right) \\
&\quad = \frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|^2 + \tau \left\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{w}_* - \boldsymbol{u}_k \right\rangle - \tau\mu \left\langle \boldsymbol{w}_k - \boldsymbol{u}_k, \boldsymbol{w}_* - \boldsymbol{u}_k \right\rangle .
\end{aligned} \tag{10.4.14}$$

From (10.4.11a) we have $\tau \boldsymbol{u}_k = (1+\tau)\boldsymbol{w}_k - \boldsymbol{v}_k$ so that

$$\tau \cdot (\boldsymbol{w}_k - \boldsymbol{u}_k) = \tau \boldsymbol{w}_k - (1+\tau)\boldsymbol{w}_k + \boldsymbol{v}_k = \boldsymbol{v}_k - \boldsymbol{w}_k \tag{10.4.15}$$

and using μ-strong convexity (10.1.15), we get

$$\begin{aligned}
\tau \left\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{w}_* - \boldsymbol{u}_k \right\rangle &= \tau \left\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{w}_k - \boldsymbol{u}_k \right\rangle + \tau \left\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{w}_* - \boldsymbol{w}_k \right\rangle \\
&\leq \left\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \right\rangle - \tau \cdot (f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)) - \frac{\tau\mu}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2.
\end{aligned}$$

Moreover,

$$\begin{aligned}
&-\frac{\tau\mu}{2}\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 - \tau\mu \left\langle \boldsymbol{w}_k - \boldsymbol{u}_k, \boldsymbol{w}_* - \boldsymbol{u}_k \right\rangle \\
&\qquad = -\frac{\tau\mu}{2}\Big(\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 - 2\left\langle \boldsymbol{w}_k - \boldsymbol{u}_k, \boldsymbol{w}_k - \boldsymbol{w}_* \right\rangle + 2\left\langle \boldsymbol{w}_k - \boldsymbol{u}_k, \boldsymbol{w}_k - \boldsymbol{u}_k \right\rangle \Big) \\
&\qquad = -\frac{\tau\mu}{2}(\|\boldsymbol{u}_k - \boldsymbol{w}_*\|^2 + \|\boldsymbol{w}_k - \boldsymbol{u}_k\|^2).
\end{aligned}$$

Thus, (10.4.14) is bounded by

$$\begin{aligned}
&\frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|^2 + \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle - \tau \cdot (f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)) \\
&- \frac{\tau\mu}{2}\|\boldsymbol{u}_k - \boldsymbol{w}_*\|^2 - \frac{\tau\mu}{2}\|\boldsymbol{w}_k - \boldsymbol{u}_k\|^2
\end{aligned}$$

which gives with $c = 1 - \tau$

$$\begin{aligned}
\frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{w}_*\|^2 &\leq c\frac{\mu}{2}\|\boldsymbol{u}_k - \boldsymbol{w}_*\|^2 + \frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|^2 \\
&\quad + \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle - \tau \cdot (f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)) - \frac{\tau\mu}{2}\|\boldsymbol{w}_k - \boldsymbol{u}_k\|^2.
\end{aligned} \tag{10.4.16}$$

To bound the first term in (10.4.13), we use L-smoothness (10.1.4) and (10.4.11b)

$$f(\boldsymbol{v}_{k+1}) - f(\boldsymbol{w}_k) \leq \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_{k+1} - \boldsymbol{w}_k \rangle + \frac{L}{2}\|\boldsymbol{v}_{k+1} - \boldsymbol{w}_k\|^2 = -\frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2,$$

so that

$$f(\boldsymbol{v}_{k+1}) - f(\boldsymbol{w}_*) - \tau \cdot (f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)) \leq (1-\tau)(f(\boldsymbol{w}_k) - f(\boldsymbol{w}_*)) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2$$
$$= c \cdot (f(\boldsymbol{v}_k) - f(\boldsymbol{w}_*)) + c \cdot (f(\boldsymbol{w}_k) - f(\boldsymbol{v}_k)) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2. \tag{10.4.17}$$

Now, (10.4.16) and (10.4.17) imply

$$e_{k+1} \leq ce_k + c \cdot (f(\boldsymbol{w}_k) - f(\boldsymbol{v}_k)) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2 + \frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|^2$$
$$+ \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle - \frac{\tau\mu}{2}\|\boldsymbol{w}_k - \boldsymbol{u}_k\|^2.$$

Since we wish to bound e_{k+1} by ce_k, we now show that all terms except ce_k on the right-hand side of the inequality above sum up to a non-positive value. By (10.4.11c) and (10.4.15)

$$\frac{\mu}{2}\|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|^2 = \frac{\mu}{2}\|\boldsymbol{v}_k - \boldsymbol{w}_k\|^2 - \tau \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle + \frac{\tau^2}{2\mu}\|\nabla f(\boldsymbol{w}_k)\|^2.$$

Moreover, using μ-strong convexity

$$\langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle$$
$$\leq \tau \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle + (1-\tau)\Big(f(\boldsymbol{v}_k) - f(\boldsymbol{w}_k) - \frac{\mu}{2}\|\boldsymbol{v}_k - \boldsymbol{w}_k\|^2\Big).$$

Thus, we arrive at

$$e_{k+1} \leq ce_k + c \cdot (f(\boldsymbol{w}_k) - f(\boldsymbol{v}_k)) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2 + \frac{\mu}{2}\|\boldsymbol{v}_k - \boldsymbol{w}_k\|^2$$
$$- \tau \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle + \frac{\tau^2}{2\mu}\|\nabla f(\boldsymbol{w}_k)\|^2 + \tau \langle \nabla f(\boldsymbol{w}_k), \boldsymbol{v}_k - \boldsymbol{w}_k \rangle$$
$$+ c \cdot (f(\boldsymbol{v}_k) - f(\boldsymbol{w}_k)) - c\frac{\mu}{2}\|\boldsymbol{v}_k - \boldsymbol{w}_k\|^2 - \frac{\tau\mu}{2}\|\boldsymbol{w}_k - \boldsymbol{u}_k\|^2$$
$$= ce_k + \Big(\frac{\tau^2}{2\mu} - \frac{1}{2L}\Big)\|\nabla f(\boldsymbol{w}_k)\|^2 + \frac{\mu}{2}\Big(\tau - \frac{1}{\tau}\Big)\|\boldsymbol{w}_k - \boldsymbol{v}_k\|^2$$
$$\leq ce_k$$

where we used once more (10.4.15), and the fact that $\tau^2/(2\mu) - 1/(2L) = 0$ and $\tau - 1/\tau \leq 0$ since $\tau = \sqrt{\mu/L} \in (0, 1]$. □

Comparing the result for gradient descent (10.1.16) with NAG (10.4.12), the improvement lies in the convergence rate, which is $1-\kappa^{-1}$ for gradient descent (also see Remark 10.15), and $1-\kappa^{-1/2}$ for NAG, where $\kappa = L/\mu$. In contrast to gradient descent, for NAG the convergence depends only on the *square root* of the condition number κ. For ill-conditioned problems where κ is large, we therefore expect much better performance for accelerated methods.

Finally, we mention that NAG also achieves faster convergence in the case of L-smooth and convex objective functions. While the error decays like $O(k^{-1})$ for gradient descent, see Theorem 10.11, for NAG one obtains convergence $O(k^{-2})$, see [160, 158, 241].

10.5 Other methods

In recent years, a multitude of first order (gradient descent) methods has been proposed and studied for the training of neural networks. They typically employ (a subset) of the three critical strategies: mini-batches, acceleration, and adaptive step sizes. The concept of mini-batches and acceleration have been covered in the previous sections, and we will touch upon adaptive learning rates in the present one. Specifically, we present three algorithms—AdaGrad, RMSProp, and Adam—which have been among the most influential in the field, and serve to explore the main ideas. An intuitive overview of first order methods can also be found in [194], which discusses additional variants that are omitted here. Moreover, in practice, various other techniques and heuristics such as batch normalization, gradient clipping, data augmentation, regularization and dropout, early stopping, specific weight initializations etc. are used. We do not discuss them here, and refer to [22] or to [67, Chapter 11] for a practitioners guide.

After initializing $\boldsymbol{m}_0 = \mathbf{0} \in \mathbb{R}^n$, $\boldsymbol{v}_0 = \mathbf{0} \in \mathbb{R}^n$, and $\boldsymbol{w}_0 \in \mathbb{R}^n$, all methods discussed below are special cases of the update

$$\boldsymbol{m}_{k+1} = \beta_1 \boldsymbol{m}_k + \beta_2 \nabla f(\boldsymbol{w}_k) \tag{10.5.1a}$$

$$\boldsymbol{v}_{k+1} = \gamma_1 \boldsymbol{v}_k + \gamma_2 \nabla f(\boldsymbol{w}_k) \odot \nabla f(\boldsymbol{w}_k) \tag{10.5.1b}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{m}_{k+1} \oslash \sqrt{\boldsymbol{v}_{k+1} + \varepsilon} \tag{10.5.1c}$$

for $k \in \mathbb{N}_0$, and certain hyperparameters α_k, β_1, β_2, γ_1, γ_2, and ε. Here $\odot$ and $\oslash$ denote the componentwise multiplication and division, respectively, and $\sqrt{\boldsymbol{v}_{k+1} + \varepsilon}$ is understood as the vector $(\sqrt{v_{k+1,i} + \varepsilon})_i$. We will give some default values for those hyperparameters in the following, but mention that careful problem dependent tuning can enhance performance. Equation (10.5.1a) corresponds to heavy ball momentum if $\beta_1 > 0$. If $\beta_1 = 0$, then $\boldsymbol{m}_{k+1}$ is simply a multiple of the current gradient. Equation (10.5.1b) defines a weight vector $\boldsymbol{v}_{k+1}$ that is used to set the componentwise learning rate in the update of the parameter in (10.5.1c). These type of methods are often applied using mini-batches, see Section 10.2. For simplicity we present them with the full gradients.

10.5.1 AdaGrad

In Section 10.2 we argued, that for stochastic methods the learning rate should decrease in order to get convergence. The choice of how to decrease the learning rate can have significant impact in practice. AdaGrad [57], which stands for adaptive gradient algorithm, provides a method to dynamically adjust learning rates during optimization. Moreover, it does so by using individual learning rates for each component.

AdaGrad correspond to (10.5.1) with

$$\beta_1 = 0, \qquad \gamma_1 = \beta_2 = \gamma_2 = 1, \qquad \alpha_k = \alpha \quad \text{for all } k \in \mathbb{N}_0.$$

This leaves the hyperparameters $\varepsilon > 0$ and $\alpha > 0$. The constant $\varepsilon > 0$ is chosen small but positive to avoid division by zero in (10.5.1c). Possible default values are $\alpha = 0.01$ and $\varepsilon = 10^{-8}$. The AdaGrad update then reads

$$\begin{aligned} \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \nabla f(\boldsymbol{w}_k) \odot \nabla f(\boldsymbol{w}_k) \\ \boldsymbol{w}_{k+1} &= \boldsymbol{w}_k - \alpha \nabla f(\boldsymbol{w}_k) \oslash \sqrt{\boldsymbol{v}_{k+1} + \varepsilon}. \end{aligned}$$

Due to

$$\boldsymbol{v}_{k+1} = \sum_{j=0}^{k} \nabla f(\boldsymbol{w}_j) \odot \nabla f(\boldsymbol{w}_j), \tag{10.5.2}$$

the algorithm scales the gradient $\nabla f(\boldsymbol{w}_k)$ in the update component-wise by the inverse square root of the sum over all past squared gradients plus ε. Note that the scaling factor $(v_{k+1,i} + \varepsilon)^{-1/2}$ for component i will be large, if the previous gradients for that component were small, and vice versa. In the words of the authors of [57]: "our procedures give frequently occurring features very low learning rates and infrequent features high learning rates."

Remark 10.26. A benefit of the componentwise scaling can be observed for the ill-conditioned objective function in (10.4.1). Since in this case $\nabla f(\boldsymbol{w}_j) = (\lambda_1 w_{j,1}, \lambda_2 w_{j,2})^\top$ for each $j = 1, \dots, k$, setting $\varepsilon = 0$ AdaGrad performs the update

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha \begin{pmatrix} w_{k,1}(\sum_{j=0}^{k} w_{j,1}^2)^{-1/2} \\ w_{k,2}(\sum_{j=0}^{k} w_{j,1}^2)^{-1/2} \end{pmatrix}.$$

Note how the λ_1 and λ_2 factors in the update have vanished due to the division by $\sqrt{\boldsymbol{v}_{k+1}}$. This makes the method invariant to a componentwise rescaling of the gradient, and results in a more direct path towards the minimizer.

10.5.2 RMSProp

The sum of past squared gradients can increase rapidly, leading to a significant reduction in learning rates when training neural networks with AdaGrad. This often results in slow convergence, see for example [242]. RMSProp [90] seeks to rectify this by adjusting the learning rates using an exponentially weighted average of past gradients.

RMSProp corresponds to (10.5.1) with

$$\beta_1 = 0, \qquad \beta_2 = 1, \qquad \gamma_2 = 1 - \gamma_1 \in (0, 1), \qquad \alpha_k = \alpha \quad \text{for all } k \in \mathbb{N}_0,$$

effectively leaving the hyperparameters $\varepsilon > 0$, $\gamma_1 \in (0, 1)$ and $\alpha > 0$. Typically, recommended default values are $\varepsilon = 10^{-8}$, $\alpha = 0.01$ and $\gamma_1 = 0.9$. The algorithm is given through

$$\boldsymbol{v}_{k+1} = \gamma_1 \boldsymbol{v}_k + (1 - \gamma_1) \nabla f(\boldsymbol{w}_k) \odot \nabla f(\boldsymbol{w}_k) \tag{10.5.3a}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha \nabla f(\boldsymbol{w}_k) \oslash \sqrt{\boldsymbol{v}_{k+1} + \varepsilon}. \tag{10.5.3b}$$

Note that

$$\boldsymbol{v}_{k+1} = (1 - \gamma_1) \sum_{j=0}^{k} \gamma_1^j \nabla f(\boldsymbol{w}_{k-j}) \odot \nabla f(\boldsymbol{w}_{k-j}),$$

so that, contrary to AdaGrad (10.5.2), the influence of gradient $\nabla f(\boldsymbol{w}_{k-j})$ on the weight $\boldsymbol{v}_{k+1}$ decays exponentially in j.

10.5.3 Adam

Adam [116], short for adaptive moment estimation, combines adaptive learning rates based on exponentially weighted averages as in RMSProp, with heavy ball momentum. Contrary to AdaGrad an RMSProp it thus uses a value $\beta_1 > 0$.

More precisely, Adam corresponds to (10.5.1) with

$$\beta_2 = 1 - \beta_1 \in (0,1), \qquad \gamma_2 = 1 - \gamma_1 \in (0,1), \qquad \alpha_k = \alpha \frac{\sqrt{1-\gamma_1^{k+1}}}{1-\beta_1^{k+1}}$$

for all $k \in \mathbb{N}_0$, for some $\alpha > 0$. The default values for the remaining parameters recommended in [116] are $\varepsilon = 10^{-8}$, $\alpha = 0.001$, $\beta_1 = 0.9$ and $\gamma_1 = 0.999$. The update can be formulated as

$$\boldsymbol{m}_{k+1} = \beta_1 \boldsymbol{m}_k + (1-\beta_1)\nabla f(\boldsymbol{w}_k), \qquad \hat{\boldsymbol{m}}_{k+1} = \frac{\boldsymbol{m}_{k+1}}{1-\beta_1^{k+1}} \tag{10.5.4a}$$

$$\boldsymbol{v}_{k+1} = \gamma_1 \boldsymbol{v}_k + (1-\gamma_1)\nabla f(\boldsymbol{w}_k) \odot \nabla f(\boldsymbol{w}_k), \qquad \hat{\boldsymbol{v}}_{k+1} = \frac{\boldsymbol{v}_{k+1}}{1-\gamma_1^{k+1}} \tag{10.5.4b}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha \hat{\boldsymbol{m}}_{k+1} \oslash \sqrt{\hat{\boldsymbol{v}}_{k+1} + \varepsilon}. \tag{10.5.4c}$$

Note that $\boldsymbol{m}_{k+1}$ equals

$$\boldsymbol{m}_{k+1} = (1-\beta_1)\sum_{j=0}^{k} \beta_1^j \nabla f(\boldsymbol{w}_{k-j})$$

and thus correspond to heavy ball style momentum with momentum parameter $\beta = \beta_1$, see (10.4.6). The normalized version $\hat{\boldsymbol{m}}_{k+1}$ is introduced to account for the bias towards $\mathbf{0}$, stemming from the initialization $\boldsymbol{m}_0 = \mathbf{0}$. The weight-vector $\boldsymbol{v}_{k+1}$ in (10.5.4b) is analogous to the exponentially weighted average of RMSProp in (10.5.3a), and the normalization again serves to counter the bias from $\boldsymbol{v}_0 = \mathbf{0}$.

It should be noted that there exist examples of convex functions for which Adam does *not converge to a minimizer*, see [190]. The authors of [190] propose a modification termed AMSGrad, which avoids this issue and their analysis also applies to RMSProp. Nonetheless, Adam remains a highly popular and successful algorithm for the training of neural networks. We also mention that the proof of convergence in the stochastic setting requires k-dependent decreasing learning rates such as $\alpha = O(k^{-1/2})$ in (10.5.3b) and (10.5.4c).

Bibliography and further reading

Section 10.1 on gradient descent is based on standard textbooks such as [20, 25, 163] and especially [159]. These are also good references for further reading on convex optimization. In particular Theorem 10.11 and the Lemmas leading up to it closely follow Nesterov's arguments in [159]. Convergence proofs under the PL inequality can be found in [114]. Stochastic gradient descent discussed in Section 10.2 originally dates back to Robbins and Monro [191]. The first non-asymptotic convergence analysis for strongly convex objective functions was given in [154]. The proof presented here is similar to [76] and in particular uses their choice of step size. A good overview of proofs for (stochastic) gradient descent algorithms together with detailed references can be found in [65],

and for a textbook specifically on stochastic optimization also see [126]. The backpropagation algorithm discussed in Section 10.3 was popularized by Rumelhart, Hinton and Williams [197]; for further details on the historical developement we refer to [202, Section 5.5], and for a more in-depth discussion of the algorithm, see for instance [84]. The heavy ball method in Section 10.4 goes back to Polyak [180]. To motivate the algorithm we proceed similar as in [70, 181, 183]. For the analysis of Nesterov acceleration [160], we follow the Lyapunov type proofs given in [231, 241]. Finally, for Section 10.5 on other algorithms, we refer to the original works that introduced AdaGrad [57], RMSProp [90] and Adam [116]. A good overview of gradient descent methods popular for deep learning can be found in [194]. Regarding the analysis of RMSProp and Adam, we refer to [190] which gave an example of a convex function for which Adam does not converge, and provide a provably convergent modification of the algorithm. Convergence proofs (for variations of) AdaGrad and Adam can also be found in [49].

For a general discussion and analysis of optimization algorithms in machine learning see [23]. Details on implementations in Python can for example be found in [67], and for recommendations and tricks regarding the implementation we also refer to [22, 129].

Exercises

Exercise 10.27. Let $f \in C^1(\mathbb{R}^n)$. Show that f is convex in the sense of Definition 10.8 if and only if

$$f(\boldsymbol{w}) + \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w} \rangle \leq f(\boldsymbol{v}) \qquad \text{for all } \boldsymbol{w}, \boldsymbol{v} \in \mathbb{R}^n.$$

Exercise 10.28. Find a function $f : \mathbb{R} \to \mathbb{R}$ that is L-smooth, satisfies the PL-inequality (10.1.19) for some $\mu > 0$, has a unique minimizer $w_* \in \mathbb{R}$, but is not convex and thus also not strongly convex.

Exercise 10.29. Prove Theorem 10.17, i.e. show that L-smoothness and the PL-inequality (10.1.19) yield linear convergence of $f(\boldsymbol{w}_k) \to f(\boldsymbol{w}_*)$ as $k \to \infty$.

Definition 10.30. For convex $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g} \in \mathbb{R}^n$ is called a **subgradient** (or subdifferential) of f at $\boldsymbol{v}$ if and only if

$$f(\boldsymbol{w}) \geq f(\boldsymbol{v}) + \langle \boldsymbol{g}, \boldsymbol{w} - \boldsymbol{v} \rangle \qquad \text{for all } \boldsymbol{w} \in \mathbb{R}^n. \tag{10.5.5}$$

The set of all subgradients of f at $\boldsymbol{v}$ is denoted by $\partial f(\boldsymbol{v})$.

A subgradient always exists, i.e. $\partial f(\boldsymbol{v})$ is necessarily nonempty. This statement is also known under the name "Hyperplane separation theorem". Subgradients generalize the notion of gradients for convex functions, since for any convex continuously differentiable f, (10.5.5) is satisfied with $\boldsymbol{g} = \nabla f(\boldsymbol{v})$.

Exercise 10.31. Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex and $\mathrm{Lip}(f) \leq L$. Show that for any $\boldsymbol{g} \in \partial f(\boldsymbol{v})$ holds $\|\boldsymbol{g}\| \leq L$.

Exercise 10.32. Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex, $\mathrm{Lip}(f) \leq L$ and suppose that $\boldsymbol{w}_*$ is a minimizer of f. Fix $\boldsymbol{w}_0 \in \mathbb{R}^d$, and for $k \in \mathbb{N}_0$ define the **subgradient descent** update

$$\boldsymbol{w}_{k+1} := \boldsymbol{w}_k - h_k \boldsymbol{g}_k,$$

where $\boldsymbol{g}_k$ is an arbitrary fixed element of $\partial f(\boldsymbol{w}_k)$. Show that

$$\min_{i \leq k} f(\boldsymbol{w}_i) - f(\boldsymbol{w}_*) \leq \frac{\|\boldsymbol{w}_0 - \boldsymbol{w}_*\|^2 + L^2 \sum_{i=1}^{k} h_i^2}{2 \sum_{i=1}^{k} h_i}.$$

Hint: Start by recursively expanding $\|\boldsymbol{w}_k - \boldsymbol{w}_*\|^2 = \cdots$, and then apply the property of the subgradient.

Exercise 10.33. Consider the setting of Exercise 10.32. Determine step sizes $h_1, \ldots, h_k$ (which may depend on k, i.e. $h_{k,1}, \ldots, h_{k,k}$) such that for any arbitrarily small $\delta > 0$

$$\min_{i \leq k} f(\boldsymbol{w}_i) - f(\boldsymbol{w}_*) = O(k^{-1/2+\delta}) \qquad \text{as } k \to \infty.$$

Exercise 10.34. Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be symmetric positive semidefinite, $\boldsymbol{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Denote the eigenvalues of $\boldsymbol{A}$ by $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$. Show that the objective function

$$f(\boldsymbol{w}) := \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}^\top \boldsymbol{w} + c \tag{10.5.6}$$

is convex and λ_1-smooth. Moreover, if $\lambda_n > 0$, then f is λ_n-strongly convex. Show that these values are optimal in the sense that f is neither L-smooth nor μ-strongly convex if $L < \lambda_1$ and $\mu > \lambda_n$.

Hint: Note that L-smoothness and μ-strong convexity are invariant under shifts and the addition of constants. That is, for every $\alpha \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}^n$, $\tilde{f}(\boldsymbol{w}) := \alpha + f(\boldsymbol{w}+\boldsymbol{\beta})$ is L-smooth or μ-strongly convex if and only if f is. It thus suffices to consider $\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w}/2$.

Exercise 10.35. Let f be as in Exercise 10.34. Show that gradient descent converges for arbitrary initialization $\boldsymbol{w}_0 \in \mathbb{R}^n$, if and only if

$$\max_{j=1,\ldots,n} |1 - h\lambda_j| < 1.$$

Show that $\operatorname{argmin}_{h>0} \max_{j=1,\ldots,n} |1 - h\lambda_j| = 2/(\lambda_1 + \lambda_n)$ and conclude that the convergence will be slow if f is ill-conditioned, i.e. if $\lambda_1/\lambda_n \gg 1$.

Hint: Assume first that $\boldsymbol{b} = \boldsymbol{0} \in \mathbb{R}^n$ and $c = 0 \in \mathbb{R}$ in (10.5.6), and use the singular value decomposition $\boldsymbol{A} = \boldsymbol{U}^\top \operatorname{diag}(\lambda_1, \ldots, \lambda_n)\boldsymbol{U}$.

Exercise 10.36. Show that (10.4.10) can equivalently be written as (10.4.11) with $\tau = \sqrt{\mu/L}$, $\alpha = 1/L$, $\beta = (1-\tau)/(1+\tau)$ and the initialization $\boldsymbol{u}_0 = ((1+\tau)\boldsymbol{w}_0 - \boldsymbol{v}_0)/\tau$.

Chapter 11

Wide neural networks and the neural tangent kernel

In this chapter we explore the dynamics of training neural networks of large width. Throughout we focus on the situation where we have data pairs

$$(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R} \qquad i \in \{1, \dots, m\}, \tag{11.0.1a}$$

and wish to train a neural network $\Phi(\boldsymbol{x}, \boldsymbol{w})$ depending on the input $\boldsymbol{x} \in \mathbb{R}^d$ and the parameters $\boldsymbol{w} \in \mathbb{R}^n$, by minimizing the square loss objective defined as

$$f(\boldsymbol{w}) := \sum_{i=1}^{m} (\Phi(\boldsymbol{x}_i, \boldsymbol{w}) - y_i)^2, \tag{11.0.1b}$$

which is a multiple of the empirical risk $\widehat{\mathcal{R}}_S(\Phi)$ in (1.2.3) for the sample $S = (\boldsymbol{x}_i, y_i)_{i=1}^m$ and the square-loss. We exclusively focus on gradient descent with a constant step size h, which yields a sequence of parameters $(\boldsymbol{w}_k)_{k \in \mathbb{N}}$. We aim to understand the evolution of $\Phi(\boldsymbol{x}, \boldsymbol{w}_k)$ as k progresses. For linear mappings $\boldsymbol{w} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w})$, the objective function (11.0.1b) is convex. As established in the previous chapter, gradient descent then finds a global minimizer. For typical neural network architectures, $\boldsymbol{w} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w})$ is not linear, and such a statement is in general not true.

Recent research has shown that neural network behavior tends to linearize in the parameters as network width increases [106]. This allows to transfer some of the results and techniques from the linear case to the training of neural networks. We start this chapter in Sections 11.1 and 11.2 by recalling (kernel) least-squares methods, which describe linear (in $\boldsymbol{w}$) models. Following [131], the subsequent sections explore why in the infinite width limit neural networks exhibit linear-like behavior. In Section 11.5.2 we formally introduce the linearization of $\boldsymbol{w} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w})$. Section 11.4 presents an abstract result showing convergence of gradient descent, under the condition that Φ does not deviate too much from its linearization. In Sections 11.5 and 11.6, we then detail the implications for wide neural networks for two (slightly) different architectures. In particular, we will prove that gradient descent can find global minimizers when applied to (11.0.1b) for networks of very large width. We emphasize that this analysis treats the case of strong overparametrization, specifically the case of increasing the network width while keeping the number of data points m fixed.

11.1 Linear least-squares

Arguably one of the simplest machine learning algorithms is linear least-squares regression. Given data (11.0.1a), linear regression tries to fit a linear function $\Phi(\boldsymbol{x}, \boldsymbol{w}) := \boldsymbol{x}^\top \boldsymbol{w}$ in terms of $\boldsymbol{w}$ by minimizing $f(\boldsymbol{w})$ in (11.0.1b). With

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_m^\top \end{pmatrix} \in \mathbb{R}^{m \times d} \qquad \text{and} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \in \mathbb{R}^m \tag{11.1.1}$$

it holds

$$f(\boldsymbol{w}) = \|\boldsymbol{A}\boldsymbol{w} - \boldsymbol{y}\|^2. \tag{11.1.2}$$

Remark 11.1. More generally, the ansatz $\Phi(\boldsymbol{x}, (\boldsymbol{w}, b)) := \boldsymbol{w}^\top \boldsymbol{x} + b$ corresponds to

$$\Phi(\boldsymbol{x}, (\boldsymbol{w}, b)) = (1, \boldsymbol{x}^\top) \begin{pmatrix} b \\ \boldsymbol{w} \end{pmatrix}.$$

Therefore, additionally allowing for a bias can be treated analogously.

The model $\Phi(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{x}^\top \boldsymbol{w}$ is linear in both $\boldsymbol{x}$ and $\boldsymbol{w}$. In particular, $\boldsymbol{w} \mapsto f(\boldsymbol{w})$ is a convex function by Exercise 10.34, and we may apply the convergence results of Chapter 10 when using gradient based algorithms. If $\boldsymbol{A}$ is invertible, then f has a unique minimizer given by $\boldsymbol{w}_* = \boldsymbol{A}^{-1}\boldsymbol{y}$. If $\operatorname{rank}(\boldsymbol{A}) = d$, then f is strongly convex by Exercise 10.34, and there still exists a unique minimizer. If however $\operatorname{rank}(\boldsymbol{A}) < d$, then $\ker(\boldsymbol{A}) \neq \{\boldsymbol{0}\}$ and there exist infinitely many minimizers of f. To ensure uniqueness, we look for the **minimum norm solution** (or minimum 2-norm solution)

$$\boldsymbol{w}_* := \operatorname{argmin}_{\{\boldsymbol{w} \in \mathbb{R}^d \,|\, f(\boldsymbol{w}) \leq f(\boldsymbol{v}) \ \forall \boldsymbol{v} \in \mathbb{R}^d\}} \|\boldsymbol{w}\|. \tag{11.1.3}$$

The following proposition establishes the uniqueness of $\boldsymbol{w}_*$ and demonstrates that it can be represented as a superposition of the $(\boldsymbol{x}_i)_{i=1}^m$.

Proposition 11.2. *Let $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{y} \in \mathbb{R}^m$ be as in* (11.1.1)*. There exists a unique minimum* 2*-norm solution of* (11.1.2)*. Denoting $\tilde{H} := \operatorname{span}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subseteq \mathbb{R}^d$, it is the unique element*

$$\boldsymbol{w}_* = \operatorname{argmin}_{\tilde{\boldsymbol{w}} \in \tilde{H}} f(\tilde{\boldsymbol{w}}) \in \tilde{H}. \tag{11.1.4}$$

Proof. We start with existence and uniqueness. Let $C \subseteq \mathbb{R}^m$ be the space spanned by the columns of $\boldsymbol{A}$. Then C is closed and convex, and therefore $\boldsymbol{y}_* = \operatorname{argmin}_{\tilde{\boldsymbol{y}} \in C} \|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|$ exists and is unique (this is a fundamental property of Hilbert spaces, see Theorem B.14). In particular, the set $M = \{\boldsymbol{w} \in \mathbb{R}^d \,|\, \boldsymbol{A}\boldsymbol{w} = \boldsymbol{y}_*\} \subseteq \mathbb{R}^d$ of minimizers of f is not empty. Clearly M is also closed and convex. By the same argument as before, $\boldsymbol{w}_* = \operatorname{argmin}_{\boldsymbol{w}_* \in M} \|\boldsymbol{w}_*\|$ exists and is unique.

It remains to show (11.1.4). Denote by $\boldsymbol{w}_*$ the minimum norm solution and decompose $\boldsymbol{w}_* = \tilde{\boldsymbol{w}} + \hat{\boldsymbol{w}}$ with $\tilde{\boldsymbol{w}} \in \tilde{H}$ and $\hat{\boldsymbol{w}} \in \tilde{H}^\perp$. We have $\boldsymbol{A}\boldsymbol{w}_* = \boldsymbol{A}\tilde{\boldsymbol{w}}$ and $\|\boldsymbol{w}_*\|^2 = \|\tilde{\boldsymbol{w}}\|^2 + \|\hat{\boldsymbol{w}}\|^2$. Since $\boldsymbol{w}_*$ is the minimal norm solution it must hold $\hat{\boldsymbol{w}} = \boldsymbol{0}$. Thus $\boldsymbol{w}_* \in \tilde{H}$. Finally assume there exists a minimizer $\boldsymbol{v}$ of f in $\tilde{H}$ different from $\boldsymbol{w}_*$. Then $\boldsymbol{0} \neq \boldsymbol{w}_* - \boldsymbol{v} \in \tilde{H}$, and since $\tilde{H}$ is spanned by the rows of $\boldsymbol{A}$ we have $\boldsymbol{A}(\boldsymbol{w}_* - \boldsymbol{v}) \neq \boldsymbol{0}$. Thus $\boldsymbol{y}_* = \boldsymbol{A}\boldsymbol{w}_* \neq \boldsymbol{A}\boldsymbol{v}$, which contradicts that $\boldsymbol{v}$ minimizes f. □

The condition of minimizing the 2-norm is a form of regularization. Interestingly, gradient descent converges to the minimum norm solution for the quadratic objective (11.1.2), as long as $\boldsymbol{w}_0$ is initialized within $\tilde{H} = \operatorname{span}\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$ (e.g. $\boldsymbol{w}_0 = 0$). Therefore, it does not find an "arbitrary" minimizer but implicitly regularizes the problem in this sense. In the following $s_{\max}(\boldsymbol{A})$ denotes the maximal singular value of $\boldsymbol{A}$.

Theorem 11.3. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ be as in (11.1.1), let $\boldsymbol{w}_0 = \tilde{\boldsymbol{w}}_0 + \hat{\boldsymbol{w}}_0$ where $\tilde{\boldsymbol{w}}_0 \in \tilde{H}$ and $\hat{\boldsymbol{w}}_0 \in \tilde{H}^\perp$. Fix $h \in (0, 1/(2s_{\max}(\boldsymbol{A})^2))$ and set*

$$\boldsymbol{w}_{k+1} := \boldsymbol{w}_k - h\nabla f(\boldsymbol{w}_k) \qquad \text{for all } k \in \mathbb{N} \tag{11.1.5}$$

with f in (11.1.2). Then

$$\lim_{k\to\infty} \boldsymbol{w}_k = \boldsymbol{w}_* + \hat{\boldsymbol{w}}_0.$$

We sketch the argument in case $\boldsymbol{w}_0 \in \tilde{H}$, and leave the full proof to the reader, see Exercise 11.32. Note that $\tilde{H}$ is the space spanned by the rows of $\boldsymbol{A}$ (or the columns of $\boldsymbol{A}^\top$). The gradient of the objective function equals

$$\nabla f(\boldsymbol{w}) = 2\boldsymbol{A}^\top(\boldsymbol{A}\boldsymbol{w} - \boldsymbol{y}).$$

Therefore, if $\boldsymbol{w}_0 \in \tilde{H}$, then the iterates of gradient descent never leave the subspace $\tilde{H}$. By Exercise 10.34 and Theorem 10.11, for small enough step size, it holds $f(\boldsymbol{w}_k) \to 0$. By Proposition 11.2 there only exists one minimizer in $\tilde{H}$, corresponding to the minimum norm solution. Thus $\boldsymbol{w}_k$ converges to the minimal norm solution.

11.2 Kernel least-squares

Let again $(\boldsymbol{x}_j, y_j) \in \mathbb{R}^d \times \mathbb{R}$, $j = 1, \dots, m$. In many applications linear models are too simplistic, and are not able to capture the true relation between $\boldsymbol{x}$ and y. Kernel methods allow to overcome this problem by introducing nonlinearity in $\boldsymbol{x}$, but retaining linearity in the parameter $\boldsymbol{w}$.

Let H be a Hilbert space with inner product $\langle\cdot,\cdot\rangle_H$, that is also referred to as the **feature space**. For a (typically nonlinear) **feature map** $\phi : \mathbb{R}^d \to H$, consider the model

$$\Phi(\boldsymbol{x}, \boldsymbol{w}) = \langle \phi(\boldsymbol{x}), \boldsymbol{w}\rangle_H \tag{11.2.1}$$

with $\boldsymbol{w} \in H$. If $H = \mathbb{R}^n$, the components of ϕ are referred to as features. With the objective function

$$f(\boldsymbol{w}) := \sum_{j=1}^{m} \big(\langle \phi(\boldsymbol{x}_j), \boldsymbol{w}\rangle_H - y_j\big)^2 \qquad \boldsymbol{w} \in H, \tag{11.2.2}$$

we wish to determine a minimizer of f. To ensure uniqueness and regularize the problem, we again consider the minimum H-norm solution

$$\boldsymbol{w}_* := \operatorname{argmin}_{\{\boldsymbol{w}\in H \,|\, f(\boldsymbol{w})\le f(\boldsymbol{v})\ \forall \boldsymbol{v}\in H\}} \|\boldsymbol{w}\|_H.$$

As we will see below, $\boldsymbol{w}_*$ is well-defined. We will call $\Phi(\boldsymbol{x}, \boldsymbol{w}_*) = \langle \phi(\boldsymbol{x}), \boldsymbol{w}_*\rangle_H$ the **kernel least squares estimator**. The nonlinearity of the feature map allows for more expressive models $\boldsymbol{x} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w})$ capable of capturing more complicated structures beyond linearity in the data.

Remark 11.4 (Gradient descent). Let $H = \mathbb{R}^n$ be equipped with the Euclidean inner product. Consider the sequence $(\boldsymbol{w}_k)_{k\in\mathbb{N}_0} \subseteq \mathbb{R}^n$ generated by gradient descent to minimize (11.2.2). Assuming sufficiently small step size, by Theorem 11.3 for $\boldsymbol{x} \in \mathbb{R}^d$

$$\lim_{k\to\infty} \Phi(\boldsymbol{x}, \boldsymbol{w}_k) = \langle \phi(\boldsymbol{x}), \boldsymbol{w}_* \rangle + \langle \phi(\boldsymbol{x}), \hat{\boldsymbol{w}}_0 \rangle \,. \tag{11.2.3}$$

Here, $\hat{\boldsymbol{w}}_0 \in \mathbb{R}^n$ denotes the orthogonal projection of $\boldsymbol{w}_0 \in \mathbb{R}^n$ onto $\tilde{H}^\perp$ where $\tilde{H} := \operatorname{span}\{\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_m)\}$. Gradient descent thus yields the kernel least squares estimator plus $\langle \phi(\boldsymbol{x}), \hat{\boldsymbol{w}}_0 \rangle$. Notably, on the set

$$\{\boldsymbol{x} \in \mathbb{R}^d \,|\, \phi(\boldsymbol{x}) \in \operatorname{span}\{\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_m)\}\}, \tag{11.2.4}$$

(11.2.3) thus coincides with the kernel least squares estimator independent of the initialization $\boldsymbol{w}_0$.

11.2.1 Examples

To motivate the concept of feature maps consider the following example from [155].

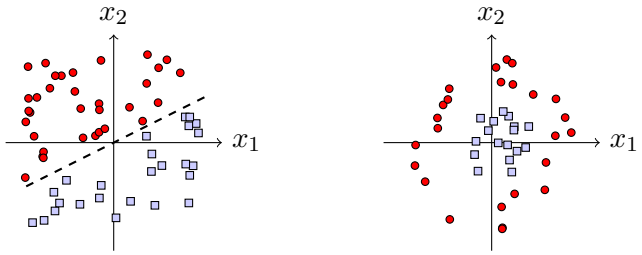
Example 11.5. Let $\boldsymbol{x}_i \in \mathbb{R}^2$ with associated labels $y_i \in \{-1, 1\}$ for $i = 1, \dots, m$. The goal is to find some model $\Phi(\cdot, \boldsymbol{w}) : \mathbb{R}^2 \to \mathbb{R}$, for which

$$\operatorname{sign}(\Phi(\boldsymbol{x}, \boldsymbol{w})) \tag{11.2.5}$$

predicts the label y of $\boldsymbol{x}$. For a linear (in $\boldsymbol{x}$) model

$$\Phi(\boldsymbol{x}, (\boldsymbol{w}, b)) = \boldsymbol{x}^\top \boldsymbol{w} + b,$$

the decision boundary of (11.2.5) equals $\{\boldsymbol{x} \in \mathbb{R}^2 \,|\, \boldsymbol{x}^\top \boldsymbol{w} + b = 0\}$ in $\mathbb{R}^2$. Hence, by adjusting $\boldsymbol{w}$ and b, (11.2.5) can separate data by affine hyperplanes in $\mathbb{R}^2$. Consider two datasets represented by light blue squares for $+1$ and red circles for -1 labels:



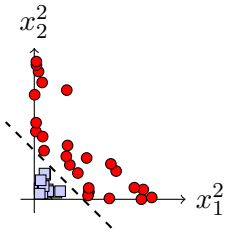
dataset 1 dataset 2

The first dataset is separable by an affine hyperplane as depicted by the dashed line. Thus a linear model is capable of correctly classifying all datapoints. For the second dataset this is not possible.

To enhance model expressivity, introduce a feature map $\phi : \mathbb{R}^2 \to \mathbb{R}^6$ via

$$\phi(\boldsymbol{x}) = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)^\top \in \mathbb{R}^6 \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^2. \tag{11.2.6}$$

For $\boldsymbol{w} \in \mathbb{R}^6$, this allows $\Phi(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$ to represent arbitrary polynomials of degree 2. With this kernel approach, the decision boundary of (11.2.5) becomes the set of all hyperplanes *in the feature space* passing through $\mathbf{0} \in \mathbb{R}^6$. Visualizing the last two features of the second dataset, we obtain



features 5 and 6 of dataset 2

Note how in the feature space $\mathbb{R}^6$, the datapoints are again separated by such a hyperplane. Thus, with the feature map in (11.2.6), the predictor (11.2.5) can perfectly classify all points also for the second dataset.

In the above example we chose the feature space $H = \mathbb{R}^6$. It is also possible to work with infinite dimensional feature spaces as the next example demonstrates.

Example 11.6. Let $H = \ell^2(\mathbb{N})$ be the space of square summable sequences and $\phi : \mathbb{R}^d \to \ell^2(\mathbb{N})$ some map. Fitting the corresponding model

$$\Phi(\boldsymbol{x}, \boldsymbol{w}) = \langle \phi(\boldsymbol{x}), \boldsymbol{w} \rangle_{\ell^2} = \sum_{i \in \mathbb{N}} \phi_i(\boldsymbol{x}) w_i$$

to data $(\boldsymbol{x}_i, y_i)_{i=1}^m$ requires to minimize

$$f(\boldsymbol{w}) = \sum_{j=1}^m \left(\Big(\sum_{i \in \mathbb{N}} \phi_i(\boldsymbol{x}_j) w_i \Big) - y_j \right)^2 \qquad \boldsymbol{w} \in \ell^2(\mathbb{N}).$$

Hence we have to determine an *infinite sequence* of parameters $(w_i)_{i \in \mathbb{N}}$.

11.2.2 Kernel trick

At first glance, computing a (minimal H-norm) minimizer $\boldsymbol{w}$ in the possibly infinite-dimensional Hilbert space H seems infeasible. The so-called *kernel trick* allows to do this computation. To explain it, we first revisit the foundational representer theorem.

Theorem 11.7 (Representer theorem)**.** *There is a unique minimum H-norm solution $\boldsymbol{w}_* \in H$ of (11.2.2). With $\tilde{H} := \operatorname{span}\{\phi(\boldsymbol{x}_1), \dots, \phi(\boldsymbol{x}_m)\}$ it equals the unique element*

$$\boldsymbol{w}_* = \operatorname{argmin}_{\tilde{\boldsymbol{w}} \in \tilde{H}} f(\tilde{\boldsymbol{w}}) \in \tilde{H}. \tag{11.2.7}$$

Proof. Let $\tilde{\boldsymbol{w}}_1, \dots, \tilde{\boldsymbol{w}}_n$ be a basis of $\tilde{H}$. If $\tilde{H} = \{0\}$ the statement is trivial, so we assume $1 \le n \le m$. Let $\boldsymbol{A} = (\langle \phi(\boldsymbol{x}_i), \tilde{\boldsymbol{w}}_j \rangle)_{ij} \in \mathbb{R}^{m \times n}$. Every $\tilde{\boldsymbol{w}} \in \tilde{H}$ has a unique representation $\tilde{\boldsymbol{w}} = \sum_{j=1}^n \alpha_j \tilde{\boldsymbol{w}}_j$ for some $\boldsymbol{\alpha} \in \mathbb{R}^n$. With this ansatz

$$f(\tilde{\boldsymbol{w}}) = \sum_{i=1}^m (\langle \phi(\boldsymbol{x}_i), \tilde{\boldsymbol{w}} \rangle - y_i)^2 = \sum_{i=1}^m \Big(\sum_{j=1}^n \langle \phi(\boldsymbol{x}_i), \tilde{\boldsymbol{w}}_j \rangle \, \alpha_j - y_i \Big)^2 = \|\boldsymbol{A}\boldsymbol{\alpha} - \boldsymbol{y}\|^2. \tag{11.2.8}$$

Note that $\boldsymbol{A} : \mathbb{R}^n \to \mathbb{R}^m$ is injective since for every $\boldsymbol{\alpha} \in \mathbb{R}^n \backslash \{0\}$ holds $\sum_{j=1}^n \alpha_j \tilde{\boldsymbol{w}}_j \in \tilde{H} \setminus \{0\}$ and hence $\boldsymbol{A}\alpha = (\left\langle \phi(\boldsymbol{x}_i), \sum_{j=1}^n \alpha_j \tilde{\boldsymbol{w}}_j \right\rangle)_{i=1}^m \neq 0$. Therefore, there exists a unique minimizer $\boldsymbol{\alpha} \in \mathbb{R}^n$ of the right-hand side of (11.2.8), and thus there exists a unique minimizer $\boldsymbol{w}_* \in \tilde{H}$ in (11.2.7).

For arbitrary $\boldsymbol{w} \in H$ we wish to show $f(\boldsymbol{w}) \geq f(\boldsymbol{w}_*)$, so that $\boldsymbol{w}_*$ minimizes f in H. Decompose $\boldsymbol{w} = \tilde{\boldsymbol{w}} + \hat{\boldsymbol{w}}$ with $\tilde{\boldsymbol{w}} \in \tilde{H}$ and $\hat{\boldsymbol{w}} \in \tilde{H}^\perp$, i.e. $\langle \phi(\boldsymbol{x}_j), \hat{\boldsymbol{w}} \rangle_H = 0$ for all $j = 1, \ldots, m$. Then, using that $\boldsymbol{w}_*$ minimizes f in $\tilde{H}$,

$$f(\boldsymbol{w}) = \sum_{j=1}^m (\langle \phi(\boldsymbol{x}_j), \boldsymbol{w} \rangle_H - y_j)^2 = \sum_{j=1}^m (\langle \phi(\boldsymbol{x}_j), \tilde{\boldsymbol{w}} \rangle_H - y_j)^2 = f(\tilde{\boldsymbol{w}}) \geq f(\boldsymbol{w}_*).$$

Finally, let $\boldsymbol{w} \in H$ be any minimizer of f in H different from $\boldsymbol{w}_*$. It remains to show $\|\boldsymbol{w}\|_H > \|\boldsymbol{w}_*\|_H$. Decompose again $\boldsymbol{w} = \tilde{\boldsymbol{w}} + \hat{\boldsymbol{w}}$ with $\tilde{\boldsymbol{w}} \in \tilde{H}$ and $\hat{\boldsymbol{w}} \in \tilde{H}^\perp$. As above $f(\boldsymbol{w}) = f(\tilde{\boldsymbol{w}})$ and thus $\tilde{\boldsymbol{w}}$ is a minimizer of f. Uniqueness of $\boldsymbol{w}_*$ in (11.2.7) implies $\tilde{\boldsymbol{w}} = \boldsymbol{w}_*$. Therefore $\hat{\boldsymbol{w}} \neq 0$ and $\|\boldsymbol{w}_*\|_H^2 < \|\tilde{\boldsymbol{w}}\|_H^2 + \|\hat{\boldsymbol{w}}\|_H^2 = \|\boldsymbol{w}\|_H^2$. □

Instead of looking for the minimum norm minimizer $\boldsymbol{w}_*$ in the Hilbert space H, by Proposition 11.2 it suffices to determine the unique minimizer in the at most m-dimensional subspace $\tilde{H}$ spanned by $\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_m)$. This significantly simplifies the problem. To do so we first introduce the notion of kernels.

Definition 11.8. A symmetric function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is called a **kernel** if for any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathbb{R}^d$ the **kernel matrix** $\boldsymbol{G} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m \in \mathbb{R}^{m \times m}$ is symmetric positive semidefinite.

Given a feature map $\phi : \mathbb{R}^d \to H$, it is easy to check that

$$K(\boldsymbol{x}, \boldsymbol{x}') := \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_H \qquad \text{for all } \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d,$$

defines a kernel. The corresponding kernel matrix $\boldsymbol{G} \in \mathbb{R}^{m \times m}$ is given by

$$G_{ij} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_H = K(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

With the ansatz $\boldsymbol{w} = \sum_{j=1}^m \alpha_j \phi(\boldsymbol{x}_j)$, minimizing the objective (11.2.2) in $\tilde{H}$ is equivalent to minimizing

$$\|\boldsymbol{G}\boldsymbol{\alpha} - \boldsymbol{y}\|^2, \tag{11.2.9}$$

in $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^m$.

Proposition 11.9. *Let $\boldsymbol{\alpha} \in \mathbb{R}^m$ be any minimizer of* (11.2.9). *Then $\boldsymbol{w}_* = \sum_{j=1}^m \alpha_j \phi(\boldsymbol{x}_j)$ is the unique minimum H-norm solution of* (11.2.2).

Proposition 11.9, the proof of which is left as an exercise, suggests the following algorithm to compute the kernel least squares estimator:

(i) compute the kernel matrix $\boldsymbol{G} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m$,

(ii) determine a minimizer $\boldsymbol{\alpha} \in \mathbb{R}^m$ of $\|\boldsymbol{G}\boldsymbol{\alpha} - \boldsymbol{y}\|$,

(iii) evaluate $\Phi(\boldsymbol{x}, \boldsymbol{w}_*)$ via

$$\Phi(\boldsymbol{x}, \boldsymbol{w}_*) = \left\langle \phi(\boldsymbol{x}), \sum_{j=1}^m \alpha_j \phi(\boldsymbol{x}_j) \right\rangle_H = \sum_{j=1}^m \alpha_j K(\boldsymbol{x}, \boldsymbol{x}_j). \tag{11.2.10}$$

Thus, minimizing (11.2.2) and expressing the kernel least squares estimator does neither require explicit knowledge of the feature map ϕ nor of the minimum norm solution $\boldsymbol{w}_* \in H$. It is sufficient to choose a kernel map $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$; this is known as the kernel trick. Given a kernel K, we will therefore also refer to (11.2.10) as the kernel least squares estimator without specifying H or ϕ.

Example 11.10. Common examples of kernels include the **polynomial kernel**

$$K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^\top \boldsymbol{x}' + c)^r \qquad c \geq 0,\ r \in \mathbb{N},$$

the radial basis function (**RBF**) **kernel**

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-c\|\boldsymbol{x} - \boldsymbol{x}'\|^2) \qquad c > 0,$$

and the **Laplace kernel**

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-c\|\boldsymbol{x} - \boldsymbol{x}'\|) \qquad c > 0.$$

Remark 11.11. If $\Omega \subseteq \mathbb{R}^d$ is compact and $K : \Omega \times \Omega \to \mathbb{R}$ is a continuous kernel, then Mercer's theorem implies existence of a Hilbert space H and a feature map $\phi : \mathbb{R}^d \to H$ such that

$$K(\boldsymbol{x}, \boldsymbol{x}') = \big\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \big\rangle_H \qquad \text{for all } \boldsymbol{x}, \boldsymbol{x}' \in \Omega,$$

i.e. K is the corresponding kernel. See for instance [217, Thm. 4.49].

11.3 Tangent kernel

Consider again a general model $\Phi(\boldsymbol{x}, \boldsymbol{w})$ with input $\boldsymbol{x} \in \mathbb{R}^d$ and parameters $\boldsymbol{w} \in \mathbb{R}^n$. The goal remains to minimize the square loss objective (11.0.1b) given the data (11.0.1a). If $\boldsymbol{w} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w})$ is not linear, then unlike in Sections 11.1 and 11.2, the objective function (11.0.1b) is in general not convex, and most results on first order methods in Chapter 10 are not directly applicable.

We now simplify the situation by *linearizing the model in $\boldsymbol{w} \in \mathbb{R}^n$ around the initialization*: Fixing $\boldsymbol{w}_0 \in \mathbb{R}^n$, let

$$\Phi^{\text{lin}}(\boldsymbol{x}, \boldsymbol{w}) := \Phi(\boldsymbol{x}, \boldsymbol{w}_0) + \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{w}_0)^\top (\boldsymbol{w} - \boldsymbol{w}_0) \qquad \text{for all } \boldsymbol{w} \in \mathbb{R}^n, \tag{11.3.1}$$

which is the first order Taylor approximation of Φ around the initial parameter $\boldsymbol{w}_0$. Introduce the notation

$$\delta_i := \Phi(\boldsymbol{x}_i, \boldsymbol{w}_0) - \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}_0)^\top \boldsymbol{w}_0 - y_i \qquad \text{for all } i = 1, \dots, m. \tag{11.3.2}$$

The square loss for the linearized model then reads

$$f^{\text{lin}}(\boldsymbol{w}) := \sum_{j=1}^{m} (\Phi^{\text{lin}}(\boldsymbol{x}_i, \boldsymbol{w}) - y_i)^2$$
$$= \sum_{j=1}^{m} (\langle \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}_0), \boldsymbol{w} \rangle + \delta_i)^2, \tag{11.3.3}$$

where $\langle \cdot, \cdot \rangle$ stands for the Euclidean inner product in $\mathbb{R}^n$. Comparing with (11.2.2), minimizing f^{lin} corresponds to a kernel least squares regression with feature map

$$\phi(\boldsymbol{x}) = \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{w}_0) \in \mathbb{R}^n.$$

The corresponding kernel is

$$\hat{K}_n(\boldsymbol{x}, \boldsymbol{x}') = \left\langle \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{w}_0), \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}', \boldsymbol{w}_0) \right\rangle. \tag{11.3.4}$$

We refer to $\hat{K}_n$ as the empirical **tangent kernel**, as it arises from the first order Taylor approximation (the tangent) of the original model Φ around initialization $\boldsymbol{w}_0$. Note that the kernel depends on the choice of $\boldsymbol{w}_0$. As explained in Remark 11.4, training Φ^{lin} with gradient descent yields the kernel least-squares estimator with kernel $\hat{K}_n$ plus an additional term depending on $\boldsymbol{w}_0$.

Of course the linearized model Φ^{lin} only captures the behaviour of Φ for parameters $\boldsymbol{w}$ that are close to $\boldsymbol{w}_0$. If we assume for the moment that during training of Φ, the parameters remain close to initialization, then we can expect similar behaviour and performance of Φ and Φ^{lin}. Under certain assumptions, we will see in the next sections that this is precisely what happens, when the width of a neural network increases. Before we make this precise, in Section 11.4 we investigate whether gradient descent applied to $f(\boldsymbol{w})$ will find a *global* minimizer, under the assumption that Φ^{lin} is a good approximation of Φ.

11.4 Convergence to global minimizers

Intuitively, if $\boldsymbol{w} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w})$ is not linear but "close enough to its linearization" Φ^{lin} defined in (11.3.1), we expect that the objective function is close to a convex function and gradient descent can still find global minimizers of (11.0.1b). To motivate this, consider Figures 11.1 and 11.2 where we chose the number of training data $m = 1$ and the number of parameters $n = 1$. As we can see, essentially we require the difference of Φ and Φ^{lin} and of their derivatives to be small in a neighbourhood of w_0. The size of the neighbourhood crucially depends on the initial error $\Phi(\boldsymbol{x}_1, w_0) - y_1$, and on the size of the derivative $\frac{d}{dw}\Phi(\boldsymbol{x}_1, w_0)$.

For general m and n, we now make the required assumptions on Φ precise.

Assumption 11.12. Let $\Phi \in C^1(\mathbb{R}^d \times \mathbb{R}^n)$ and $\boldsymbol{w}_0 \in \mathbb{R}^n$. There exist constants $r > 0$, U, $L < \infty$ and $0 < \lambda_{\min} \leq \lambda_{\max} < \infty$ such that

(a) the kernel matrix of the empirical tangent kernel

$$(\hat{K}_n(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m = \left(\langle \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}_0), \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_j, \boldsymbol{w}_0) \rangle \right)_{i,j=1}^m \in \mathbb{R}^{m \times m} \tag{11.4.1}$$

is regular and its eigenvalues belong to $[\lambda_{\min}, \lambda_{\max}]$,

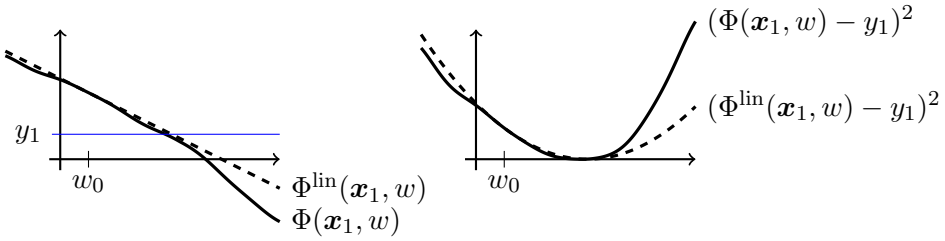


Figure 11.1: Graph of $w \mapsto \Phi(\boldsymbol{x}_1, w)$ and the linearization $w \mapsto \Phi^{\text{lin}}(\boldsymbol{x}_1, w)$ at the initial parameter w_0, s.t. $\frac{d}{dw}\Phi(\boldsymbol{x}_1, w_0) \neq 0$. If Φ and Φ^{lin} are close, then there exists w s.t. $\Phi(\boldsymbol{x}_1, w) = y_1$ (left). If the derivatives are also close, the loss $(\Phi(\boldsymbol{x}_1, w) - y_1)^2$ is nearly convex in w, and gradient descent finds a global minimizer (right).

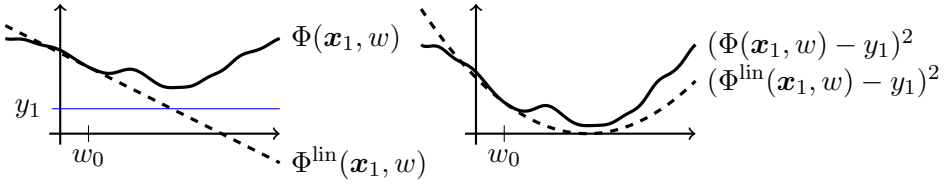


Figure 11.2: Same as Figure 11.1. If Φ and Φ^{lin} are not close, there need not exist w such that $\Phi(\boldsymbol{x}_1, w) = y_1$, and gradient descent need not converge to a global minimizer.

(b) for all $i \in \{1, \dots, m\}$ holds

$$\begin{aligned} \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w})\| &\leq U && \text{for all } \boldsymbol{w} \in B_r(\boldsymbol{w}_0) \\ \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}) - \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{v})\| &\leq L\|\boldsymbol{w} - \boldsymbol{v}\| && \text{for all } \boldsymbol{w},\ \boldsymbol{v} \in B_r(\boldsymbol{w}_0), \end{aligned} \tag{11.4.2}$$

(c) and

$$L \leq \frac{\lambda_{\min}^2}{12 m^{3/2} U^2 \sqrt{f(\boldsymbol{w}_0)}} \qquad \text{and} \qquad r = \frac{2\sqrt{m} U}{\lambda_{\min}} \sqrt{f(\boldsymbol{w}_0)}. \tag{11.4.3}$$

The regularity of the kernel matrix in Assumption 11.12 (a) is equivalent to $(\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}_0)^\top)_{i=1}^m \in \mathbb{R}^{m \times n}$ having full rank $m \leq n$ (in particular we have at least as many parameters n as training data m). In the context of Figure 11.1, this means that $\frac{d}{dw}\Phi(\boldsymbol{x}_1, w_0) \neq 0$ and thus Φ^{lin} is a not a constant function. This condition guarantees that there exists $\boldsymbol{w}$ such that $\Phi^{\text{lin}}(\boldsymbol{x}_i, \boldsymbol{w}) = y_i$ for all $i = 1, \dots, m$. In other words, already the linearized model Φ^{lin} is sufficiently expressive to interpolate the data. Assumption 11.12 (b) formalizes the closeness condition of Φ and Φ^{lin}. Apart from giving an upper bound on $\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w})$, it assumes $\boldsymbol{w} \mapsto \Phi(\boldsymbol{x}_i, \boldsymbol{w})$ to be L-smooth in a ball of radius $r > 0$ around $\boldsymbol{w}_0$, for all $i = 1, \dots, m$. This allows to control how far $\Phi(\boldsymbol{x}_i, \boldsymbol{w})$ and $\Phi^{\text{lin}}(\boldsymbol{x}_i, \boldsymbol{w})$ and their derivatives may deviate from each other for $\boldsymbol{w}$ in this ball. Finally Assumption 11.12 (c) ties together all constants, ensuring the full model to be sufficiently close to its linearization in a large enough neighbourhood of $\boldsymbol{w}_0$.

We are now ready to state the following theorem, which is a variant of [131, Thm. G.1]. In Section 11.5 we will see that its main requirement—Assumption 11.12—is satisfied with high probability for certain (wide) neural networks.

Theorem 11.13. *Let Assumption 11.12 be satisfied and fix a positive learning rate*

$$h \leq \frac{1}{\lambda_{\min} + \lambda_{\max}}. \tag{11.4.4}$$

Set for all $k \in \mathbb{N}$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - h\nabla f(\boldsymbol{w}_k). \tag{11.4.5}$$

It then holds for all $k \in \mathbb{N}$

$$\|\boldsymbol{w}_k - \boldsymbol{w}_0\| \leq \frac{2\sqrt{m}U}{\lambda_{\min}}\sqrt{f(\boldsymbol{w}_0)} \tag{11.4.6a}$$

$$f(\boldsymbol{w}_k) \leq (1 - h\lambda_{\min})^{2k} f(\boldsymbol{w}_0). \tag{11.4.6b}$$

Proof. In the following denote the error in prediction by

$$E(\boldsymbol{w}) := (\Phi(\boldsymbol{x}_i, \boldsymbol{w}) - y_i)_{i=1}^m \in \mathbb{R}^m$$

such that

$$\nabla E(\boldsymbol{w}) = (\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}))_{i=1}^m \in \mathbb{R}^{m \times n}$$

and with the empirical tangent kernel $\hat{K}_n$ in Assumption 11.12

$$\nabla E(\boldsymbol{w}) \nabla E(\boldsymbol{w})^\top = (\hat{K}_n(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m \in \mathbb{R}^{m \times m}. \tag{11.4.7}$$

Moreover, (11.4.2) gives

$$\|\nabla E(\boldsymbol{w})\|^2 \leq \|\nabla E(\boldsymbol{w})\|_F^2 = \sum_{i=1}^m \|\nabla \Phi(\boldsymbol{x}_i, \boldsymbol{w})\|^2 \leq mU^2 \qquad \text{for all } \boldsymbol{w} \in B_r(\boldsymbol{w}_0), \tag{11.4.8a}$$

and similarly

$$\begin{aligned} \|\nabla E(\boldsymbol{w}) - \nabla E(\boldsymbol{v})\|^2 &\leq \sum_{i=1}^m \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}) - \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{v})\|^2 \\ &\leq mL^2 \|\boldsymbol{w} - \boldsymbol{v}\|^2 \qquad \text{for all } \boldsymbol{w},\ \boldsymbol{v} \in B_r(\boldsymbol{w}_0). \end{aligned} \tag{11.4.8b}$$

Denote $c := 1 - h\lambda_{\min} \in (0, 1)$. We use induction over k to prove

$$\sum_{j=0}^{k-1} \|\boldsymbol{w}_{j+1} - \boldsymbol{w}_j\| \leq h2\sqrt{m}U\|E(\boldsymbol{w}_0)\| \sum_{j=0}^{k-1} c^j, \tag{11.4.9a}$$

$$\|E(\boldsymbol{w}_k)\|^2 \leq \|E(\boldsymbol{w}_0)\|^2 c^{2k}, \tag{11.4.9b}$$

for all $k \in \mathbb{N}_0$ and where an empty sum is understood as zero. Since $\sum_{j=0}^\infty c^j = (1-c)^{-1} = (h\lambda_{\min})^{-1}$ and $f(\boldsymbol{w}_k) = \|E(\boldsymbol{w}_k)\|^2$, these inequalities directly imply (11.4.6).

The case $k = 0$ is trivial. For the induction step, assume (11.4.9) holds for some $k \in \mathbb{N}_0$.

Step 1. We show (11.4.9a) for $k+1$. The induction assumption and (11.4.3) give

$$\|\boldsymbol{w}_k - \boldsymbol{w}_0\| \le 2h\sqrt{m}U\|E(\boldsymbol{w}_0)\| \sum_{j=0}^{\infty} c^j = \frac{2\sqrt{m}U}{\lambda_{\min}}\sqrt{f(\boldsymbol{w}_0)} = r, \tag{11.4.10}$$

and thus $\boldsymbol{w}_k \in B_r(\boldsymbol{w}_0)$. Next

$$\nabla f(\boldsymbol{w}_k) = \nabla(E(\boldsymbol{w}_k)^\top E(\boldsymbol{w}_k)) = 2\nabla E(\boldsymbol{w}_k)^\top E(\boldsymbol{w}_k). \tag{11.4.11}$$

Using the iteration rule (11.4.5), the bound (11.4.8a), and (11.4.9b)

$$\begin{aligned}\|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\| &= 2h\|\nabla E(\boldsymbol{w}_k)^\top E(\boldsymbol{w}_k)\| \\ &\le 2h\sqrt{m}U\|E(\boldsymbol{w}_k)\| \\ &\le 2h\sqrt{m}U\|E(\boldsymbol{w}_0)\|c^k.\end{aligned}$$

This shows (11.4.9a) for $k+1$. In particular, as in (11.4.10) we conclude

$$\boldsymbol{w}_{k+1},\ \boldsymbol{w}_k \in B_r(\boldsymbol{w}_0). \tag{11.4.12}$$

Step 2. We show (11.4.9b) for $k+1$. Since E is continuously differentiable, there exists $\tilde{\boldsymbol{w}}_k$ in the convex hull of $\boldsymbol{w}_k$ and $\boldsymbol{w}_{k+1}$ such that

$$E(\boldsymbol{w}_{k+1}) = E(\boldsymbol{w}_k) + \nabla E(\tilde{\boldsymbol{w}}_k)(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k) = E(\boldsymbol{w}_k) - h\nabla E(\tilde{\boldsymbol{w}}_k)\nabla f(\boldsymbol{w}_k),$$

and thus by (11.4.11)

$$\begin{aligned}E(\boldsymbol{w}_{k+1}) &= E(\boldsymbol{w}_k) - 2h\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_k)^\top E(\boldsymbol{w}_k) \\ &= \big(\boldsymbol{I}_m - 2h\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_k)^\top\big)E(\boldsymbol{w}_k),\end{aligned}$$

where $\boldsymbol{I}_m \in \mathbb{R}^{m\times m}$ is the identity matrix. We wish to show that

$$\|\boldsymbol{I}_m - 2h\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_k)^\top\| \le c, \tag{11.4.13}$$

which then implies (11.4.9b) for $k+1$ and concludes the proof.

Using (11.4.8) and the fact that $\boldsymbol{w}_k$, $\tilde{\boldsymbol{w}}_k \in B_r(\boldsymbol{w}_0)$ by (11.4.12),

$$\begin{aligned}&\|\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_k)^\top - \nabla E(\boldsymbol{w}_0)\nabla E(\boldsymbol{w}_0)^\top\| \\ &\quad\le \|\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_k)^\top - \nabla E(\boldsymbol{w}_k)\nabla E(\boldsymbol{w}_k)^\top\| \\ &\qquad + \|\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_k)^\top - \nabla E(\boldsymbol{w}_k)\nabla E(\boldsymbol{w}_0)^\top\| \\ &\qquad + \|\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_0)^\top - \nabla E(\boldsymbol{w}_0)\nabla E(\boldsymbol{w}_0)^\top\| \\ &\quad\le 3mULr.\end{aligned}$$

Since the eigenvalues of $\nabla E(\boldsymbol{w}_0)\nabla E(\boldsymbol{w}_0)^\top$ belong to $[\lambda_{\min}, \lambda_{\max}]$ by (11.4.7) and Assumption 11.12 (a), as long as $h \le (\lambda_{\min} + \lambda_{\max})^{-1}$ we have

$$\begin{aligned}\|\boldsymbol{I}_m - 2h\nabla E(\tilde{\boldsymbol{w}}_k)\nabla E(\boldsymbol{w}_k)^\top\| &\le \|\boldsymbol{I}_m - 2h\nabla E(\boldsymbol{w}_0)\nabla E(\boldsymbol{w}_0)^\top\| + 6hmULr \\ &\le 1 - 2h\lambda_{\min} + 6hmULr \\ &\le 1 - 2h(\lambda_{\min} - 3mULr) \\ &\le 1 - h\lambda_{\min} = c,\end{aligned}$$

where we have used the equality for r and the upper bound for L in (11.4.3). □

Let us emphasize the main statement of Theorem 11.13. By (11.4.6b), full batch gradient descent (11.4.5) achieves zero loss in the limit, i.e. the data is interpolated by the limiting model. In particular, this yields convergence for the (possibly nonconvex) optimization problem of minimizing $f(\boldsymbol{w})$.

11.5 Training dynamics for LeCun initialization

In this and the next section we discuss the implications of Theorem 11.13 for wide neural networks. For ease of presentation we focus on shallow networks with only one hidden layer, but stress that similar considerations also hold for deep networks, see the bibliography section.

11.5.1 Architecture

Let $\Phi : \mathbb{R}^d \to \mathbb{R}$ be a neural network of depth one and width $n \in \mathbb{N}$ of type

$$\Phi(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{v}^\top \sigma(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) + c. \tag{11.5.1}$$

Here $\boldsymbol{x} \in \mathbb{R}^d$ is the input, and $\boldsymbol{U} \in \mathbb{R}^{n\times d}$, $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$ are the parameters which we collect in the vector $\boldsymbol{w} = (\boldsymbol{U}, \boldsymbol{b}, \boldsymbol{v}, c) \in \mathbb{R}^{n(d+2)+1}$ (with $\boldsymbol{U}$ suitably reshaped). For future reference we note that

$$\begin{aligned}
\nabla_{\boldsymbol{U}} \Phi(\boldsymbol{x}, \boldsymbol{w}) &= (\boldsymbol{v} \odot \sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}))\boldsymbol{x}^\top \in \mathbb{R}^{n\times d} \\
\nabla_{\boldsymbol{b}} \Phi(\boldsymbol{x}, \boldsymbol{w}) &= \boldsymbol{v} \odot \sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) \in \mathbb{R}^n \\
\nabla_{\boldsymbol{v}} \Phi(\boldsymbol{x}, \boldsymbol{w}) &= \sigma(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) \in \mathbb{R}^n \\
\nabla_c \Phi(\boldsymbol{x}, \boldsymbol{w}) &= 1 \in \mathbb{R},
\end{aligned} \tag{11.5.2}$$

where $\odot$ denotes the Hadamard product. We also write $\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{w}) \in \mathbb{R}^{n(d+2)+1}$ to denote the full gradient with respect to all parameters.

In practice, it is common to initialize the weights randomly, and in this section we consider so-called LeCun initialization. The following condition on the distribution used for this initialization will be assumed throughout the rest of Section 11.5.

Assumption 11.14. The distribution $\mathcal{W}$ on $\mathbb{R}$ has expectation zero, variance one, and finite moments up to order eight.

To explicitly indicate the expectation and variance in the notation, we also write $\mathcal{W}(0, 1)$ instead of $\mathcal{W}$, and for $\mu \in \mathbb{R}$ and $\varsigma > 0$ we use $\mathcal{W}(\mu, \varsigma^2)$ to denote the corresponding scaled and shifted measure with expectation μ and variance ς^2; thus, if $X \sim \mathcal{W}(0, 1)$ then $\mu + \varsigma X \sim \mathcal{W}(\mu, \varsigma^2)$. LeCun initialization [129] sets the variance of the weights in each layer to be reciprocal to the input dimension of the layer, thereby normalizing the output variance across all network nodes. The initial parameters

$$\boldsymbol{w}_0 = (\boldsymbol{U}_0, \boldsymbol{b}_0, \boldsymbol{v}_0, c_0)$$

are thus randomly initialized with components

$$U_{0;ij} \overset{\text{iid}}{\sim} \mathcal{W}\Big(0, \frac{1}{d}\Big), \qquad v_{0;i} \overset{\text{iid}}{\sim} \mathcal{W}\Big(0, \frac{1}{n}\Big), \qquad b_{0;i},\ c_0 = 0, \tag{11.5.3}$$

independently for all $i = 1, \dots, n$, $j = 1, \dots, d$. For a fixed $\varsigma > 0$ one might choose variances ς^2/d and ς^2/n in (11.5.3), which would require only minor modifications in the rest of this section. Biases

are set to zero for simplicity, with nonzero initialization discussed in the exercises. All expectations and probabilities in Section 11.5 are understood with respect to this random initialization.

Example 11.15. Typical examples for $\mathcal{W}(0,1)$ are the standard normal distribution on $\mathbb{R}$ or the uniform distribution on $[-\sqrt{3},\sqrt{3}]$.

11.5.2 Neural tangent kernel

We begin our analysis by investigating the empirical tangent kernel

$$\hat{K}_n(\boldsymbol{x},\boldsymbol{z}) = \langle \nabla_{\boldsymbol{w}}\Phi(\boldsymbol{x},\boldsymbol{w}_0), \nabla_{\boldsymbol{w}}\Phi(\boldsymbol{z},\boldsymbol{w}_0)\rangle$$

of the shallow network (11.5.1). Scaled properly, it converges in the infinite width limit $n\to\infty$ towards a specific kernel known as the **neural tangent kernel** (NTK). Its precise formula depends on the architecture and initialization. For the LeCun initialization (11.5.3) we denote it by K^{LC}.

Theorem 11.16. *Let $R<\infty$ such that $|\sigma(x)|\le R\cdot(1+|x|)$ and $|\sigma'(x)|\le R\cdot(1+|x|)$ for all $x\in\mathbb{R}$. For any $\boldsymbol{x}$, $\boldsymbol{z}\in\mathbb{R}^d$ and $u_i \overset{\mathrm{iid}}{\sim} \mathcal{W}(0,1/d)$, $i=1,\dots,d$, it then holds*

$$\lim_{n\to\infty}\frac{1}{n}\hat{K}_n(\boldsymbol{x},\boldsymbol{z}) = \mathbb{E}[\sigma(\boldsymbol{u}^\top\boldsymbol{x})\sigma(\boldsymbol{u}^\top\boldsymbol{z})] =: K^{\mathrm{LC}}(\boldsymbol{x},\boldsymbol{z})$$

almost surely.

Moreover, for every δ, $\varepsilon>0$ there exists $n_0(\delta,\varepsilon,R)\in\mathbb{N}$ such that for all $n\ge n_0$ and all $\boldsymbol{x}$, $\boldsymbol{z}\in\mathbb{R}^d$ with $\|\boldsymbol{x}\|$, $\|\boldsymbol{z}\|\le R$

$$\mathbb{P}\Big[\Big\|\frac{1}{n}\hat{K}_n(\boldsymbol{x},\boldsymbol{z}) - K^{\mathrm{LC}}(\boldsymbol{x},\boldsymbol{z})\Big\| < \varepsilon\Big] \ge 1-\delta.$$

Proof. Denote $\boldsymbol{x}^{(1)} = \boldsymbol{U}_0\boldsymbol{x}+\boldsymbol{b}_0\in\mathbb{R}^n$ and $\boldsymbol{z}^{(1)} = \boldsymbol{U}_0\boldsymbol{z}+\boldsymbol{b}_0\in\mathbb{R}^n$. Due to the initialization (11.5.3) and our assumptions on $\mathcal{W}(0,1)$, the components

$$x_i^{(1)} = \sum_{j=1}^d U_{0;ij}x_j \sim \boldsymbol{u}^\top\boldsymbol{x} \qquad i=1,\dots,n$$

are i.i.d. with finite pth moment (independent of n) for all $1\le p\le 8$. Due to the linear growth bound on σ and σ', the same holds for the $(\sigma(x_i^{(1)}))_{i=1}^n$ and the $(\sigma'(x_i^{(1)}))_{i=1}^n$. Similarly, the $(\sigma(z_i^{(1)}))_{i=1}^n$ and $(\sigma'(z_i^{(1)}))_{i=1}^n$ are collections of i.i.d. random variables with finite pth moment for all $1\le p\le 8$.

Denote $\tilde{v}_i = \sqrt{n}v_{0;i}$ such that $\tilde{v}_i \overset{\mathrm{iid}}{\sim}\mathcal{W}(0,1)$. By (11.5.2)

$$\frac{1}{n}\hat{K}_n(\boldsymbol{x},\boldsymbol{z}) = (1+\boldsymbol{x}^\top\boldsymbol{z})\frac{1}{n^2}\sum_{i=1}^n \tilde{v}_i^2\sigma'(x_i^{(1)})\sigma'(z_i^{(1)}) + \frac{1}{n}\sum_{i=1}^n \sigma(x_i^{(1)})\sigma(z_i^{(1)}) + \frac{1}{n}.$$

Since

$$\frac{1}{n}\sum_{i=1}^n \tilde{v}_i^2\sigma'(x_i^{(1)})\sigma'(z_i^{(1)}) \tag{11.5.4}$$

is an average over i.i.d. random variables with finite variance, the law of large numbers implies almost sure convergence of this expression towards

$$\begin{aligned}\mathbb{E}\big[\tilde{v}_i^2\sigma'(x_i^{(1)})\sigma'(z_i^{(1)})\big] &= \mathbb{E}[\tilde{v}_i^2]\mathbb{E}[\sigma'(x_i^{(1)})\sigma'(z_i^{(1)})] \\ &= \mathbb{E}[\sigma'(\boldsymbol{u}^\top \boldsymbol{x})\sigma'(\boldsymbol{u}^\top \boldsymbol{z})],\end{aligned}$$

where we used that $\tilde{v}_i^2$ is independent of $\sigma'(x_i^{(1)})\sigma'(z_i^{(1)})$. By the same argument

$$\frac{1}{n}\sum_{i=1}^{n}\sigma(x_i^{(1)})\sigma(z_i^{(1)}) \to \mathbb{E}[\sigma(\boldsymbol{u}^\top \boldsymbol{x})\sigma(\boldsymbol{u}^\top \boldsymbol{z})]$$

almost surely as $n \to \infty$. This shows the first statement.

The existence of n_0 follows similarly by an application of Theorem A.23. □

Example 11.17 (K^{LC} for ReLU)**.** Let $\sigma(x) = \max\{0, x\}$ and let $\mathcal{W}(0,1)$ be the standard normal distribution. For $\boldsymbol{x}$, $\boldsymbol{z} \in \mathbb{R}^d$ denote by

$$\theta = \arccos\left(\frac{\boldsymbol{x}^\top \boldsymbol{z}}{\|\boldsymbol{x}\|\|\boldsymbol{z}\|}\right)$$

the angle between these vectors. Then according to [37, Appendix A], it holds with $u_i \overset{\text{iid}}{\sim} \mathcal{W}(0,1)$, $i = 1, \dots, d$,

$$K^{\mathrm{LC}}(\boldsymbol{x}, \boldsymbol{z}) = \mathbb{E}[\sigma(\boldsymbol{u}^\top \boldsymbol{x})\sigma(\boldsymbol{u}^\top \boldsymbol{z})] = \frac{\|\boldsymbol{x}\|\|\boldsymbol{z}\|}{2\pi d}(\sin(\theta) + (\pi - \theta)\cos(\theta)).$$

11.5.3 Gradient descent

We now proceed similar as in [131, Appendix G], to show that Theorem 11.13 is applicable to the wide neural network (11.5.1) with high probability under random initialization (11.5.3). This will imply that gradient descent can find global minimizers when training wide neural networks. We work under the following assumptions on the activation function and training data.

Assumption 11.18. There exist $R < \infty$ and $0 < \lambda_{\min}^{\mathrm{LC}} \leq \lambda_{\max}^{\mathrm{LC}} < \infty$ such that

(a) for the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ holds $|\sigma(0)|$, $\mathrm{Lip}(\sigma)$, $\mathrm{Lip}(\sigma') \leq R$,

(b) $\|\boldsymbol{x}_i\|$, $|y_i| \leq R$ for all training data $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$, $i = 1, \dots, m$,

(c) the kernel matrix of the neural tangent kernel

$$(K^{\mathrm{LC}}(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m \in \mathbb{R}^{m\times m}$$

is regular and its eigenvalues belong to $[\lambda_{\min}^{\mathrm{LC}}, \lambda_{\max}^{\mathrm{LC}}]$.

We start by showing Assumption 11.12 (a) for the present setting. More precisely, we give bounds for the eigenvalues of the empirical tangent kernel.

Lemma 11.19. *Let Assumption 11.18 be satisfied. Then for every $\delta > 0$ there exists $n_0(\delta, \lambda_{\min}^{\mathrm{LC}}, m, R) \in \mathbb{R}$ such that for all $n \geq n_0$ with probability at least $1-\delta$ all eigenvalues of*

$$(\hat{K}_n(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m = \big(\langle \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_i, \boldsymbol{w}_0), \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}_j, \boldsymbol{w}_0) \rangle \big)_{i,j=1}^m \in \mathbb{R}^{m \times m}$$

belong to $[n\lambda_{\min}^{\mathrm{LC}}/2, 2n\lambda_{\max}^{\mathrm{LC}}]$.

Proof. Denote $\hat{\boldsymbol{G}}_n := (\hat{K}_n(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m$ and $\boldsymbol{G}^{\mathrm{LC}} := (K^{\mathrm{LC}}(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^m$. By Theorem 11.16, there exists n_0 such that for all $n \geq n_0$ holds with probability at least $1-\delta$ that

$$\left\| \boldsymbol{G}^{\mathrm{LC}} - \frac{1}{n} \hat{\boldsymbol{G}}_n \right\| \leq \frac{\lambda_{\min}^{\mathrm{LC}}}{2}.$$

Assuming this bound to hold

$$\frac{1}{n}\|\hat{\boldsymbol{G}}_n\| = \sup_{\substack{\boldsymbol{a} \in \mathbb{R}^m \\ \|\boldsymbol{a}\|=1}} \frac{1}{n} \|\hat{\boldsymbol{G}}_n \boldsymbol{a}\| \geq \inf_{\substack{\boldsymbol{a} \in \mathbb{R}^m \\ \|\boldsymbol{a}\|=1}} \|\boldsymbol{G}^{\mathrm{LC}} \boldsymbol{a}\| - \frac{\lambda_{\min}^{\mathrm{LC}}}{2} \geq \lambda_{\min}^{\mathrm{LC}} - \frac{\lambda_{\min}^{\mathrm{LC}}}{2} \geq \frac{\lambda_{\min}^{\mathrm{LC}}}{2},$$

where we have used that $\lambda_{\min}^{\mathrm{LC}}$ is the smallest eigenvalue, and thus singular value, of the symmetric positive definite matrix $\boldsymbol{G}^{\mathrm{LC}}$. This shows that the smallest eigenvalue of $\hat{\boldsymbol{G}}_n$ is larger or equal to $\lambda_{\min}^{\mathrm{LC}}/2$. Similarly, we conclude that the largest eigenvalue is bounded from above by $\lambda_{\max}^{\mathrm{LC}} + \lambda_{\min}^{\mathrm{LC}}/2 \leq \lambda_{\max}^{\mathrm{LC}}$. This concludes the proof. □

Next we check Assumption 11.12 (b). To this end we first bound the norm of a random matrix.

Lemma 11.20. *Let $\mathcal{W}(0,1)$ be as in Assumption 11.14, and let $\boldsymbol{W} \in \mathbb{R}^{n \times d}$ with $W_{ij} \overset{\mathrm{iid}}{\sim} \mathcal{W}(0,1)$. Denote the fourth moment of $\mathcal{W}(0,1)$ by μ_4. Then*

$$\mathbb{P}\Big[\|\boldsymbol{W}\| \leq \sqrt{n(d+1)} \Big] \geq 1 - \frac{d\mu_4}{n}.$$

Proof. It holds

$$\|\boldsymbol{W}\| \leq \|\boldsymbol{W}\|_F = \Big(\sum_{i=1}^n \sum_{j=1}^d W_{ij}^2 \Big)^{1/2}.$$

The $\alpha_i := \sum_{j=1}^d W_{ij}^2$, $i = 1, \ldots, n$, are i.i.d. distributed with expectation d and finite variance dC, where $C \leq \mu_4$ is the variance of W_{11}^2. By Theorem A.23

$$\mathbb{P}\Big[\|\boldsymbol{W}\| > \sqrt{n(d+1)} \Big] \leq \mathbb{P}\Big[\frac{1}{n} \sum_{i=1}^n \alpha_i > d+1 \Big] \leq \mathbb{P}\Big[\Big| \frac{1}{n} \sum_{i=1}^n \alpha_i - d \Big| > 1 \Big] \leq \frac{d\mu_4}{n},$$

which concludes the proof. □

Lemma 11.21. *Let Assumption 11.18 (a) be satisfied with some constant R. Then there exists $M(R)$, and for all c, $\delta > 0$ there exists $n_0(c, d, \delta, R) \in \mathbb{N}$ such that for all $n \geq n_0$ it holds with probability at least $1 - \delta$*

$$\begin{aligned} \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{w})\| &\leq M\sqrt{n} && \text{for all } \boldsymbol{w} \in B_{cn^{-1/2}}(\boldsymbol{w}_0) \\ \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{w}) - \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{v})\| &\leq M\sqrt{n}\|\boldsymbol{w} - \boldsymbol{v}\| && \text{for all } \boldsymbol{w},\ \boldsymbol{v} \in B_{cn^{-1/2}}(\boldsymbol{w}_0) \end{aligned}$$

for all $\boldsymbol{x} \in \mathbb{R}^d$ with $\|\boldsymbol{x}\| \leq R$.

Proof. Due to the initialization (11.5.3), by Lemma 11.20 we can find $n_0(\delta, d)$ such that for all $n \geq n_0$ holds with probability at least $1 - \delta$ that

$$\|\boldsymbol{v}_0\| \leq 2 \qquad \text{and} \qquad \|\boldsymbol{U}_0\| \leq 2\sqrt{n}. \tag{11.5.5}$$

For the rest of this proof we fix arbitrary $\boldsymbol{x} \in \mathbb{R}^d$ and $n \geq n_0 \geq c^2$ such that

$$\|\boldsymbol{x}\| \leq R \qquad \text{and} \qquad n^{-1/2}c \leq 1.$$

We need to show that the claimed inequalities hold as long as (11.5.5) is satisfied. We will several times use that for all $\boldsymbol{p}$, $\boldsymbol{q} \in \mathbb{R}^n$

$$\|\boldsymbol{p} \odot \boldsymbol{q}\| \leq \|\boldsymbol{p}\|\|\boldsymbol{q}\| \qquad \text{and} \qquad \|\sigma(\boldsymbol{p})\| \leq R\sqrt{n} + R\|\boldsymbol{p}\|$$

since $|\sigma(x)| \leq R \cdot (1 + |x|)$. The same holds for σ'.

Step 1. We show the bound on the gradient. Fix

$$\boldsymbol{w} = (\boldsymbol{U}, \boldsymbol{b}, \boldsymbol{v}, c) \qquad \text{s.t.} \qquad \|\boldsymbol{w} - \boldsymbol{w}_0\| \leq cn^{-1/2}.$$

Using formula (11.5.2) for $\nabla_{\boldsymbol{b}} \Phi$ and the above inequalities

$$\begin{aligned} \|\nabla_{\boldsymbol{b}} \Phi(\boldsymbol{x}, \boldsymbol{w})\| &\leq \|\nabla_{\boldsymbol{b}} \Phi(\boldsymbol{x}, \boldsymbol{w}_0)\| + \|\nabla_{\boldsymbol{b}} \Phi(\boldsymbol{x}, \boldsymbol{w}) - \nabla_{\boldsymbol{b}} \Phi(\boldsymbol{x}, \boldsymbol{w}_0)\| \\ &= \|\boldsymbol{v}_0 \odot \sigma'(\boldsymbol{U}_0\boldsymbol{x})\| + \|\boldsymbol{v} \odot \sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) - \boldsymbol{v}_0 \odot \sigma'(\boldsymbol{U}_0\boldsymbol{x})\| \\ &\leq 2(R\sqrt{n} + 2R^2\sqrt{n}) + \|\boldsymbol{v} \odot \sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) - \boldsymbol{v}_0 \odot \sigma'(\boldsymbol{U}_0\boldsymbol{x})\|. \end{aligned} \tag{11.5.6}$$

Due to

$$\|\boldsymbol{U}\| \leq \|\boldsymbol{U}_0\| + \|\boldsymbol{U}_0 - \boldsymbol{U}\|_F \leq 2\sqrt{n} + cn^{-1/2} \leq 3\sqrt{n}, \tag{11.5.7}$$

the last norm in (11.5.6) is bounded by

$$\begin{aligned} &\|(\boldsymbol{v} - \boldsymbol{v}_0) \odot \sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})\| + \|\boldsymbol{v}_0 \odot (\sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) - \sigma'(\boldsymbol{U}_0\boldsymbol{x}))\| \\ &\quad \leq cn^{-1/2}(R\sqrt{n} + R \cdot (\|\boldsymbol{U}\|\|\boldsymbol{x}\| + \|\boldsymbol{b}\|)) + 2R \cdot (\|\boldsymbol{U} - \boldsymbol{U}_0\|\|\boldsymbol{x}\| + \|\boldsymbol{b}\|) \\ &\quad \leq R\sqrt{n} + 3\sqrt{n}R^2 + cn^{-1/2}R + 2R \cdot (cn^{-1/2}R + cn^{-1/2}) \\ &\quad \leq \sqrt{n}(4R + 5R^2) \end{aligned}$$

and therefore

$$\|\nabla_{\boldsymbol{b}} \Phi(\boldsymbol{x}, \boldsymbol{w})\| \leq \sqrt{n}(6R + 9R^2).$$

For the gradient with respect to $\boldsymbol{U}$ we use $\nabla_{\boldsymbol{U}}\Phi(\boldsymbol{x},\boldsymbol{w}) = \nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x},\boldsymbol{w})\boldsymbol{x}^\top$, so that

$$\|\nabla_{\boldsymbol{U}}\Phi(\boldsymbol{x},\boldsymbol{w})\|_F = \|\nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x},\boldsymbol{w})\boldsymbol{x}^\top\|_F = \|\nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x},\boldsymbol{w})\|\,\|\boldsymbol{x}\| \le \sqrt{n}(6R^2+9R^3).$$

Next

$$\begin{aligned}\|\nabla_{\boldsymbol{v}}\Phi(\boldsymbol{x},\boldsymbol{w})\| &= \|\sigma(\boldsymbol{U}\boldsymbol{x}+\boldsymbol{b})\| \\ &\le R\sqrt{n} + R\|\boldsymbol{U}\boldsymbol{x}+\boldsymbol{b}\| \\ &\le R\sqrt{n} + R\cdot(3\sqrt{n}R + cn^{-1/2}) \\ &\le \sqrt{n}(2R+3R^2),\end{aligned}$$

and finally $\nabla_c\Phi(\boldsymbol{x},\boldsymbol{w}) = 1$. In all, with $M_1(R) := (1+8R+12R^2)$

$$\|\nabla_{\boldsymbol{w}}\Phi(\boldsymbol{x},\tilde{\boldsymbol{w}})\| \le \sqrt{n}M_1(R).$$

Step 2. We show Lipschitz continuity. Fix

$$\boldsymbol{w} = (\boldsymbol{U},\boldsymbol{b},\boldsymbol{v},c) \qquad \text{and} \qquad \tilde{\boldsymbol{w}} = (\tilde{\boldsymbol{U}},\tilde{\boldsymbol{b}},\tilde{\boldsymbol{v}},\tilde{c})$$

such that $\|\boldsymbol{w}-\boldsymbol{w}_0\|$, $\|\tilde{\boldsymbol{w}}-\boldsymbol{w}_0\| \le cn^{-1/2}$. Then

$$\|\nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x},\boldsymbol{w}) - \nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x},\tilde{\boldsymbol{w}})\| = \|\boldsymbol{v}\odot\sigma'(\boldsymbol{U}\boldsymbol{x}+\boldsymbol{b}) - \tilde{\boldsymbol{v}}\odot\sigma'(\tilde{\boldsymbol{U}}\boldsymbol{x}+\tilde{\boldsymbol{b}})\|.$$

Using $\|\tilde{\boldsymbol{v}}\| \le \|\boldsymbol{v}_0\| + cn^{-1/2} \le 3$ and (11.5.7), this term is bounded by

$$\begin{aligned}&\|(\boldsymbol{v}-\tilde{\boldsymbol{v}})\odot\sigma'(\boldsymbol{U}\boldsymbol{x}+\boldsymbol{b})\| + \|\tilde{\boldsymbol{v}}\odot(\sigma'(\boldsymbol{U}\boldsymbol{x}+\boldsymbol{b}) - \sigma'(\tilde{\boldsymbol{U}}\boldsymbol{x}+\tilde{\boldsymbol{b}}))\| \\ &\quad\le \|\boldsymbol{v}-\tilde{\boldsymbol{v}}\|(R\sqrt{n} + R\cdot(\|\boldsymbol{U}\|\|\boldsymbol{x}\| + \|\boldsymbol{b}\|)) + 3R\cdot(\|\boldsymbol{x}\|\|\boldsymbol{U}-\tilde{\boldsymbol{U}}\| + \|\boldsymbol{b}-\tilde{\boldsymbol{b}}\|) \\ &\quad\le \|\boldsymbol{w}-\tilde{\boldsymbol{w}}\|\sqrt{n}(5R+6R^2).\end{aligned}$$

For $\nabla_{\boldsymbol{U}}\Phi(\boldsymbol{x},\boldsymbol{w})$ we obtain similar as in Step 1

$$\begin{aligned}\|\nabla_{\boldsymbol{U}}\Phi(\boldsymbol{x},\boldsymbol{w}) - \nabla_{\boldsymbol{U}}\Phi(\boldsymbol{x},\tilde{\boldsymbol{w}})\|_F &= \|\boldsymbol{x}\|\|\nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x},\boldsymbol{w}) - \nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x},\tilde{\boldsymbol{w}})\| \\ &\le \|\boldsymbol{w}-\tilde{\boldsymbol{w}}\|\sqrt{n}(5R^2+6R^3).\end{aligned}$$

Next

$$\begin{aligned}\|\nabla_{\boldsymbol{v}}\Phi(\boldsymbol{x},\boldsymbol{w}) - \nabla_{\boldsymbol{v}}\Phi(\boldsymbol{x},\tilde{\boldsymbol{w}})\| &= \|\sigma(\boldsymbol{U}\boldsymbol{x}+\boldsymbol{b}) - \sigma(\tilde{\boldsymbol{U}}\boldsymbol{x}-\tilde{\boldsymbol{b}})\| \\ &\le R\cdot(\|\boldsymbol{U}-\tilde{\boldsymbol{U}}\|\|\boldsymbol{x}\| + \|\boldsymbol{b}-\tilde{\boldsymbol{b}}\|) \\ &\le \|\boldsymbol{w}-\tilde{\boldsymbol{w}}\|(R^2+R)\end{aligned}$$

and finally $\nabla_c\Phi(\boldsymbol{x},\boldsymbol{w}) = 1$ is constant. With $M_2(R) := R+6R^2+6R^3$ this shows

$$\|\nabla_{\boldsymbol{w}}\Phi(\boldsymbol{x},\boldsymbol{w}) - \nabla_{\boldsymbol{w}}\Phi(\boldsymbol{x},\tilde{\boldsymbol{w}})\| \le \sqrt{n}M_2(R)\|\boldsymbol{w}-\tilde{\boldsymbol{w}}\|.$$

In all, this concludes the proof with $M(R) := \max\{M_1(R), M_2(R)\}$. □

Before coming to the main result of this section, we first show that the initial error $f(\boldsymbol{w}_0)$ remains bounded with high probability.

Lemma 11.22. *Let Assumption 11.18 (a), (b) be satisfied. Then for every $\delta > 0$ exists $R_0(\delta, m, R) > 0$ such that for all $n \in \mathbb{N}$*

$$\mathbb{P}[f(\boldsymbol{w}_0) \leq R_0] \geq 1 - \delta.$$

Proof. Let $i \in \{1, \dots, m\}$, and set $\boldsymbol{\alpha} := \boldsymbol{U}_0 \boldsymbol{x}_i$ and $\tilde{v}_j := \sqrt{n} v_{0;j}$ for $j = 1, \dots, n$, so that $\tilde{v}_j \overset{\text{iid}}{\sim} \mathcal{W}(0, 1)$. Then

$$\Phi(\boldsymbol{x}_i, \boldsymbol{w}_0) = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \tilde{v}_j \sigma(\alpha_j).$$

By Assumption 11.14 and (11.5.3), the $\tilde{v}_j \sigma(\alpha_j)$, $j = 1, \dots, n$, are i.i.d. centered random variables with finite variance bounded by a constant $C(R)$ independent of n. Thus the variance of $\Phi(\boldsymbol{x}_i, \boldsymbol{w}_0)$ is also bounded by $C(R)$. By Chebyshev's inequality, see Lemma A.22, for every $k > 0$

$$\mathbb{P}[|\Phi(\boldsymbol{x}_i, \boldsymbol{w}_0)| \geq k\sqrt{C}] \leq \frac{1}{k^2}.$$

Setting $k = \sqrt{m/\delta}$

$$\begin{aligned} \mathbb{P}\Big[\sum_{i=1}^{m} |\Phi(\boldsymbol{x}_i, \boldsymbol{w}_0) - y_i|^2 \geq m(k\sqrt{C} + R)^2 \Big] &\leq \sum_{i=1}^{m} \mathbb{P}\Big[|\Phi(\boldsymbol{x}_i, \boldsymbol{w}_0) - y_i| \geq k\sqrt{C} + R \Big] \\ &\leq \sum_{i=1}^{m} \mathbb{P}\Big[|\Phi(\boldsymbol{x}_i, \boldsymbol{w}_0)| \geq k\sqrt{C} \Big] \leq \delta, \end{aligned}$$

which shows the claim with $R_0 = m \cdot (\sqrt{Cm/\delta} + R)^2$. □

The next theorem is the main result of this section. It states that in the present setting gradient descent converges to a global minimizer and the limiting network achieves zero loss, i.e. interpolates the data. Moreover, during training the network weights remain close to initialization if the network width n is large.

Theorem 11.23. *Let Assumption 11.18 be satisfied, and let the parameters $\boldsymbol{w}_0$ of the neural network Φ in* (11.5.1) *be initialized according to* (11.5.3)*. Fix a learning rate*

$$h < \frac{2}{\lambda_{\min}^{\mathrm{LC}} + 4\lambda_{\max}^{\mathrm{LC}}} \frac{1}{n}$$

and with the objective function (11.0.1b) *let for all $k \in \mathbb{N}$*

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - h \nabla f(\boldsymbol{w}_k).$$

Then for every $\delta > 0$ there exist $C > 0$, $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ holds with probability at least $1 - \delta$ that for all $k \in \mathbb{N}$

$$\|\boldsymbol{w}_k - \boldsymbol{w}_0\| \leq \frac{C}{\sqrt{n}}$$
$$f(\boldsymbol{w}_k) \leq C\Big(1 - \frac{hn}{2\lambda_{\min}^{\mathrm{LC}}}\Big)^{2k}.$$

Proof. We wish to apply Theorem 11.13, which requires Assumption 11.12 to be satisfied. By Lemma 11.19, 11.21 and 11.22, for every $c > 0$ we can find n_0 such that for all $n \geq n_0$ with probability at least $1 - \delta$ we have $\sqrt{f(\boldsymbol{w}_0)} \leq \sqrt{R_0}$ and Assumption 11.12 (a), (b) holds with the values

$$L = M\sqrt{n}, \quad U = M\sqrt{n}, \quad r = cn^{-1/2}, \quad \lambda_{\min} = \frac{n\lambda_{\min}^{\mathrm{LC}}}{2}, \quad \lambda_{\max} = 2n\lambda_{\max}^{\mathrm{LC}}.$$

For Assumption 11.12 (c), it suffices that

$$M\sqrt{n} \leq \frac{n^2(\lambda_{\min}^{\mathrm{LC}}/2)^2}{12m^{3/2}M^2n\sqrt{R_0}} \qquad \text{and} \qquad cn^{-1/2} \geq \frac{2mM\sqrt{n}}{n}\sqrt{R_0}.$$

Choosing $c > 0$ and n large enough, the inequalities hold. The statement is now a direct consequence of Theorem 11.13. □

11.5.4 Proximity to linearized model

The analysis thus far was based on the linearization Φ^{lin} describing the behaviour of the full network Φ well in a neighbourhood of the initial parameters $\boldsymbol{w}_0$. Moreover, Theorem 11.23 states that the parameters remain in an $O(n^{-1/2})$ neighbourhood of $\boldsymbol{w}_0$ during training. This suggests that the trained full model $\lim_{k\to\infty} \Phi(\boldsymbol{x}, \boldsymbol{w}_k)$ yields predictions similar to the trained linearized model.

To describe this phenomenon, we adopt again the notations $\Phi^{\mathrm{lin}} : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}$ and f^{lin} from (11.3.1) and (11.3.3). Initializing $\boldsymbol{w}_0$ according to (11.5.3) and setting $\boldsymbol{p}_0 = \boldsymbol{w}_0$, gradient descent computes the parameter updates

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - h\nabla_{\boldsymbol{w}} f(\boldsymbol{w}_k), \qquad \boldsymbol{p}_{k+1} = \boldsymbol{p}_k - h\nabla_{\boldsymbol{w}} f^{\mathrm{lin}}(\boldsymbol{p}_k)$$

for the full and linearized models, respectively. Let us consider the dynamics of the prediction of the network on the training data. Writing

$$\Phi(\boldsymbol{X}, \boldsymbol{w}) := (\Phi(\boldsymbol{x}_i, \boldsymbol{w}))_{i=1}^m \in \mathbb{R}^m \qquad \text{such that} \qquad \nabla_{\boldsymbol{w}}\Phi(\boldsymbol{X}, \boldsymbol{w}) \in \mathbb{R}^{m\times n}$$

it holds

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \|\Phi(\boldsymbol{X}, \boldsymbol{w}) - \boldsymbol{y}\|^2 = 2\nabla_{\boldsymbol{w}}\Phi(\boldsymbol{X}, \boldsymbol{w})^\top (\Phi(\boldsymbol{X}, \boldsymbol{w}) - \boldsymbol{y}).$$

Thus for the full model

$$\begin{aligned}\Phi(\boldsymbol{X}, \boldsymbol{w}_{k+1}) &= \Phi(\boldsymbol{X}, \boldsymbol{w}_k) + \nabla_{\boldsymbol{w}}\Phi(\boldsymbol{X}, \tilde{\boldsymbol{w}}_k)(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k) \\ &= \Phi(\boldsymbol{X}, \boldsymbol{w}_k) - 2h\nabla_{\boldsymbol{w}}\Phi(\boldsymbol{X}, \tilde{\boldsymbol{w}}_k)\nabla_{\boldsymbol{w}}\Phi(\boldsymbol{X}, \boldsymbol{w}_k)^\top (\Phi(\boldsymbol{X}, \boldsymbol{w}_k) - \boldsymbol{y}), \end{aligned} \tag{11.5.8}$$

where $\tilde{\boldsymbol{w}}_k$ is in the convex hull of $\boldsymbol{w}_k$ and $\boldsymbol{w}_{k+1}$.

Similarly, for the linearized model with (cp. (11.3.1))

$$\Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{w}) := (\Phi^{\mathrm{lin}}(\boldsymbol{x}_i, \boldsymbol{w}))_{i=1}^m \in \mathbb{R}^m \qquad \text{and} \qquad \nabla_{\boldsymbol{p}} \Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}) = \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0) \in \mathbb{R}^{m \times n}$$

such that

$$\nabla_{\boldsymbol{p}} f^{\mathrm{lin}}(\boldsymbol{p}) = \nabla_{\boldsymbol{p}} \|\Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}) - \boldsymbol{y}\|^2 = 2 \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0)^\top (\Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}) - \boldsymbol{y})$$

and

$$\begin{aligned} \Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}_{k+1}) &= \Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}_k) + \nabla_{\boldsymbol{p}} \Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}_0)(\boldsymbol{p}_{k+1} - \boldsymbol{p}_k) \\ &= \Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}_k) - 2h \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0)^\top (\Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}_k) - \boldsymbol{y}). \end{aligned} \tag{11.5.9}$$

Remark 11.24. From (11.5.9) it is easy to see that with $\boldsymbol{A} := 2h \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0)^\top$ and $\boldsymbol{B} := \boldsymbol{I}_m - \boldsymbol{A}$ holds the explicit formula

$$\Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}_k) = \boldsymbol{B}^k \Phi^{\mathrm{lin}}(\boldsymbol{X}, \boldsymbol{p}_0) + \sum_{j=0}^{k-1} \boldsymbol{B}^k \boldsymbol{A} \boldsymbol{y}$$

for the prediction of the linear model in step k. Note that if $\boldsymbol{A}$ is regular and h is small enough, then $\boldsymbol{B}^k$ converges to the zero matrix as $k \to \infty$ and $\sum_{j=0}^{\infty} \boldsymbol{B}^k = \boldsymbol{A}^{-1}$ since this is a Neumann series.

Comparing the two dynamics (11.5.8) and (11.5.9), the difference only lies in the two $\mathbb{R}^{m \times m}$ matrices

$$2h \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \tilde{\boldsymbol{w}}_k) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_k)^\top \qquad \text{and} \qquad 2h \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_0)^\top.$$

Recall that the step size h in Theorem 11.23 scales like $1/n$.

Proposition 11.25. *Consider the setting of Theorem 11.23. Then there exists $C < \infty$, and for every $\delta > 0$ there exists n_0 such that for all $n \geq n_0$ holds with probability at least $1 - \delta$ that for all $k \in \mathbb{N}$*

$$\frac{1}{n} \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \tilde{\boldsymbol{w}}_k) \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_k)^\top - \nabla_{\boldsymbol{p}} \Phi(\boldsymbol{X}, \boldsymbol{p}_0) \nabla_{\boldsymbol{p}} \Phi(\boldsymbol{X}, \boldsymbol{p}_0)^\top\| \leq C n^{-1/2}.$$

Proof. Consider the setting of the proof of Theorem 11.23. Then for every $k \in \mathbb{N}$ holds $\|\boldsymbol{w}_k - \boldsymbol{w}_0\| \leq r$ and thus also $\|\tilde{\boldsymbol{w}}_k - \boldsymbol{w}_0\| \leq r$, where $r = cn^{-1/2}$. Thus Lemma 11.21 implies the norm to be bounded by

$$\begin{aligned} &\frac{1}{n} \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \tilde{\boldsymbol{w}}_k) - \nabla_{\boldsymbol{p}} \Phi(\boldsymbol{X}, \boldsymbol{p}_0)\| \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_k)^\top\| + \\ &\qquad \frac{1}{n} \|\nabla_{\boldsymbol{p}} \Phi(\boldsymbol{X}, \boldsymbol{p}_0)\| \|\nabla_{\boldsymbol{w}} \Phi(\boldsymbol{X}, \boldsymbol{w}_k)^\top - \nabla_{\boldsymbol{p}} \Phi(\boldsymbol{X}, \boldsymbol{p}_0)^\top\| \\ &\leq mM(\|\tilde{\boldsymbol{w}}_k - \boldsymbol{p}_0\| + \|\boldsymbol{w}_k - \boldsymbol{p}_0\|) \leq cmMn^{-1/2} \end{aligned}$$

which gives the statement. □

By Proposition 11.25 the two matrices driving the dynamics (11.5.8) and (11.5.9) remain in an $O(n^{-1/2})$ neighbourhood of each other throughout training. This allows to show the following proposition, which states that the prediction function learned by the network gets arbitrarily close to the one learned by the linearized version in the limit $n \to \infty$. The proof, which we omit, is based on Grönwall's inequality. See [106, 131].

Proposition 11.26. *Consider the setting of Theorem 11.23. Then there exists $C < \infty$, and for every $\delta > 0$ there exists n_0 such that for all $n \geq n_0$ holds with probability at least $1 - \delta$ that for all $\|\boldsymbol{x}\| \leq 1$*

$$\sup_{k \in \mathbb{N}} |\Phi(\boldsymbol{x}, \boldsymbol{w}_k) - \Phi^{\text{lin}}(\boldsymbol{x}, \boldsymbol{p}_k)| \leq C n^{-1/2}.$$

11.5.5 Connection to Gaussian processes

In the previous section, we established that for large widths, the trained neural network mirrors the behaviour of the trained linearized model, which itself is closely connected to kernel least-squares with the neural tangent kernel. Yet, as pointed out in Remark 11.4, the obtained model still strongly depends on the choice of random initialization $\boldsymbol{w}_0 \in \mathbb{R}^n$. We should thus understand both the model at initialization $\boldsymbol{x} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w}_0)$ and the model after training $\boldsymbol{x} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w}_k)$, as random draws of a certain distribution over functions. To make this precise, let us introduce Gaussian processes.

Definition 11.27. Let $(\Omega, \mathbb{P})$ be a probability space, and let $g : \mathbb{R}^d \times \Omega \to \mathbb{R}$. We call g a **Gaussian process** with mean function $m : \mathbb{R}^d \to \mathbb{R}$ and covariance function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ if

(a) for each $\boldsymbol{x} \in \mathbb{R}^d$ holds $\omega \mapsto g(\boldsymbol{x}, \omega)$ is a random variable,

(b) for all $k \in \mathbb{N}$ and all $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k \in \mathbb{R}^d$ the random variables $g(\boldsymbol{x}_1, \cdot), \dots, g(\boldsymbol{x}_k, \cdot)$ have a joint Gaussian distribution such that

$$(g(\boldsymbol{x}_1, \omega), \dots, g(\boldsymbol{x}_k, \omega)) \sim \mathrm{N}\Big(m(\boldsymbol{x}_i)_{i=1}^k, (c(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^k\Big).$$

In words, g is a Gaussian process, if $\omega \mapsto g(\boldsymbol{x}, \omega)$ defines a collection of random variables indexed over $\boldsymbol{x} \in \mathbb{R}^d$, such that the joint distribution of $(g(\boldsymbol{x}_1, \cdot))_{j=1}^n$ is a Gaussian whose mean and variance are determined by m and c respectively. Fixing $\omega \in \Omega$, we can then interpret $\boldsymbol{x} \mapsto g(\boldsymbol{x}, \omega)$ as a random draw from a distribution over functions.

As first observed in [157], certain neural networks at initialization tend to Gaussian processes in the infinite width limit.

Proposition 11.28. *Consider depth-n networks Φ_n as in* (11.5.1) *with initialization* (11.5.3)*, and define with $u_i \overset{\text{iid}}{\sim} \mathcal{W}(0, 1/d)$, $i = 1, \dots, d$,*

$$c(\boldsymbol{x}, \boldsymbol{z}) := \mathbb{E}[\sigma(\boldsymbol{u}^\top \boldsymbol{x})\sigma(\boldsymbol{u}^\top \boldsymbol{z})] \qquad \text{for all } \boldsymbol{x},\ \boldsymbol{z} \in \mathbb{R}^d.$$

Then for all distinct $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k \in \mathbb{R}^d$ it holds that

$$\lim_{n\to\infty} (\Phi_n(\boldsymbol{x}_1, \boldsymbol{w}_0), \dots, \Phi_n(\boldsymbol{x}_k, \boldsymbol{w}_0)) \sim \mathrm{N}(\boldsymbol{0}, (c(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^k)$$

with weak convergence.

Proof. Set $\tilde{v}_i := \sqrt{n} v_{0,i}$ and $\tilde{\boldsymbol{u}}_i = (U_{0,i1}, \dots, U_{0,id}) \in \mathbb{R}^d$, so that $\tilde{v}_i \overset{\text{iid}}{\sim} \mathcal{W}(0, 1)$, and the $\tilde{\boldsymbol{u}}_i \in \mathbb{R}^d$ are also i.i.d., with each component distributed according to $\mathcal{W}(0, 1/d)$.

Then for any $\boldsymbol{x}_1, \dots, \boldsymbol{x}_k$

$$\boldsymbol{Z}_i := \begin{pmatrix} \tilde{v}_i \sigma(\tilde{\boldsymbol{u}}_i^\top \boldsymbol{x}_1) \\ \vdots \\ \tilde{v}_i \sigma(\tilde{\boldsymbol{u}}_i^\top \boldsymbol{x}_k) \end{pmatrix} \in \mathbb{R}^k \qquad i = 1, \dots, n,$$

defines n centered i.i.d. vectors in $\mathbb{R}^k$. By the central limit theorem, see Theorem A.25,

$$\begin{pmatrix} \Phi(\boldsymbol{x}_1, \boldsymbol{w}_0) \\ \vdots \\ \Phi(\boldsymbol{x}_k, \boldsymbol{w}_0) \end{pmatrix} = \frac{1}{\sqrt{n}} \sum_{j=1}^n \boldsymbol{Z}_i$$

converges weakly to $\mathrm{N}(\boldsymbol{0}, \boldsymbol{C})$, where

$$C_{ij} = \mathbb{E}[\tilde{v}_1^2 \sigma(\tilde{\boldsymbol{u}}_1^\top \boldsymbol{x}_i)\sigma(\tilde{\boldsymbol{u}}_1^\top \boldsymbol{x}_j)] = \mathbb{E}[\sigma(\tilde{\boldsymbol{u}}_1^\top \boldsymbol{x}_i)\sigma(\tilde{\boldsymbol{u}}_1^\top \boldsymbol{x}_j)].$$

This concludes the proof. □

In the sense of Proposition 11.28, the network $\Phi(\boldsymbol{x}, \boldsymbol{w}_0)$ converges to a Gaussian process as the width n tends to infinity. Using the explicit dynamics of the linearized network outlined in Remark 11.24, one can show that the linearized network after training also corresponds to a Gaussian process (for some mean and covariance function depending on the data, the architecture, and the initialization). As the full and linearized models converge in the infinite width limit, we can infer that wide networks post-training resemble draws from a Gaussian process, see [131, Sec. 2.3.1] and [46].

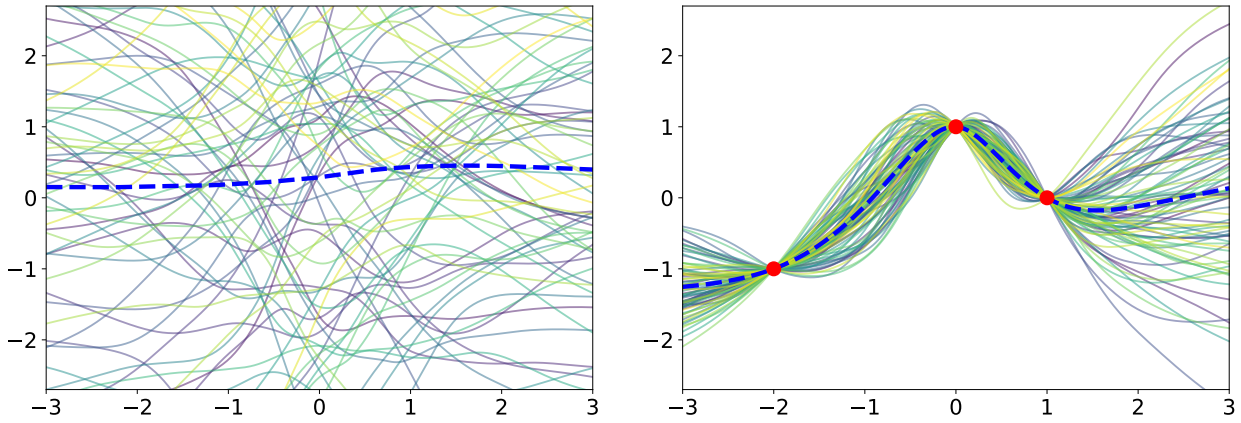
Rather than delving into the technical details of such statements, in Figure 11.3 we plot 80 different realizations of a neural network before and after training, i.e. the functions

$$\boldsymbol{x} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w}_0) \qquad \text{and} \qquad \boldsymbol{x} \mapsto \Phi(\boldsymbol{x}, \boldsymbol{w}_k). \tag{11.5.10}$$

We chose the architecture as (11.5.1) with activation function $\sigma = \arctan(x)$, width $n = 250$ and initialization

$$U_{0;ij} \overset{\text{iid}}{\sim} \mathrm{N}\Big(0, \frac{3}{d}\Big), \qquad v_{0;i} \overset{\text{iid}}{\sim} \mathrm{N}\Big(0, \frac{3}{n}\Big), \qquad b_{0;i},\ c_0 \overset{\text{iid}}{\sim} \mathrm{N}(0, 2). \tag{11.5.11}$$

The network was trained on a dataset of size $m = 3$ with $k = 1000$ steps of gradient descent and constant step size $h = 1/n$. Before training, the network's outputs resemble random draws from a Gaussian process with a constant zero mean function. Post-training, the outputs show minimal variance at the data points, since they essentially interpolate the data, cp. Remark 11.4 and (11.2.4). They exhibit increased variance further from these points, with the precise amount depending on the initialization variance chosen in (11.5.11).



(a) before training (b) after training

Figure 11.3: 80 realizations of a neural network at initialization (a) and after training on the red data points (b). The blue dashed line shows the mean. Figure based on [131, Fig. 2].

11.6 Normalized initialization

Consider the gradient $\nabla_{\boldsymbol{w}}\Phi(\boldsymbol{x}, \boldsymbol{w}_0)$ as in (11.5.2) with LeCun initialization. Since the components of $\boldsymbol{v}$ behave like $v_i \overset{\text{iid}}{\sim} \mathcal{W}(0, 1/n)$, it is easy to check that in terms of the width n

$$\begin{aligned}
\mathbb{E}[\|\nabla_{\boldsymbol{U}}\Phi(\boldsymbol{x}, \boldsymbol{w}_0)\|] &= \mathbb{E}[\|(\boldsymbol{v} \odot \sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}))\boldsymbol{x}^\top\|] &&= O(1)\\
\mathbb{E}[\|\nabla_{\boldsymbol{b}}\Phi(\boldsymbol{x}, \boldsymbol{w}_0)\|] &= \mathbb{E}[\|\boldsymbol{v} \odot \sigma'(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})\|] &&= O(1)\\
\mathbb{E}[\|\nabla_{\boldsymbol{v}}\Phi(\boldsymbol{x}, \boldsymbol{w}_0)\|] &= \mathbb{E}[\|\sigma(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})\|] &&= O(n)\\
\mathbb{E}[\|\nabla_{c}\Phi(\boldsymbol{x}, \boldsymbol{w}_0)\|] &= \mathbb{E}[|1|] &&= O(1).
\end{aligned}$$

As a result of this different scaling, gradient descent with step width $O(n^{-1})$ as in Theorem 11.23, will primarily train the weigths $\boldsymbol{v}$ in the output layer, and will barely move the remaining parameters $\boldsymbol{U}$, $\boldsymbol{b}$, and c. This is also reflected in the expression for the obtained kernel K^{LC} computed in Theorem 11.16, which corresponds to the contribution of the term $\langle \nabla_{\boldsymbol{v}}\Phi, \nabla_{\boldsymbol{v}}\Phi \rangle$.

Remark 11.29. For optimization methods such as ADAM, which scale each component of the gradient individually, the same does not hold in general.

LeCun initialization aims to normalize the variance of the output of all nodes at initialization (the forward dynamics). To also normalize the variance of the gradients (the backward dynamics), in this section we shortly dicuss a different architecture and initialization, consistent with the one used in the original NTK paper [106].

11.6.1 Architecture

Let $\Phi : \mathbb{R}^d \to \mathbb{R}$ be a depth-one neural network

$$\Phi(\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{\sqrt{n}} \boldsymbol{v}^\top \sigma\Big(\frac{1}{\sqrt{d}} \boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}\Big) + c, \tag{11.6.1}$$

with input $\boldsymbol{x} \in \mathbb{R}^d$ and parameters $\boldsymbol{U} \in \mathbb{R}^{n \times d}$, $\boldsymbol{v} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. We initialize the weights randomly according to $\boldsymbol{w}_0 = (\boldsymbol{U}_0, \boldsymbol{b}_0, \boldsymbol{v}_0, c_0)$ with parameters

$$U_{0;ij} \overset{\text{iid}}{\sim} \mathcal{W}(0,1), \qquad v_{0;i} \overset{\text{iid}}{\sim} \mathcal{W}(0,1), \qquad b_{0;i},\ c_0 = 0. \tag{11.6.2}$$

At initialization, (11.6.1), (11.6.2) is equivalent to (11.5.1), (11.5.3). However, for the gradient we obtain

$$\begin{aligned}
\nabla_{\boldsymbol{U}} \Phi(\boldsymbol{x}, \boldsymbol{w}) &= n^{-1/2} \Big(\boldsymbol{v} \odot \sigma'(d^{-1/2}\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})\Big) d^{-1/2} \boldsymbol{x}^\top \in \mathbb{R}^{n \times d} \\
\nabla_{\boldsymbol{b}} \Phi(\boldsymbol{x}, \boldsymbol{w}) &= n^{-1/2} \boldsymbol{v} \odot \sigma'\Big(d^{-1/2}\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b})\Big) \in \mathbb{R}^n \\
\nabla_{\boldsymbol{v}} \Phi(\boldsymbol{x}, \boldsymbol{w}) &= n^{-1/2} \sigma(d^{-1/2}\boldsymbol{U}\boldsymbol{x} + \boldsymbol{b}) \in \mathbb{R}^n \\
\nabla_{c} \Phi(\boldsymbol{x}, \boldsymbol{w}) &= 1 \in \mathbb{R}.
\end{aligned} \tag{11.6.3}$$

Contrary to (11.5.2), the three gradients with $O(n)$ entries are all scaled by the factor $n^{-1/2}$. This leads to a different training dynamics.

11.6.2 Neural tangent kernel

We compute again the neural tangent kernel. Unlike for LeCun initialization, there is no $1/n$ scaling required to obtain convergence of

$$\hat{K}_n(\boldsymbol{x}, \boldsymbol{z}) = \langle \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{x}, \boldsymbol{w}_0), \nabla_{\boldsymbol{w}} \Phi(\boldsymbol{z}, \boldsymbol{w}_0) \rangle$$

as $n \to \infty$. Here and in the following we consider the setting (11.6.1)–(11.6.2) for Φ and $\boldsymbol{w}_0$. Since this is also referred to as the NTK initialization, we denote the kernel by K^{NTK}. Due to the different training dynamics, we obtain additional terms in the NTK compared to Theorem 11.23.

Theorem 11.30. *Let $R < \infty$ such that $|\sigma(x)| \le R \cdot (1 + |x|)$ and $|\sigma'(x)| \le R \cdot (1 + |x|)$ for all $x \in \mathbb{R}$, and let $\mathcal{W}$ satisfy Assumption 11.14. For any $\boldsymbol{x}$, $\boldsymbol{z} \in \mathbb{R}^d$ and $u_i \overset{\text{iid}}{\sim} \mathcal{W}(0, 1/d)$, $i = 1, \dots, d$, it then holds*

$$\begin{aligned}
\lim_{n \to \infty} \hat{K}_n(\boldsymbol{x}, \boldsymbol{z}) &= \Big(1 + \frac{\boldsymbol{x}^\top \boldsymbol{z}}{d}\Big) \mathbb{E}[\sigma'(\boldsymbol{u}^\top \boldsymbol{x})^\top \sigma'(\boldsymbol{u}^\top \boldsymbol{z})] + \mathbb{E}[\sigma(\boldsymbol{u}^\top \boldsymbol{x})^\top \sigma(\boldsymbol{u}^\top \boldsymbol{z})] + 1 \\
&=: K^{\text{NTK}}(\boldsymbol{x}, \boldsymbol{z})
\end{aligned}$$

almost surely.

Proof. Denote $\boldsymbol{x}^{(1)} = \boldsymbol{U}_0\boldsymbol{x} + \boldsymbol{b}_0 \in \mathbb{R}^n$ and $\boldsymbol{z}^{(1)} = \boldsymbol{U}_0\boldsymbol{z} + \boldsymbol{b}_0 \in \mathbb{R}^n$. Due to the initialization (11.6.2) and our assumptions on $\mathcal{W}(0,1)$, the components

$$x_i^{(1)} = \sum_{j=1}^{d} U_{0;ij} x_j \sim \boldsymbol{u}^\top \boldsymbol{x} \qquad i = 1, \dots, n$$

are i.i.d. with finite pth moment (independent of n) for all $1 \leq p \leq 8$, and the same holds for the $(\sigma(x_i^{(1)}))_{i=1}^n$, $(\sigma'(x_i^{(1)}))_{i=1}^n$, $(\sigma(z_i^{(1)}))_{i=1}^n$, and $(\sigma'(z_i^{(1)}))_{i=1}^n$.

Then

$$\hat{K}_n(\boldsymbol{x}, \boldsymbol{z}) = \Big(1 + \frac{\boldsymbol{x}^\top \boldsymbol{z}}{d}\Big) \frac{1}{n} \sum_{i=1}^{n} v_i^2 \sigma'(x_i^{(1)}) \sigma'(z_i^{(1)}) + \frac{1}{n} \sum_{i=1}^{n} \sigma(x_i^{(1)}) \sigma(z_i^{(1)}) + 1.$$

By the law of large numbers and because $\mathbb{E}[v_i^2] = 1$, this converges almost surely to $K^{\mathrm{NTK}}(\boldsymbol{x}, \boldsymbol{z})$.

The existence of n_0 follows similarly by an application of Theorem A.23. □

Example 11.31 (K^{NTK} for ReLU)**.** Let $\sigma(x) = \max\{0, x\}$ and let $\mathcal{W}(0, 1/d)$ be the centered normal distribution on $\mathbb{R}$ with variance $1/d$. For $\boldsymbol{x}$, $\boldsymbol{z} \in \mathbb{R}^d$ holds by [37, Appendix A] (also see Exercise 11.36), that with $u_i \overset{\text{iid}}{\sim} \mathcal{W}(0, 1/d)$, $i = 1, \dots, d$,

$$\mathbb{E}[\sigma'(\boldsymbol{u}^\top \boldsymbol{x}) \sigma'(\boldsymbol{u}^\top \boldsymbol{z})] = \frac{\pi - \arccos\left(\frac{\boldsymbol{x}^\top \boldsymbol{z}}{\|\boldsymbol{x}\| \|\boldsymbol{z}\|}\right)}{2\pi}.$$

Together with Example 11.17, this yields an explizit formula for K^{NTK} in Theorem 11.30.

For this network architecture and under suitable assumptions on $\mathcal{W}$, similar arguments as in Section 11.5 can be used to show convergence of gradient descent to a global minimizer and proximity of the full to the linearized model. We refer to the literature in the bibliography section.

Bibliography and further reading

The discussion on linear and kernel regression in Sections 11.1 and 11.2 is quite standard, and can similarly be found in many textbooks. For more details on kernel methods we refer for instance to [42, 206]. The neural tangent kernel and its connection to the training dynamics was first investigated in [106] using an architecture similar to the one in Section 11.6. Since then, many works have extended this idea and presented differing perspectives on the topic, see for instance [2, 56, 5, 36]. Our presentation in Sections 11.4, 11.5, and 11.6 primarily follows [131] who also discussed the case of LeCun initialization. Especially for the main results in Theorem 11.13 and Theorem 11.23, we largely follow the arguments in this paper. The above references additionally treat the case of deep networks, which we have omitted here for simplicity. The explicit formula for the NTK of ReLU networks as presented in Examples 11.17 and 11.31 was given in [37]. The observation that neural networks at initialization behave like Gaussian processes discussed in Section 11.5.5 was first made in [157]. For a general reference on Gaussian processes see the textbook [188]. When only training the last layer of a network (in which the network is affine linear), there are strong links to random feature methods [186]. Recent developements on this topic can also be found in the literature under the name "Neural network Gaussian processes", or NNGPs for short [130, 47].

Exercises

Exercise 11.32. Prove Theorem 11.3.

Hint: Assume first that $\boldsymbol{w}_0 \in \ker(\boldsymbol{A})^\perp$ (i.e. $\boldsymbol{w}_0 \in \tilde{H}$). For $\operatorname{rank}(\boldsymbol{A}) < d$, using $\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - h\nabla f(\boldsymbol{w}_{k-1})$ and the singular value decomposition of $\boldsymbol{A}$, write down an explicit formula for $\boldsymbol{w}_k$. Observe that due to $1/(1-x) = \sum_{k\in\mathbb{N}_0} x^k$ for all $x \in (0,1)$ it holds $\boldsymbol{w}_k \to \boldsymbol{A}^\dagger \boldsymbol{y}$ as $k \to \infty$, where $\boldsymbol{A}^\dagger$ is the Moore-Penrose pseudoinverse of $\boldsymbol{A}$.

Exercise 11.33. Let $\boldsymbol{x}_i \in \mathbb{R}^d$, $i = 1, \dots, m$. Show that there exists a "feature map" $\phi : \mathbb{R}^d \to \mathbb{R}^m$, such that for any configuration of labels $y_i \in \{-1, 1\}$, there always exists a hyperplane in $\mathbb{R}^m$ separating the two sets $\{\phi(\boldsymbol{x}_i) \,|\, y_i = 1\}$ and $\{\phi(\boldsymbol{x}_i) \,|\, y_i = -1\}$.

Exercise 11.34. Consider the RBF kernel $K : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, $K(x, x') := \exp(-(x - x')^2)$. Find a Hilbert space H and a feature map $\phi : \mathbb{R} \to H$ such that $K(x, x') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_H$.

Exercise 11.35. Let $n \in \mathbb{N}$ and consider the polynomial kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, $K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^\top \boldsymbol{x}')^r$. Find a Hilbert space H and a feature map $\phi : \mathbb{R}^d \to H$, such that $K(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_H$.

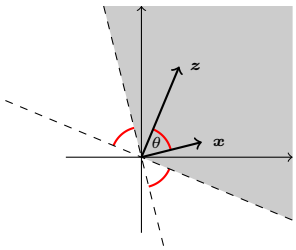
Hint: Use the multinomial formula.

Exercise 11.36. Let $u_i \overset{\text{iid}}{\sim} \mathrm{N}(0, 1)$ be i.i.d. standard Gaussian distributed random variables for $i = 1, \dots, d$. Show that for all nonzero $\boldsymbol{x}$, $\boldsymbol{z} \in \mathbb{R}^d$

$$\mathbb{E}[\mathbb{1}_{[0,\infty)}(\boldsymbol{u}^\top \boldsymbol{x}) \mathbb{1}_{[0,\infty)}(\boldsymbol{u}^\top \boldsymbol{z})] = \frac{\pi - \theta}{2\pi}, \qquad \theta = \arccos\Big(\frac{\boldsymbol{x}\boldsymbol{z}^\top}{\|\boldsymbol{x}\| \, \|\boldsymbol{z}\|}\Big).$$

This shows the formula for the ReLU NTK with Gaussian initialization as discussed in Example 11.31.

Hint: Consider the following sketch



Exercise 11.37. Consider the network (11.5.1) with LeCun initialization as in (11.5.3), but with the biases instead initialized as

$$c,\ b_i \overset{\text{iid}}{\sim} \mathcal{W}(0, 1) \qquad \text{for all } i = 1, \dots, n. \tag{11.6.4}$$

Compute the corresponding NTK as in Theorem 11.23. Moreover, compute the NTK also for the normalized network (11.6.1) with initialization (11.6.2) as in Theorem 11.30, but replace again the bias initialization with that given in (11.6.4).

Chapter 12

Loss landscape analysis

In Chapter 10, we saw how the weights of neural networks get adapted during training, using, e.g., variants of gradient descent. For certain cases, including the wide networks considered in Chapter 11, the corresponding iterative scheme converges to a global minimizer. In general, this is not guaranteed, and gradient descent can for instance get stuck in non-global minima or saddle points.

To get a better understanding of these situations, in this chapter we discuss the so-called loss landscape. This term refers to the graph of the empirical risk as a function of the weights. We give a more rigorous definition below, and first introduce notation for neural networks and their realizations for a fixed architecture.

Definition 12.1. Let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma \colon \mathbb{R} \to \mathbb{R}$ be an activation function, and let $B > 0$. We denote the set of neural networks Φ with L layers, layer widths $d_0, d_1, \dots, d_{L+1}$, all weights bounded in modulus by B, and using the activation function σ by $\mathcal{N}(\sigma; \mathcal{A}, B)$. Additionally, we define

$$\mathcal{PN}(\mathcal{A}, B) \coloneqq \bigtimes_{\ell=0}^{L} \left([-B, B]^{d_{\ell+1} \times d_\ell} \times [-B, B]^{d_{\ell+1}} \right),$$

and the **realization map**

$$\begin{aligned} R_\sigma \colon \mathcal{PN}(\mathcal{A}, B) &\to \mathcal{N}(\sigma; \mathcal{A}, B) \\ (\boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(\ell)})_{\ell=0}^{L} &\mapsto \Phi, \end{aligned} \tag{12.0.1}$$

where Φ is the neural network with weights and biases given by $(\boldsymbol{W}^{(\ell)}, \boldsymbol{b}^{(\ell)})_{\ell=0}^{L}$.

Throughout, we will identify $\mathcal{PN}(\mathcal{A}, B)$ with the cube $[-B, B]^{n_{\mathcal{A}}}$, where $n_{\mathcal{A}} \coloneqq \sum_{\ell=0}^{L} d_{\ell+1}(d_\ell + 1)$. Now we can introduce the loss landscape of a neural network architecture.

Definition 12.2. Let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma \colon \mathbb{R} \to \mathbb{R}$. Let $m \in \mathbb{N}$, and $S = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{m} \in (\mathbb{R}^{d_0} \times \mathbb{R}^{d_{L+1}})^m$ be a sample and let $\mathcal{L}$ be a loss function. Then, the **loss landscape**

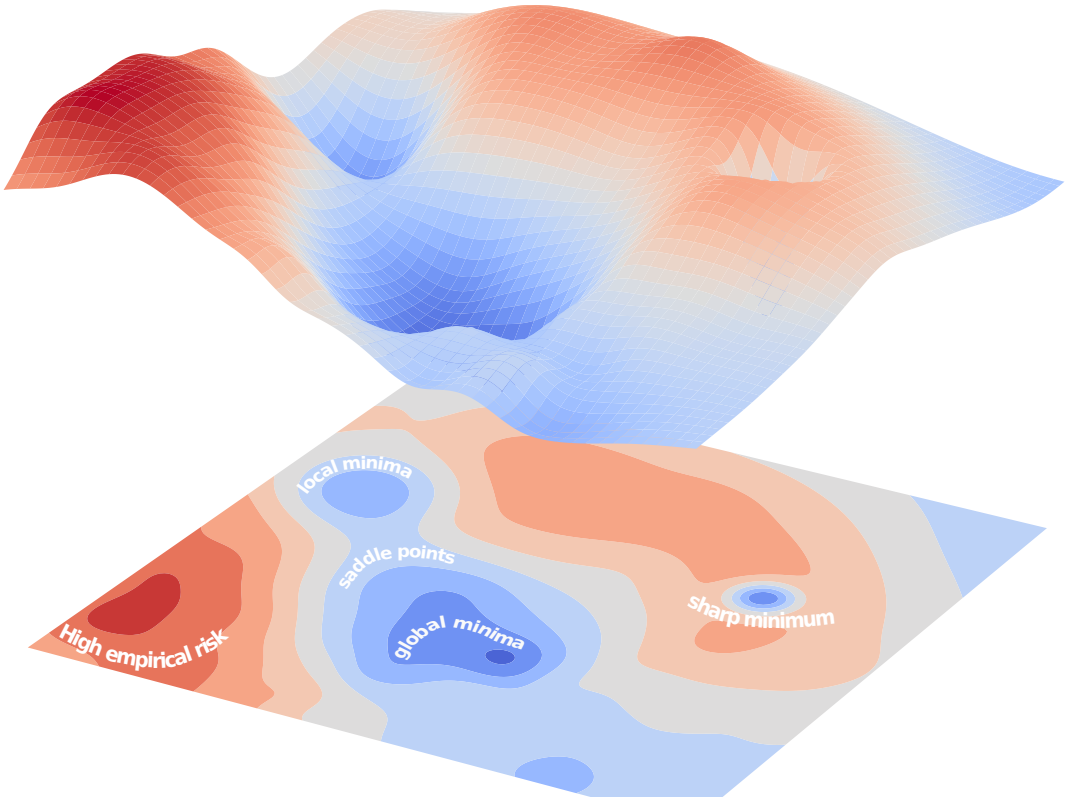


Figure 12.1: Two-dimensional section of a loss landscape. The loss landscape shows a spurious valley with local minima, global minima, as well as a region where saddle points appear. Moreover, a sharp minimum is shown.

is the graph of the function $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ defined as

$$\begin{aligned}\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}} : \mathcal{PN}(\mathcal{A};\infty) &\to \mathbb{R} \\ \theta &\mapsto \widehat{\mathcal{R}}_S(R_\sigma(\theta)).\end{aligned}$$

with $\widehat{\mathcal{R}}_S$ in (1.2.3) and R_σ in (12.0.1).

Identifying $\mathcal{PN}(\mathcal{A},\infty)$ with $\mathbb{R}^{n_\mathcal{A}}$, we can consider $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ as a map on $\mathbb{R}^{n_\mathcal{A}}$ and the loss landscape is a subset of $\mathbb{R}^{n_\mathcal{A}} \times \mathbb{R}$. The loss landscape is a high-dimensional surface, with hills and valleys. For visualization a two-dimensional section of a loss landscape is shown in Figure 12.1.

Questions of interest regarding the loss landscape include for example: How likely is it that we find local instead of global minima? Are these local minima typically sharp, having small volume, or are they part of large flat valleys that are difficult to escape? How bad is it to end up in a local minimum? Are most local minima as deep as the global minimum, or can they be significantly higher? How rough is the surface generally, and how do these characteristics depend on the network architecture? While providing complete answers to these questions is hard in general, in the rest of this chapter we give some intuition and mathematical insights for specific cases.

12.1 Visualization of loss landscapes

Visualizing loss landscapes can provide valuable insights into the effects of neural network depth, width, and activation functions. However, we can only visualize an at most two-dimensional surface embedded into three-dimensional space, whereas the loss landscape is a very high-dimensional object (unless the neural networks have only very few weights and biases).

To make the loss landscape accessible, we need to reduce its dimensionality. This can be achieved by evaluating the function $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ on a two-dimensional subspace of $\mathcal{PN}(\mathcal{A},\infty)$. Specifically, we choose three-parameters μ, θ_1, θ_2 and examine the function

$$\mathbb{R}^2 \ni (\alpha_1, \alpha_2) \mapsto \Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\mu + \alpha_1\theta_1 + \alpha_2\theta_2). \tag{12.1.1}$$

There are various natural choices for μ, θ_1, θ_2:

- *Random directions:* This was, for example used in [74, 102]. Here θ_1, θ_2 are chosen randomly, while μ is either a minimum of $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ or also chosen randomly. This simple approach can offer a quick insight into how rough the surface can be. However, as was pointed out in [134], random directions will very likely be orthogonal to the trajectory of the optimization procedure. Hence, they will likely miss the most relevant features.

- *Principal components of learning trajectory:* To address the shortcomings of random directions, another possibility is to determine μ, θ_1, θ_2, which best capture some given learning trajectory; For example, if $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(N)}$ are the parameters resulting from the training by SGD, we may determine μ, θ_1, θ_2 such that the hyperplane $\{\mu + \alpha_1\theta_1 + \alpha_2\theta_2 \,|\, \alpha_1, \alpha_2 \in \mathbb{R}\}$ minimizes the mean squared distance to the $\theta^{(j)}$ for $j \in \{1, \dots, N\}$. This is the approach of [134], and can be achieved by a principal component analysis.

- *Based on critical points:* For a more global perspective, μ, θ_1, θ_2 can be chosen to ensure the observation of multiple critical points. One way to achieve this is by running the optimization procedure three times with final parameters $\theta^{(1)}$, $\theta^{(2)}$, $\theta^{(3)}$. If the procedures have converged, then each of these parameters is close to a critical point of $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$. We can now set $\mu = \theta^{(1)}$, $\theta_1 = \theta^{(2)} - \mu$, $\theta_2 = \theta^{(3)} - \mu$. This then guarantees that (12.1.1) passes through or at least comes very close to three critical points (at $(\alpha_1, \alpha_2) = (0,0), (0,1), (1,0)$). We present six visualizations of this form in Figure 12.2.

Figure 12.2 gives some interesting insight into the effect of depth and width on the shape of the loss landscape. For very wide and shallow neural networks, we have the widest minima, which, in the case of the tanh activation function also seem to belong to the same valley. With increasing depth and smaller width the minima get steeper and more disconnected.

12.2 Spurious valleys

From the perspective of optimization, the ideal loss landscape has one global minimum in the center of a large valley, so that gradient descent converges towards the minimum irrespective of the chosen initialization.

This situation is not realistic for deep neural networks. Indeed, for a simple shallow neural network

$$\mathbb{R}^d \ni \boldsymbol{x} \mapsto \Phi(\boldsymbol{x}) = \boldsymbol{W}^{(1)}\sigma(\boldsymbol{W}^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(0)}) + \boldsymbol{b}^{(1)},$$

it is clear that for every permutation matrix $\boldsymbol{P}$

$$\Phi(\boldsymbol{x}) = \boldsymbol{W}^{(1)}\boldsymbol{P}^T\sigma(\boldsymbol{P}\boldsymbol{W}^{(0)}\boldsymbol{x} + \boldsymbol{P}\boldsymbol{b}^{(0)}) + \boldsymbol{b}^{(1)} \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^d.$$

Hence, in general there exist multiple parameterizations realizing the same output function. Moreover, if at least one global minimum with non-permutation-invariant weights exists, then there are more than one global minima of the loss landscape.

This is not problematic; in fact, having many global minima is beneficial. The larger issue is the existence of non-global minima. Following [235], we start by generalizing the notion of non-global minima to *spurious valleys*.

Definition 12.3. Let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$ and $\sigma\colon \mathbb{R} \to \mathbb{R}$. Let $m \in \mathbb{N}$, and $S = (\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^m \in (\mathbb{R}^{d_0} \times \mathbb{R}^{d_{L+1}})^m$ be a sample and let $\mathcal{L}$ be a loss function. For $c \in \mathbb{R}$, we define the sub-level set of $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ as

$$\Omega_\Lambda(c) \coloneqq \{\theta \in \mathcal{PN}(\mathcal{A}, \infty) \,|\, \Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\theta) \leq c\}.$$

A path-connected component of $\Omega_\Lambda(c)$, which does not contain a global minimum of $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ is called a **spurious valley**.

The next proposition shows that spurious local minima do not exist for shallow overparameterized neural networks, i.e., for neural networks that have at least as many parameters in the hidden layer as there are training samples.

Proposition 12.4. *Let $\mathcal{A} = (d_0, d_1, 1) \in \mathbb{N}^3$ and let $S = (\boldsymbol{x}_i, y_i)_{i=1}^m \in (\mathbb{R}^{d_0} \times \mathbb{R})^m$ be a sample such that $m \leq d_1$. Furthermore, let $\sigma \in \mathcal{M}$ be not a polynomial, and let $\mathcal{L}$ be a convex loss function. Further assume that $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ has at least one global minimum. Then, $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$, has no spurious valleys.*

Proof. Let $\theta_a, \theta_b \in \mathcal{PN}(\mathcal{A}, \infty)$ with $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\theta_a) > \Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\theta_b)$. Then we will show below that there is another parameter θ_c such that

- $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\theta_b) = \Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\theta_c)$
- there is a continuous path $\alpha : [0,1] \to \mathcal{PN}(\mathcal{A}, \infty)$ such that $\alpha(0) = \theta_a$, $\alpha(1) = \theta_c$, and $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\alpha)$ is monotonically decreasing.

By Exercise 12.7, the construction above rules out the existence of spurious valleys by choosing θ_a an element of a spurious valley and θ_b a global minimum.

Next, we present the construction: Let us denote

$$\theta_o = \left(\left(\boldsymbol{W}_o^{(\ell)}, \boldsymbol{b}_o^{(\ell)}\right)_{\ell=0}^{1}\right) \qquad \text{for } o \in \{a, b, c\}.$$

Moreover, for $j = 1, \dots, d_1$, we introduce $\boldsymbol{v}_o^j \in \mathbb{R}^m$ defined as

$$(\boldsymbol{v}_o^j)_i = \left(\sigma\left(\boldsymbol{W}_o^{(0)}\boldsymbol{x}_i + \boldsymbol{b}_o^{(0)}\right)\right)_j \qquad \text{for } i = 1, \dots, m.$$

Notice that, if we set $\boldsymbol{V}_o = ((\boldsymbol{v}_o^j)^\top)_{j=1}^{d_1}$, then

$$\boldsymbol{W}_o^{(1)}\boldsymbol{V}_o = \left(R_\sigma(\theta_o)(\boldsymbol{x}_i) - \boldsymbol{b}_o^{(1)}\right)_{i=1}^m, \tag{12.2.1}$$

where the right-hand side is considered a row-vector.

We will now distinguish between two cases. For the first the result is trivial and the second can be transformed into the first one.

Case 1: Assume that $\boldsymbol{V}_a$ has rank m. In this case, it is obvious from (12.2.1), that there exists $\widetilde{\boldsymbol{W}}$ such that

$$\widetilde{\boldsymbol{W}}\boldsymbol{V}_a = \left(R_\sigma(\theta_b)(\boldsymbol{x}_i) - \boldsymbol{b}_a^{(1)}\right)_{i=1}^m.$$

We can thus set $\alpha(t) = ((\boldsymbol{W}_a^{(0)}, \boldsymbol{b}_a^{(0)}), ((1-t)\boldsymbol{W}_a^{(1)} + t\widetilde{\boldsymbol{W}}, \boldsymbol{b}_a^{(1)}))$.

Note that by construction $\alpha(0) = \theta_a$ and $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\alpha(1)) = \Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\theta_b)$. Moreover, $t \mapsto (R_\sigma(\alpha(t))(\boldsymbol{x}_i))_{i=1}^m$ describes a straight path in $\mathbb{R}^m$ and hence, by the convexity of $\mathcal{L}$ it is clear that $t \mapsto \Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\alpha(t))$ is monotonically decreasing.

Case 2: Assume that V_a has rank less than m. In this case, we show that we find a continuous path from θ_a to another neural network parameter with higher rank. The path will be such that $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ is monotonically decreasing.

Under the assumptions, we have that one $\boldsymbol{v}_a^j$ can be written as a linear combination of the remaining $\boldsymbol{v}_a^i$, $i \neq j$. Without loss of generality, we assume $j = 1$. Then, there exist $(\alpha_i)_{i=2}^m$ such that

$$\boldsymbol{v}_a^1 = \sum_{i=2}^m \alpha_i \boldsymbol{v}_a^i. \tag{12.2.2}$$

Next, we observe that there exists $\boldsymbol{v}^* \in \mathbb{R}^m$ which is linearly independent from all $(\boldsymbol{v}_a^j)_{i=1}^m$ and can be written as $(\boldsymbol{v}^*)_i = \sigma((\boldsymbol{w}^*)^\top \boldsymbol{x}_i + b^*)$ for some $\boldsymbol{w}^* \in \mathbb{R}^{d_0}, b^* \in \mathbb{R}$. Indeed, if we assume that such $\boldsymbol{v}^*$ does not exist, then it follows that $\operatorname{span}\{(\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b))_{i=1}^m \,|\, \boldsymbol{w} \in \mathbb{R}^{d_0}, b \in \mathbb{R}\}$ is an $m-1$ dimensional subspace of $\mathbb{R}^m$ which yields a contradiction to Theorem 9.3.

Now, we define two paths: First,

$$\alpha_1(t) = ((\boldsymbol{W}_a^{(0)}, \boldsymbol{b}_a^{(0)}), (\boldsymbol{W}_a^{(1)}(t), \boldsymbol{b}_a^{(1)})), \qquad \text{for } t \in [0, 1/2]$$

where

$$(\boldsymbol{W}_a^{(1)}(t))_1 = (1-2t)(\boldsymbol{W}_a^{(1)})_1 \qquad \text{and} \qquad (\boldsymbol{W}_a^{(1)}(t))_i = (\boldsymbol{W}_a^{(1)})_i + 2t\alpha_i(\boldsymbol{W}_a^{(1)})_1$$

for $i = 2, \dots, d_1$, for $t \in [0, 1/2]$. Second,

$$\alpha_2(t) = ((\boldsymbol{W}_a^{(0)}(t), \boldsymbol{b}_a^{(0)}(t)), (\boldsymbol{W}_a^{(1)}(1/2), \boldsymbol{b}_a^{(1)})), \text{ for } t \in (1/2, 1],$$

where

$$(\boldsymbol{W}_a^{(0)}(t))_1 = 2(t-1/2)(\boldsymbol{W}_a^{(0)})_1 + (2t-1)\boldsymbol{w}^* \qquad \text{and} \qquad (\boldsymbol{W}_a^{(0)}(t))_i = (\boldsymbol{W}_a^{(0)})_i$$

for $i = 2, \dots, d_1$, $(\boldsymbol{b}_a^{(0)}(t))_1 = 2(t-1/2)(\boldsymbol{b}_a^{(0)})_1 + (2t-1)b^*$, and $(\boldsymbol{b}_a^{(0)}(t))_i = (\boldsymbol{b}_a^{(0)})_i$ for $i = 2, \dots, d_1$.

It is clear by (12.2.2) that $(R_\sigma(\alpha_1)(\boldsymbol{x}_i))_{i=1}^m$ is constant. Moreover, $R_\sigma(\alpha_2)(\boldsymbol{x})$ is constant for all $\boldsymbol{x} \in \mathbb{R}^{d_0}$. In addition, by construction for

$$\bar{\boldsymbol{v}}^j \coloneqq \left(\left(\sigma \left(\boldsymbol{W}_a^{(0)}(1)\boldsymbol{x}_i + \boldsymbol{b}_a^{(0)}(1) \right) \right)_j \right)_{i=1}^m$$

it holds that $((\bar{\boldsymbol{v}}^j)^\top)_{j=1}^{d_1}$ has rank larger than that of $\boldsymbol{V}_a$. Concatenating α_1 and α_2 now yields a continuous path from θ_a to another neural network parameter with higher associated rank such that $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}$ is monotonically decreasing along the path. Iterating this construction, we can find a path to a neural network parameter where the associated matrix has full rank. This reduces the problem to Case 1. □

12.3 Saddle points

Saddle points are critical points of the loss landscape at which the loss decreases in one direction. In this sense, saddle points are not as problematic as local minima or spurious valleys if the updates in the learning iteration have some stochasticity. Eventually, a random step in the right direction could be taken and the saddle point can be escaped.

If most of the critical points are saddle points, then, even though the loss landscape is challenging for optimization, one still has a good chance of eventually reaching the global minimum. Saddle points of the loss landscape were studied in [45, 172] and we will review some of the findings in a simplified way below. The main observation in [172] is that, under some quite strong assumptions, it holds that *critical points in the loss landscape associated to a large loss are typically saddle points, whereas those associated to small loss correspond to minima.* This situation is encouraging for the prospects of optimization in deep learning, since, even if we get stuck in a local minimum, it will very likely be such that the loss is close to optimal.

The results of [172] use random matrix theory, which we do not recall here. Moreover, it is hard to gauge if the assumptions made are satisfied for a specific problem. Nonetheless, we recall the main idea, which provides some intuition to support the above claim.

Let $\mathcal{A} = (d_0, d_1, 1) \in \mathbb{N}^3$. Then, for a neural network parameter $\theta \in \mathcal{PN}(\mathcal{A}, \infty)$ and activation function σ, we set $\Phi_\theta \coloneqq R_\sigma(\theta)$ and define for a sample $S = (\boldsymbol{x}_i, y_i)_{i=1}^m$ the errors

$$e_i = \Phi_\theta(\boldsymbol{x}_i) - y_i \qquad \text{for } i = 1, \dots, m.$$

If we use the square loss, then

$$\widehat{\mathcal{R}_S}(\Phi_\theta) = \frac{1}{m} \sum_{i=1}^m e_i^2. \tag{12.3.1}$$

Next, we study the Hessian of $\widehat{\mathcal{R}}_S(\Phi_\theta)$.

Proposition 12.5. *Let $\mathcal{A} = (d_0, d_1, 1)$ and $\sigma : \mathbb{R} \to \mathbb{R}$. Then, for every $\theta \in \mathcal{PN}(\mathcal{A}, \infty)$ where $\widehat{\mathcal{R}}_S(\Phi_\theta)$ in (12.3.1) is twice continuously differentiable with respect to the weights, it holds that*

$$\boldsymbol{H}(\theta) = \boldsymbol{H}_0(\theta) + \boldsymbol{H}_1(\theta),$$

where $\boldsymbol{H}(\theta)$ is the Hessian of $\widehat{\mathcal{R}}_S(\Phi_\theta)$ at θ, $\boldsymbol{H}_0(\theta)$ is a positive semi-definite matrix which is independent from $(y_i)_{i=1}^m$, and $\boldsymbol{H}_1(\theta)$ is a symmetric matrix that for fixed θ and $(\boldsymbol{x}_i)_{i=1}^m$ depends linearly on $(e_i)_{i=1}^m$.

Proof. Using the identification introduced after Definition 12.2, we can consider θ a vector in $\mathbb{R}^{n_\mathcal{A}}$. For $k = 1, \dots, n_\mathcal{A}$, we have that

$$\frac{\partial \widehat{\mathcal{R}}_S(\Phi_\theta)}{\partial \theta_k} = \frac{2}{m} \sum_{i=1}^m e_i \frac{\partial \Phi_\theta(\boldsymbol{x}_i)}{\partial \theta_k}.$$

Therefore, for $j = 1, \dots, n_\mathcal{A}$, we have, by the Leibniz rule, that

$$\begin{aligned} \frac{\partial^2 \widehat{\mathcal{R}}_S(\Phi_\theta)}{\partial \theta_j \partial \theta_k} &= \frac{2}{m} \sum_{i=1}^m \left(\frac{\partial \Phi_\theta(\boldsymbol{x}_i)}{\partial \theta_j} \frac{\partial \Phi_\theta(\boldsymbol{x}_i)}{\partial \theta_k} \right) + \frac{2}{m} \left(\sum_{i=1}^m e_i \frac{\partial^2 \Phi_\theta(\boldsymbol{x}_i)}{\partial \theta_j \partial \theta_k} \right) \\ &=: \boldsymbol{H}_0(\theta) + \boldsymbol{H}_1(\theta). \end{aligned} \tag{12.3.2}$$

It remains to show that $\boldsymbol{H}_0(\theta)$ and $\boldsymbol{H}_1(\theta)$ have the asserted properties. Note that, setting

$$J_{i,\theta} = \begin{pmatrix} \frac{\partial \Phi_\theta(\boldsymbol{x}_i)}{\partial \theta_1} \\ \vdots \\ \frac{\partial \Phi_\theta(\boldsymbol{x}_i)}{\partial \theta_{n_\mathcal{A}}} \end{pmatrix} \in \mathbb{R}^{n_\mathcal{A}},$$

we have that $\boldsymbol{H}_0(\theta) = \frac{2}{m} \sum_{i=1}^m J_{i,\theta} J_{i,\theta}^\top$ and hence $\boldsymbol{H}_0(\theta)$ is a sum of positive semi-definite matrices, which shows that $\boldsymbol{H}_0(\theta)$ is positive semi-definite.

The symmetry of $\boldsymbol{H}_1(\theta)$ follows directly from the symmetry of second derivatives which holds since we assumed twice continuous differentiability at θ. The linearity of $\boldsymbol{H}_1(\theta)$ in $(e_i)_{i=1}^m$ is clear from (12.3.2). □

How does Proposition 12.5 imply the claimed relationship between the size of the loss and the prevalence of saddle points?

Let θ correspond to a critical point. If $\boldsymbol{H}(\theta)$ has at least one negative eigenvalue, then θ cannot be a minimum, but instead must be either a saddle point or a maximum. While we do not know anything about $\boldsymbol{H}_1(\theta)$ other than that it is symmetric, it is not unreasonable to assume that it has a negative eigenvalue especially if $n_\mathcal{A}$ is very large. With this consideration, let us consider the following model:

Fix a parameter θ. Let $S^0 = (\boldsymbol{x}_i, y_i^0)_{i=1}^m$ be a sample and $(e_i^0)_{i=1}^m$ be the associated errors. Further let $\boldsymbol{H}^0(\theta), \boldsymbol{H}_0^0(\theta), \boldsymbol{H}_1^0(\theta)$ be the matrices according to Proposition 12.5.

Further let for $\lambda > 0$, $S^\lambda = (\boldsymbol{x}_i, y_i^\lambda)_{i=1}^m$ be such that the associated errors are $(e_i)_{i=1}^m = \lambda (e_i^0)_{i=1}^m$. The Hessian of $\widehat{\mathcal{R}}_{S^\lambda}(\Phi_\theta)$ at θ is then $\boldsymbol{H}^\lambda(\theta)$ satisfying

$$\boldsymbol{H}^\lambda(\theta) = \boldsymbol{H}_0^0(\theta) + \lambda \boldsymbol{H}_1^0(\theta).$$

Hence, if λ is large, then $\boldsymbol{H}^\lambda(\theta)$ is perturbation of an amplified version of $\boldsymbol{H}_1^0(\theta)$. Clearly, if $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{H}_1(\theta)$ with negative eigenvalue $-\mu$, then

$$\boldsymbol{v}^\top \boldsymbol{H}^\lambda(\theta) \boldsymbol{v} \leq (\|\boldsymbol{H}_0^0(\theta)\| - \lambda \mu) \|\boldsymbol{v}\|^2,$$

which we can expect to be negative for large λ. Thus, $\boldsymbol{H}^{\lambda}(\theta)$ has a negative eigenvalue for large λ.

On the other hand, if λ is small, then $\boldsymbol{H}^{\lambda}(\theta)$ is merely a perturbation of $\boldsymbol{H}_0^0(\theta)$ and we can expect its spectrum to resemble that of $\boldsymbol{H}_0^0$ more and more.

What we see is that, the same parameter, is more likely to be a saddle point for a sample that produces a high empirical risk than for a sample with small risk. Note that, since $\boldsymbol{H}_0^0(\theta)$ was only shown to be *semi*-definite the argument above does not rule out saddle points even for very small λ. But it does show that for small λ, every negative eigenvalue would be very small.

A more refined analysis where we compare different parameters but for the same sample and quantify the likelihood of local minima versus saddle points requires the introduction of a probability distribution on the weights. We refer to [172] for the details.

Bibliography and further reading

The results on visualization of the loss landscape are inspired by [134, 74, 102]. Results on the non-existence of spurious valleys can be found in [235] with similar results in [184]. In [39] the loss landscape was studied by linking it to so-called spin-glass models. There it was found that under strong assumptions critical points associated to lower losses are more likely to be minima than saddle points. In [172], random matrix theory is used to provide similar results, that go beyond those established in Section 12.3. On the topic of saddle points, [45] identifies the existence of saddle points as more problematic than that of local minima, and an alternative saddle-point aware optimization algorithm is introduced.

Two essential topics associated to the loss landscape that have not been discussed in this chapter are mode connectivity and the sharpness of minima. Mode connectivity, roughly speaking describes the phenomenon, that local minima found by SGD over deep neural networks are often connected by simple curves of equally low loss [64, 54]. Moreover, the sharpness of minima has been analyzed and linked to generalization capabilities of neural networks, with the idea being that wide neural networks are easier to find and also yield robust neural networks [92, 34, 247]. However, this does not appear to exclude sharp minima from generalizing well [53].

Exercises

Exercise 12.6. In view of Definition 12.3, show that a local minimum of a differentiable function is contained in a spurious valley.

Exercise 12.7. Show that if there exists a continuous path α between a parameter θ_1 and a global minimum θ_2 such that $\Lambda_{\mathcal{A},\sigma,S,\mathcal{L}}(\alpha)$ is monotonically decreasing, then θ_1 cannot be an element of a spurious valley.

Exercise 12.8. Find an example of a spurious valley for a simple architecture.

Hint: Use a single neuron ReLU neural network and observe that, for two networks one with positive and one with negative slope, every continuous path in parameter space that connects the two has to pass through a parameter corresponding to a constant function.

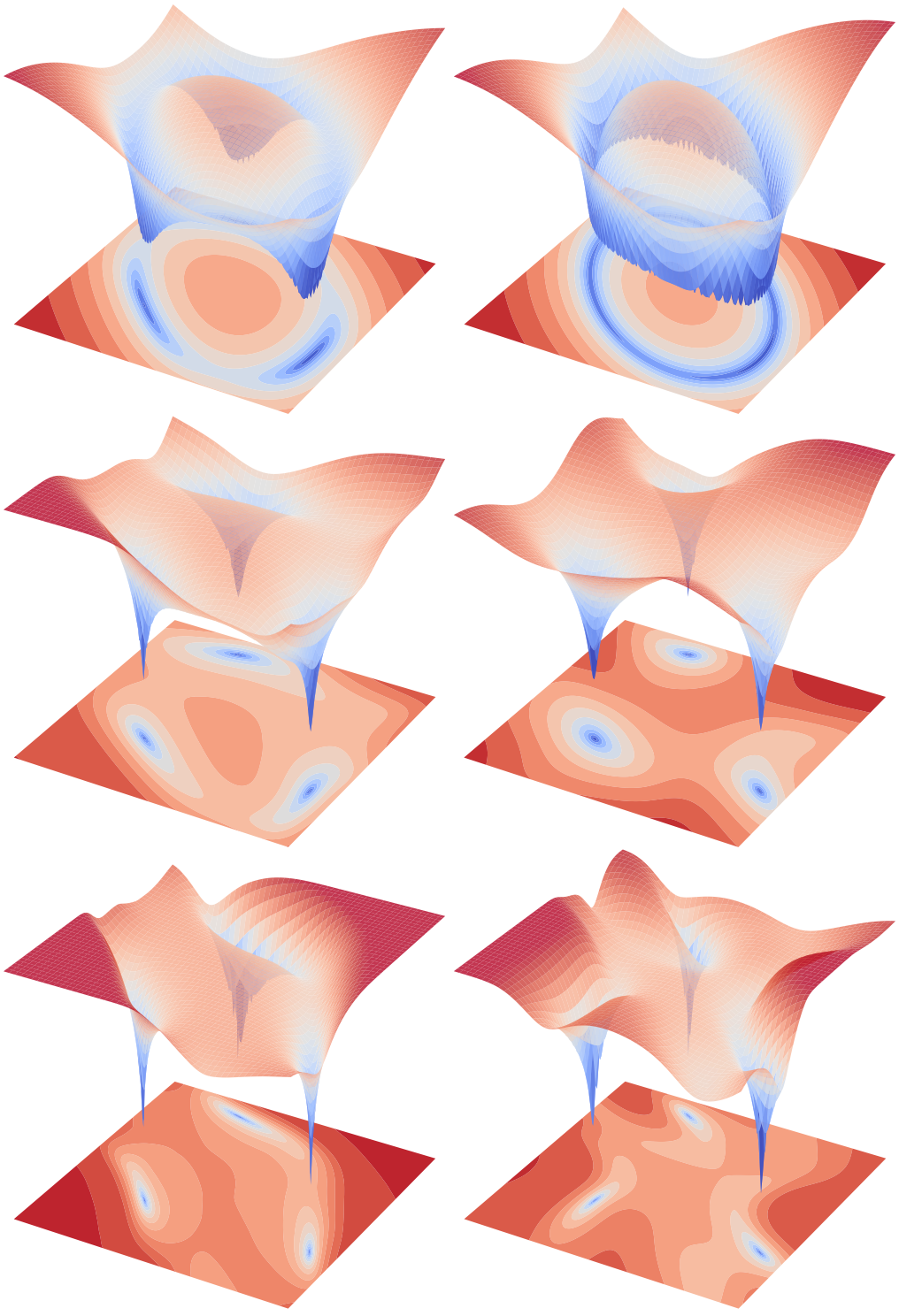


Figure 12.2: A collection of loss landscapes. In the left column are neural networks with ReLU activation function, the right column shows loss landscapes of neural networks with the hyperbolic tangent activation function. All neural networks have five dimensional input, and one dimensional output. Moreover, from top to bottom the hidden layers have sizes 1000, 20, 10, and the number of layers are 1, 4, 7.

Chapter 13

Shape of neural network spaces

As we have seen in the previous chapter, the loss landscape of neural networks can be quite intricate and is typically not convex. In some sense, the reason for this is that we take the point of view of a map from the parameterization of a neural network. Let us consider a convex loss function $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and a sample $S = (\boldsymbol{x}_i, y_i)_{i=1}^m \in (\mathbb{R}^d \times \mathbb{R})^m$.

Then, for two neural networks Φ_1, Φ_2 and for $\alpha \in (0, 1)$ it holds that

$$\begin{aligned}\widehat{\mathcal{R}}_S(\alpha\Phi_1 + (1-\alpha)\Phi_2) &= \frac{1}{m}\sum_{i=1}^m \mathcal{L}(\alpha\Phi_1(\boldsymbol{x}_i) + (1-\alpha)\Phi_2(\boldsymbol{x}_i), y_i) \\ &\leq \frac{1}{m}\sum_{i=1}^m \alpha\mathcal{L}(\Phi_1(\boldsymbol{x}_i), y_i) + (1-\alpha)\mathcal{L}(\Phi_2(\boldsymbol{x}_i), y_i) \\ &= \alpha\widehat{\mathcal{R}}_S(\Phi_1) + (1-\alpha)\widehat{\mathcal{R}}_S(\Phi_2).\end{aligned}$$

Hence, the empirical risk is convex when considered as a map depending on the neural network functions rather then the neural network parameters. A convex function does not have spurious minima or saddle points. As a result, the issues from the previous section are avoided if we take the perspective of neural network sets.

So why do we not optimize over the sets of neural networks instead of the parameters? To understand this, we will now study the set of neural networks associated with a fixed architecture as a subset of other function spaces.

We start by investigating the realization map R_σ introduced in Definition 12.1. Concretely, we show in Section 13.1, that if σ is Lipschitz, then the set of neural networks is the image of $\mathcal{PN}(\mathcal{A}, \infty)$ under a locally Lipschitz map. We will use this fact to show in Section 13.2 that sets of neural networks are typically non-convex, and even have arbitrarily large holes. Finally, in Section 13.3, we study the extent to which there exist best approximations to arbitrary functions, in the set of neural networks. We will demonstrate that the lack of best approximations causes the weights of neural networks to grow infinitely during training.

13.1 Lipschitz parameterizations

In this section, we study the realization map R_σ. The main result is the following simplified version of [173, Proposition 4].

Proposition 13.1. *Let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma\colon \mathbb{R} \to \mathbb{R}$ be C_σ-Lipschitz continuous with $C_\sigma \geq 1$, let $|\sigma(x)| \leq C_\sigma |x|$ for all $x \in \mathbb{R}$, and let $B \geq 1$.*

Then, for all $\theta, \theta' \in \mathcal{PN}(\mathcal{A}, B)$,

$$\|R_\sigma(\theta) - R_\sigma(\theta')\|_{L^\infty([-1,1]^{d_0})} \leq (2C_\sigma B d_{\max})^L n_{\mathcal{A}} \|\theta - \theta'\|_\infty,$$

where $d_{\max} = \max_{\ell=0,\dots,L+1} d_\ell$ and $n_{\mathcal{A}} = \sum_{\ell=0}^{L} d_{\ell+1}(d_\ell + 1)$.

Proof. Let $\theta, \theta' \in \mathcal{PN}(\mathcal{A}, B)$ and define $\delta \coloneqq \|\theta - \theta'\|_\infty$. Repeatedly using the triangle inequality we find a sequence $(\theta_j)_{j=0}^{n_{\mathcal{A}}}$ such that $\theta_0 = \theta$, $\theta_{n_{\mathcal{A}}} = \theta'$, $\|\theta_j - \theta_{j+1}\|_\infty \leq \delta$, and θ_j and θ_{j+1} differ in one entry only for all $j = 0, \dots n_{\mathcal{A}} - 1$. We conclude that for all $\boldsymbol{x} \in [-1,1]^{d_0}$

$$\|R_\sigma(\theta)(\boldsymbol{x}) - R_\sigma(\theta')(\boldsymbol{x})\|_\infty \leq \sum_{j=0}^{n_{\mathcal{A}}-1} \|R_\sigma(\theta_j)(\boldsymbol{x}) - R_\sigma(\theta_{j+1})(\boldsymbol{x})\|_\infty. \tag{13.1.1}$$

To upper bound (13.1.1), we now only need to understand the effect of changing one weight in a neural network by δ.

Before we can complete the proof we need two auxiliary lemmas. The first of which holds under slightly weaker assumptions of Proposition 13.1.

Lemma 13.2. *Under the assumptions of Proposition 13.1, but with B being allowed to be arbitrary positive, it holds for all $\Phi \in \mathcal{N}(\sigma; \mathcal{A}, B)$*

$$\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_\infty \leq C_\sigma^L \cdot (B d_{\max})^{L+1} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \tag{13.1.2}$$

for all $\boldsymbol{x}$, $\boldsymbol{x}' \in \mathbb{R}^{d_0}$.

Proof. We start with the case, where $L = 1$. Then, for $(d_0, d_1, d_2) = \mathcal{A}$, we have that

$$\Phi(\boldsymbol{x}) = \mathbf{W}^{(1)} \sigma(\mathbf{W}^{(0)} \boldsymbol{x} + \mathbf{b}^{(0)}) + \mathbf{b}^{(1)},$$

for certain $\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \mathbf{b}^{(0)}, \mathbf{b}^{(1)}$ with all entries bounded by B. As a consequence, we can estimate

$$\begin{aligned}
\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_\infty &= \left\| \mathbf{W}^{(1)} \left(\sigma(\mathbf{W}^{(0)} \boldsymbol{x} + \mathbf{b}^{(0)}) - \sigma(\mathbf{W}^{(0)} \boldsymbol{x}' + \mathbf{b}^{(0)}) \right) \right\|_\infty \\
&\leq d_1 B \left\| \sigma(\mathbf{W}^{(0)} \boldsymbol{x} + \mathbf{b}^{(0)}) - \sigma(\mathbf{W}^{(0)} \boldsymbol{x}' + \mathbf{b}^{(0)}) \right\|_\infty \\
&\leq d_1 B C_\sigma \left\| \mathbf{W}^{(0)} (\boldsymbol{x} - \boldsymbol{x}') \right\|_\infty \\
&\leq d_1 d_0 B^2 C_\sigma \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|_\infty \leq C_\sigma \cdot (d_{\max} B)^2 \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|_\infty,
\end{aligned}$$

where we used the Lipschitz property of σ and the fact that $\|\mathbf{A}\boldsymbol{x}\|_\infty \leq n \max_{i,j} |A_{ij}| \|\boldsymbol{x}\|_\infty$ for every matrix $\boldsymbol{A} = (A_{ij})_{i=1,j=1}^{m,n} \in \mathbb{R}^{m \times n}$.

The induction step from L to $L+1$ follows similarly. This concludes the proof of the lemma. □

Lemma 13.3. *Under the assumptions of Proposition 13.1 it holds that*

$$\|\boldsymbol{x}^{(\ell)}\|_\infty \leq (2C_\sigma B d_{\max})^\ell \quad \text{for all } \boldsymbol{x} \in [-1,1]^{d_0}. \tag{13.1.3}$$

Proof. Per Definitions (2.1.1b) and (2.1.1c), we have that for $\ell = 1, \dots, L+1$

$$\begin{aligned}\|\boldsymbol{x}^{(\ell)}\|_\infty &\leq C_\sigma \left\| \mathbf{W}^{(\ell-1)}\boldsymbol{x}^{(\ell-1)} + \mathbf{b}^{(\ell-1)} \right\|_\infty \\ &\leq C_\sigma B d_{\max} \|\boldsymbol{x}^{(\ell-1)}\|_\infty + BC_\sigma,\end{aligned}$$

where we used the triangle inequality and the estimate $\|\mathbf{A}\boldsymbol{x}\|_\infty \leq n \max_{i,j} |A_{ij}| \|\boldsymbol{x}\|_\infty$, which holds for every matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$. We obtain that

$$\begin{aligned}\|\boldsymbol{x}^{(\ell)}\|_\infty &\leq C_\sigma B d_{\max} \cdot (1 + \|\boldsymbol{x}^{(\ell-1)}\|_\infty) \\ &\leq 2C_\sigma B d_{\max} \cdot (\max\{1, \|\boldsymbol{x}^{(\ell-1)}\|_\infty\}).\end{aligned}$$

Resolving the recursive estimate of $\|\boldsymbol{x}^{(\ell)}\|_\infty$ by $2C_\sigma B d_{\max}(\max\{1, \|\boldsymbol{x}^{(\ell-1)}\|_\infty\})$, we conclude that

$$\|\boldsymbol{x}^{(\ell)}\|_\infty \leq (2C_\sigma B d_{\max})^\ell \max\{1, \|\boldsymbol{x}^{(0)}\|_\infty\} = (2C_\sigma B d_{\max})^\ell.$$

This concludes the proof of the lemma. □

We can now proceed with the proof of Proposition 13.1. Assume that θ_{j+1} and θ_j differ only in one entry. We assume this entry to be in the ℓth layer, and we start with the case $\ell < L$. It holds

$$|R_\sigma(\theta_j)(\boldsymbol{x}) - R_\sigma(\theta_{j+1})(\boldsymbol{x})| = |\Phi^\ell(\sigma(\mathbf{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + \mathbf{b}^{(\ell)})) - \Phi^\ell(\sigma(\overline{\mathbf{W}^{(\ell)}}\boldsymbol{x}^{(\ell)} + \overline{\mathbf{b}^{(\ell)}}))|,$$

where $\Phi^\ell \in \mathcal{N}(\sigma; \mathcal{A}^\ell, B)$ for $\mathcal{A}^\ell = (d_{\ell+1}, \dots, d_{L+1})$ and $(\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)})$, $(\overline{\mathbf{W}}^{(\ell)}, \overline{\mathbf{b}}^{(\ell)})$ differ in one entry only.

Using the Lipschitz continuity of Φ^ℓ of Lemma 13.2, we have

$$\begin{aligned}&|R_\sigma(\theta_j)(\boldsymbol{x}) - R_\sigma(\theta_{j+1})(\boldsymbol{x})| \\ &\quad\leq C_\sigma^{L-\ell-1}(B d_{\max})^{L-\ell} |\sigma(\mathbf{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + \mathbf{b}^{(\ell)}) - \sigma(\overline{\mathbf{W}^{(\ell)}}\boldsymbol{x}^{(\ell)} + \overline{\mathbf{b}^{(\ell)}})| \\ &\quad\leq C_\sigma^{L-\ell}(B d_{\max})^{L-\ell} \|\mathbf{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + \mathbf{b}^{(\ell)} - \overline{\mathbf{W}^{(\ell)}}\boldsymbol{x}^{(\ell)} - \overline{\mathbf{b}^{(\ell)}}\|_\infty \\ &\quad\leq C_\sigma^{L-\ell}(B d_{\max})^{L-\ell} \delta \max\{1, \|\boldsymbol{x}^{(\ell)}\|_\infty\},\end{aligned}$$

where $\delta \coloneqq \|\theta - \theta'\|_{\max}$. Invoking Lemma (13.3), we conclude that

$$\begin{aligned}|R_\sigma(\theta_j)(\boldsymbol{x}) - R_\sigma(\theta_{j+1})(\boldsymbol{x})| &\leq (2C_\sigma B d_{\max})^\ell C_\sigma^{L-\ell} \cdot (B d_{\max})^{L-\ell} \delta \\ &\leq (2C_\sigma B d_{\max})^L \|\theta - \theta'\|_{\max}.\end{aligned}$$

For the case $\ell = L$, a similar estimate can be shown. Combining this with (13.1.1) yields the result. □

Using Proposition 13.1, we can now consider the set of neural networks with a fixed architecture $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ as a subset of $L^\infty([-1,1]^{d_0})$. What is more, is that $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is the image of $\mathcal{PN}(\mathcal{A}, \infty)$ under a **locally Lipschitz map**.

13.2 Convexity of neural network spaces

As a first step towards understanding $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ as a subset of $L^\infty([-1,1]^{d_0})$, we notice that it is star-shaped with few centers. Let us first introduce the necessary terminology.

Definition 13.4. Let Z be a subset of a linear space. A point $x \in Z$ is called a **center of** Z if, for every $y \in Z$ it holds that

$$\{tx + (1-t)y \,|\, t \in [0,1]\} \subseteq Z.$$

A set is called **star-shaped** if it has at least one center.

The following proposition follows directly from the definition of a neural network and is the content of Exercise 13.15.

Proposition 13.5. *Let $L \in \mathbb{N}$ and $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$ and $\sigma \colon \mathbb{R} \to \mathbb{R}$. Then $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is scaling invariant, i.e. for every $\lambda \in \mathbb{R}$ it holds that $\lambda f \in \mathcal{N}(\sigma; \mathcal{A}, \infty)$ if $f \in \mathcal{N}(\sigma; \mathcal{A}, \infty)$, and hence $0 \in \mathcal{N}(\sigma; \mathcal{A}, \infty)$ is a center of $\mathcal{N}(\sigma; \mathcal{A}, \infty)$.*

Knowing that $\mathcal{N}(\sigma; \mathcal{A}, B)$ is star-shaped with center 0, we can also ask ourselves if $\mathcal{N}(\sigma; \mathcal{A}, B)$ has more than this one center. It is not hard to see that also every constant function is a center. The following theorem, which corresponds to [173, Proposition C.4], yields an upper bound on the number of *linearly independent* centers.

Theorem 13.6. *Let $L \in \mathbb{N}$ and $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be Lipschitz continuous. Then, $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ contains at most $n_\mathcal{A} = \sum_{\ell=0}^{L} (d_\ell + 1) d_{\ell+1}$ linearly independent centers.*

Proof. Assume by contradiction, that there are functions $(g_i)_{i=1}^{n_\mathcal{A}+1} \subseteq \mathcal{N}(\sigma; \mathcal{A}, \infty) \subseteq L^\infty([-1,1]^{d_0})$ that are linearly independent and centers of $\mathcal{N}(\sigma; \mathcal{A}, \infty)$.

By the Theorem of Hahn-Banach, there exist $(g_i')_{i=1}^{n_\mathcal{A}+1} \subseteq (L^\infty([-1,1]^{d_0}))'$ such that $g_i'(g_j) = \delta_{ij}$, for all $i, j \in \{1, \dots, L+1\}$. We define

$$T \colon L^\infty([-1,1]^{d_0}) \to \mathbb{R}^{n_\mathcal{A}+1}, \qquad g \mapsto \begin{pmatrix} g_1'(g) \\ g_2'(g) \\ \vdots \\ g_{n_\mathcal{A}+1}'(g) \end{pmatrix}.$$

Since T is continuous and linear, we have that $T \circ R_\sigma$ is locally Lipschitz continuous by Proposition 13.1. Moreover, since the $(g_i)_{i=1}^{n_\mathcal{A}+1}$ are linearly independent, we have that $T(\operatorname{span}((g_i)_{i=1}^{n_\mathcal{A}+1})) = \mathbb{R}^{n_\mathcal{A}+1}$. We denote $V \coloneqq \operatorname{span}((g_i)_{i=1}^{n_\mathcal{A}+1})$.

Next, we would like to establish that $\mathcal{N}(\sigma;\mathcal{A},\infty) \supset V$. Let $g \in V$ then

$$g = \sum_{\ell=1}^{n_{\mathcal{A}}+1} a_\ell g_\ell,$$

for some $a_1,\dots,a_{n_{\mathcal{A}}+1} \in \mathbb{R}$. We show by induction that $\tilde{g}^{(m)} \coloneqq \sum_{\ell=1}^{m} a_\ell g_\ell \in \mathcal{N}(\sigma;\mathcal{A},\infty)$ for every $m \leq n_{\mathcal{A}}+1$. This is obviously true for $m=1$. Moreover, we have that $\tilde{g}^{(m+1)} = a_{m+1}g_{m+1} + \tilde{g}^{(m)}$. Hence, the induction step holds true if $a_{m+1}=0$. If $a_{m+1} \neq 0$, then we have that

$$\widetilde{g}^{(m+1)} = 2a_{m+1} \cdot \left(\frac{1}{2} g_{m+1} + \frac{1}{2a_{m+1}} \widetilde{g}^{(m)} \right). \tag{13.2.1}$$

By the induction assumption $\widetilde{g}^{(m)} \in \mathcal{N}(\sigma;\mathcal{A},\infty)$ and hence by Proposition 13.5 $\widetilde{g}^{(m)}/(a_{m+1}) \in \mathcal{N}(\sigma;\mathcal{A},\infty)$. Additionally, since g_{m+1} is a center of $\mathcal{N}(\sigma;\mathcal{A},\infty)$, we have that $\frac{1}{2}g_{m+1} + \frac{1}{2a_{m+1}}\widetilde{g}^{(m)} \in \mathcal{N}(\sigma;\mathcal{A},\infty)$. By Proposition 13.5, we conclude that $\widetilde{g}^{(m+1)} \in \mathcal{N}(\sigma;\mathcal{A},\infty)$.

The induction shows that $g \in \mathcal{N}(\sigma;\mathcal{A},\infty)$ and thus $V \subseteq \mathcal{N}(\sigma;\mathcal{A},\infty)$. As a consequence, $T \circ R_\sigma(\mathcal{PN}(\mathcal{A},\infty)) \supseteq T(V) = \mathbb{R}^{n_{\mathcal{A}}+1}$.

It is a well known fact of basic analysis that for every $n \in \mathbb{N}$ there does not exist a surjective and locally Lipschitz continuous map from $\mathbb{R}^n$ to $\mathbb{R}^{n+1}$. We recall that $n_{\mathcal{A}} = \dim(\mathcal{PN}(\mathcal{A},\infty))$. This yields the contradiction. □

For a convex set X, the line between all two points of X is a subset of X. Hence, every point of a convex set is a center. This yields the following corollary.

Corollary 13.7. *Let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1})$, let, and let $\sigma : \mathbb{R} \to \mathbb{R}$ be Lipschitz continuous. If $\mathcal{N}(\sigma;\mathcal{A},\infty)$ contains more than $n_{\mathcal{A}} = \sum_{\ell=0}^{L}(d_\ell+1)d_{\ell+1}$ linearly independent functions, then $\mathcal{N}(\sigma;\mathcal{A},\infty)$ is not convex.*

Corollary 13.7 tells us that we cannot expect convex sets of neural networks, if the set of neural networks has many linearly independent elements. Sets of neural networks contain for each $f \in \mathcal{N}(\sigma;\mathcal{A},\infty)$ also all shifts of this function, i.e., $f(\cdot + \boldsymbol{b})$ for a $\boldsymbol{b} \in \mathbb{R}^d$ are elements of $f \in \mathcal{N}(\sigma;\mathcal{A},\infty)$. For a set of functions, being shift invariant and having only finitely many linearly independent functions at the same time, is a very restrictive condition. Indeed, it was shown in [173, Proposition C.6] that if $\mathcal{N}(\sigma;\mathcal{A},\infty)$ has only finitely many linearly independent functions and σ is differentiable in at least one point and has non-zero derivative there, then σ is necessarily a polynomial.

We conclude that the set of neural networks is in general non-convex and star-shaped with 0 and constant functions being centers. One could visualize this set in 3D as in Figure 13.1.

The fact, that the neural network space is not convex, could also mean that it merely fails to be convex at one point. For example $\mathbb{R}^2 \setminus \{0\}$ is not convex, but for an optimization algorithm this would likely not pose a problem.

We will next observe that $\mathcal{N}(\sigma;\mathcal{A},\infty)$ does not have such a benign non-convexity and in fact, has *arbitrarily large holes*.

To make this claim mathematically precise, we first introduce the notion of ε-convexity.

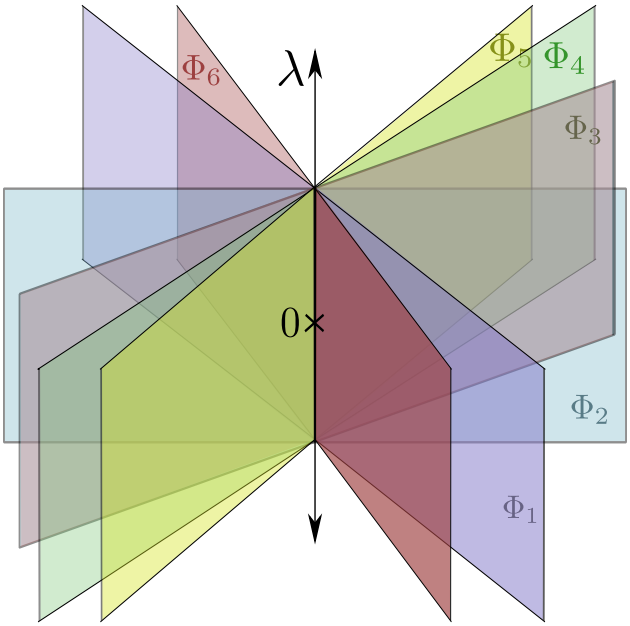


Figure 13.1: Sketch of the space of neural networks in 3D. The vertical axis corresponds to the constant neural network functions, each of which is a center. The set of neural networks consists of many low-dimensional linear subspaces spanned by certain neural networks ($\Phi_1, \dots, \Phi_6$ in this sketch) and linear functions. Between these low-dimensional subspaces, there is not always a straight-line connection by Corollary 13.7 and Theorem 13.9.

Definition 13.8. For $\varepsilon > 0$, we say that a subset A of a normed vector space X is ε-convex if

$$\mathrm{co}(A) \subseteq A + B_\varepsilon(0),$$

where $\mathrm{co}(A)$ denotes the convex hull of A and $B_\varepsilon(0)$ is an ε ball around 0 with respect to the norm of X.

Intuitively speaking, a set that is convex when one fills up all holes smaller than ε is ε-convex. Now we show that there is no $\varepsilon > 0$ such that $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is ε-convex.

Theorem 13.9. *Let $L \in \mathbb{N}$ and $\mathcal{A} = (d_0, d_1, \dots, d_L, 1) \in \mathbb{N}^{L+2}$. Let $K \subseteq \mathbb{R}^{d_0}$ be compact and let $\sigma \in \mathcal{M}$, with $\mathcal{M}$ as in (3.1.1) and assume that σ is not a polynomial. Moreover, assume that there exists an open set, where σ is differentiable and not constant.*

If there exists an $\varepsilon > 0$ such that $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is ε-convex, then $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is dense in $C(K)$.

Proof. **Step 1.** We show that ε-convexity implies $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ to be convex. By Proposition 13.5, we have that $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is scaling invariant. This implies that $\mathrm{co}(\mathcal{N}(\sigma; \mathcal{A}, \infty))$ is scaling invariant

as well. Hence, if there exists $\varepsilon > 0$ such that $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is ε-convex, then for every $\varepsilon' > 0$

$$\begin{aligned}\operatorname{co}(\mathcal{N}(\sigma; \mathcal{A}, \infty)) &= \frac{\varepsilon'}{\varepsilon} \operatorname{co}(\mathcal{N}(\sigma; \mathcal{A}, \infty)) \subseteq \frac{\varepsilon'}{\varepsilon} \left(\mathcal{N}(\sigma; \mathcal{A}, \infty) + B_\varepsilon(0)\right) \\ &= \mathcal{N}(\sigma; \mathcal{A}, \infty) + B_{\varepsilon'}(0).\end{aligned}$$

This yields that $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is ε'-convex. Since ε' was arbitrary, we have that $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is ε-convex for all $\varepsilon > 0$.

As a consequence, we have that

$$\begin{aligned}\operatorname{co}(\mathcal{N}(\sigma; \mathcal{A}, \infty)) &\subseteq \bigcap_{\varepsilon > 0} \left(\mathcal{N}(\sigma; \mathcal{A}, \infty) + B_\varepsilon(0)\right) \\ &\subseteq \bigcap_{\varepsilon > 0} \overline{\left(\mathcal{N}(\sigma; \mathcal{A}, \infty) + B_\varepsilon(0)\right)} = \overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}.\end{aligned}$$

Hence, $\overline{\operatorname{co}}(\mathcal{N}(\sigma; \mathcal{A}, \infty)) \subseteq \overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ and, by the well-known fact that in every metric vector space $\operatorname{co}(\overline{A}) \subseteq \overline{\operatorname{co}}(A)$, we conclude that $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ is convex.

Step 2. We show that $\mathcal{N}_d^1(\sigma; 1) \subseteq \overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$. If $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is ε-convex, then by Step 1 $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ is convex. The scaling invariance of $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ then shows that $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ is a closed linear subspace of $C(K)$.

Note that, by Proposition 3.16 for every $\boldsymbol{w} \in \mathbb{R}^{d_0}$ and $b \in \mathbb{R}$ there exists a function $f \in \overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ such that

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b) \qquad \text{for all } \boldsymbol{x} \in K. \tag{13.2.2}$$

By definition, every constant function is an element of $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$. Since $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ is a subspace, this implies that all constant functions are in $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$.

Since $\overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$ is a closed vector space, this implies that for all $n \in \mathbb{N}$ and all $\boldsymbol{w}_1^{(1)}, \dots, \boldsymbol{w}_n^{(1)} \in \mathbb{R}^{d_0}$, $w_1^{(2)}, \dots, w_n^{(2)} \in \mathbb{R}$, $b_1^{(1)}, \dots, b_n^{(1)} \in \mathbb{R}$, $b^{(2)} \in \mathbb{R}$

$$\boldsymbol{x} \mapsto \sum_{i=1}^n w_i^{(2)} \sigma((\boldsymbol{w}_i^{(1)})^\top \boldsymbol{x} + b_i^{(1)}) + b^{(2)} \in \overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}. \tag{13.2.3}$$

Step 3. From (13.2.3), we conclude that $\mathcal{N}_d^1(\sigma; 1) \subseteq \overline{\mathcal{N}(\sigma; \mathcal{A}, \infty)}$. In words, the whole set of shallow neural networks of arbitrary width is contained in the closure of the set of neural networks with a fixed architecture. By Theorem 3.8, we have that $\mathcal{N}_d^1(\sigma; 1)$ is dense in $C(K)$, which yields the result. □

For any activation function of practical relevance, a set of neural networks with fixed architecture is not dense in $C(K)$. This is only the case for very strange activation functions such as the one discussed in Subsection 3.2. Hence, Theorem 13.9 shows that in general, sets of neural networks of fixed architectures have arbitrarily large holes.

13.3 Closedness and best-approximation property

The non-convexity of the set of neural networks can have some serious consequences for the way we think of the approximation or learning problem by neural networks.

Consider $\mathcal{A} = (d_0, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$ and an activation function σ. Let H be a normed function space on $[-1,1]^{d_0}$ such that $\mathcal{N}(\sigma; \mathcal{A}, \infty) \subseteq H$. For $h \in H$ we would like to find a neural network that best approximates h, i.e. to find $\Phi \in \mathcal{N}(\sigma; \mathcal{A}, \infty)$ such that

$$\|\Phi - h\|_H = \inf_{\Phi^* \in \mathcal{N}(\sigma; \mathcal{A}, \infty)} \|\Phi^* - h\|_H. \tag{13.3.1}$$

We say that $\mathcal{N}(\sigma; \mathcal{A}, \infty) \subseteq H$ has

- the **best approximation property**, if for all $h \in H$ there exists at least one $\Phi \in \mathcal{N}(\sigma; \mathcal{A}, \infty)$ such that (13.3.1) holds,
- the **unique best approximation property**, if for all $h \in H$ there exists exactly one $\Phi \in \mathcal{N}(\sigma; \mathcal{A}, \infty)$ such that (13.3.1) holds,
- the **continuous selection property**, if there exists a continuous function $\phi \colon H \to \mathcal{N}(\sigma; \mathcal{A}, \infty)$ such that $\Phi = \phi(h)$ satisfies (13.3.1) for all $h \in H$.

We will see in the sequel, that, in the absence of the best approximation property, we will be able to prove that the learning problem necessarily requires the weights of the neural networks to tend to infinity, which may or may not be desirable in applications.

Moreover, having a continuous selection procedure is desirable as it implies the existence of a stable selection algorithm; that is, an algorithm which, for similar problems yields similar neural networks satisfying (13.3.1).

Below, we will study the properties above for L^p spaces, $p \in [1, \infty)$. As we will see, neural network classes typically neither satisfy the continuous selection nor the best approximation property.

13.3.1 Continuous selection

As shown in [111], neural network spaces essentially never admit the continuous selection property. To give the argument, we first recall the following result from [111, Theorem 3.4] without proof.

Theorem 13.10. *Let $p \in (1, \infty)$. Every subset of $L^p([-1,1]^{d_0})$ with the unique best approximation property is convex.*

This allows to show the next proposition.

Proposition 13.11. *Let $L \in \mathbb{N}$, $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma : \mathbb{R} \to \mathbb{R}$ be Lipschitz continuous and not a polynomial, and let $p \in (1, \infty)$.*

Then, $\mathcal{N}(\sigma; \mathcal{A}, \infty) \subseteq L^p([-1,1]^{d_0})$ does not have the continuous selection property.

Proof. We observe from Theorem 13.6 and the discussion below, that under the assumptions of this result, $\mathcal{N}(\sigma;\mathcal{A},\infty)$ is not convex.

We conclude that $\mathcal{N}(\sigma;\mathcal{A},\infty)$ does not have the unique best approximation property. Moreover, if the set $\mathcal{N}(\sigma;\mathcal{A},\infty)$ does not have the best approximation property, then it is obvious that it cannot have continuous selection. Thus, we can assume without loss of generality, that $\mathcal{N}(\sigma;\mathcal{A},\infty)$ has the best approximation property and there exists a point $h \in L^p([-1,1]^{d_0})$ and two different Φ_1,Φ_2 such that

$$\|\Phi_1 - h\|_{L^p} = \|\Phi_2 - h\|_{L^p} = \inf_{\Phi^* \in \mathcal{N}(\sigma;\mathcal{A},\infty)} \|\Phi^* - h\|_{L^p}. \tag{13.3.2}$$

Note that (13.3.2) implies that $h \notin \mathcal{N}(\sigma;\mathcal{A},\infty)$.

Let us consider the following function:

$$[-1,1] \ni \lambda \mapsto P(\lambda) = \begin{cases} (1+\lambda)h - \lambda\Phi_1 & \text{for } \lambda \leq 0, \\ (1-\lambda)h + \lambda\Phi_2 & \text{for } \lambda \geq 0. \end{cases}$$

It is clear that $P(\lambda)$ is a continuous path in L^p. Moreover, for $\lambda \in (-1,0)$

$$\|\Phi_1 - P(\lambda)\|_{L^p} = (1+\lambda)\|\Phi_1 - h\|_{L^p}.$$

Assume towards a contradiction, that there exists $\Phi^* \neq \Phi_1$ such that for $\lambda \in (-1,0)$

$$\|\Phi^* - P(\lambda)\|_{L^p} \leq \|\Phi_1 - P(\lambda)\|_{L^p}.$$

Then

$$\begin{aligned} \|\Phi^* - h\|_{L^p} &\leq \|\Phi^* - P(\lambda)\|_{L^p} + \|P(\lambda) - h\|_{L^p} \\ &\leq \|\Phi_1 - P(\lambda)\|_{L^p} + \|P(\lambda) - h\|_{L^p} \\ &= (1+\lambda)\|\Phi_1 - h\|_{L^p} + |\lambda|\|\Phi_1 - h\|_{L^p} = \|\Phi_1 - h\|_{L^p}. \end{aligned} \tag{13.3.3}$$

Since Φ_1 is a best approximation to h this implies that every inequality in the estimate above is an equality. Hence, we have that

$$\|\Phi^* - h\|_{L^p} = \|\Phi^* - P(\lambda)\|_{L^p} + \|P(\lambda) - h\|_{L^p}.$$

However, in a strictly convex space like $L^p([-1,1]^{d_0})$ for $p > 1$ this implies that

$$\Phi^* - P(\lambda) = c \cdot (P(\lambda) - h)$$

for a constant $c \neq 0$. This yields that

$$\Phi^* = h + (c+1)\lambda \cdot (h - \Phi_1)$$

and plugging into (13.3.3) yields $|(c+1)\lambda| = 1$. If $(c+1)\lambda = -1$, then we have $\Phi^* = \Phi_1$ which produces a contradiction. If $(c+1)\lambda = 1$, then

$$\begin{aligned} \|\Phi^* - P(\lambda)\|_{L^p} &= \|2h - \Phi_1 - (1+\lambda)h + \lambda\Phi_1\|_{L^p} \\ &= \|(1-\lambda)h - (1-\lambda)\Phi_1\|_{L^p} > \|P(\lambda) - \Phi_1\|_{L^p}, \end{aligned}$$

which is another contradiction.

Hence, for every $\lambda < 0$ we have that Φ_1 is the unique minimizer to $P(\lambda)$ in $\mathcal{N}(\sigma;\mathcal{A},\infty)$. The same argument holds for $\lambda > 0$ and Φ_2. We conclude that for every selection function $\phi\colon L^p([-1,1]^{d_0}) \to \mathcal{N}(\sigma;\mathcal{A},\infty)$ such that $\Phi = \phi(h)$ satisfies (13.3.1) for all $h \in L^p([-1,1]^{d_0})$ it holds that

$$\lim_{\lambda \downarrow 0} \phi(P(\lambda)) = \Phi_2 \neq \Phi_1 = \lim_{\lambda \uparrow 0} \phi(P(\lambda)).$$

As a consequence, ϕ is not continuous, which shows the result. □

13.3.2 Existence of best approximations

We have seen in Proposition 13.11 that under very mild assumptions, the continuous selection property cannot hold. Moreover, the next result shows that in many cases, also the best approximation property fails to be satisfied. We provide below a simplified version of [173, Theorem 3.1]. We also refer to [69] for earlier work on this problem.

Proposition 13.12. *Let $\mathcal{A} = (1, 2, 1)$ and let $\sigma : \mathbb{R} \to \mathbb{R}$ be Lipschitz continuous. Additionally assume that there exist $r > 0$ and $\alpha' \neq \alpha$ such that σ is differentiable for all $|x| > r$ and $\sigma'(x) \to \alpha$ for $x \to \infty$, $\sigma'(x) \to \alpha'$ for $x \to -\infty$.*

Then, there exists a sequence in $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ which converges in $L^p([-1, 1])$, for every $p \in (1, \infty)$, and the limit of this sequence is discontinuous. In particular, the limit of the sequence does not lie in $\mathcal{N}(\sigma; \mathcal{A}', \infty)$ for any $\mathcal{A}'$.

Proof. For all $n \in \mathbb{N}$ let

$$f_n(x) = \sigma(nx + 1) - \sigma(nx) \qquad \text{for all } x \in \mathbb{R}.$$

Then f_n can be written as a neural network with architecture $(\sigma; 1, 2, 1)$, i.e., $\mathcal{A} = (1, 2, 1)$. Moreover, for $x > 0$ we observe with the fundamental theorem of calculus and using integration by substitution that

$$f_n(x) = \int_x^{x+1/n} n\sigma'(nz)dz = \int_{nx}^{nx+1} \sigma'(z)dz. \tag{13.3.4}$$

It is not hard to see that the right hand side of (13.3.4) converges to α for $n \to \infty$.

Similarly, for $x < 0$, we observe that $f_n(x)$ converges to α' for $n \to \infty$. We conclude that

$$f_n \to \alpha \mathbb{1}_{\mathbb{R}_+} + \alpha' \mathbb{1}_{\mathbb{R}_-}$$

almost everywhere as $n \to \infty$. Since σ is Lipschitz continuous, we have that f_n is bounded. Therefore, we conclude that $f_n \to \alpha \mathbb{1}_{\mathbb{R}_+} + \alpha' \mathbb{1}_{\mathbb{R}_-}$ in L^p for all $p \in [1, \infty)$ by the dominated convergence theorem. □

There is a straight-forward extension of Proposition 13.12 to arbitrary architectures, that will be the content of Exercises 13.16 and 13.17.

Remark 13.13. The proof of Theorem 13.12 does not extend to the L^∞ norm. This, of course, does not mean that generally $\mathcal{N}(\sigma; \mathcal{A}, \infty)$ is a closed set in $L^\infty([-1, 1]^{d_0})$. In fact, almost all activation functions used in practice still give rise to non-closed neural network sets, see [173, Theorem 3.3]. However, there is one notable exception. For the ReLU activation function, it can be shown that $\mathcal{N}(\sigma_{\mathrm{ReLU}}; \mathcal{A}, \infty)$ is a closed set in $L^\infty([-1, 1]^{d_0})$ if $\mathcal{A}$ has only one hidden layer. The closedness of deep ReLU spaces in L^∞ is an open problem.

13.3.3 Exploding weights phenomenon

Finally, we discuss one of the consequences of the non-existence of best approximations of Proposition 13.12.

Consider a regression problem, where we aim to learn a function f using neural networks with a fixed architecture $\mathcal{N}(\mathcal{A}; \sigma, \infty)$. As discussed in the Chapters 10 and 11, we wish to produce a sequence of neural networks $(\Phi_n)_{n=1}^{\infty}$ such that the risk defined in (1.2.4) converges to 0. If the loss $\mathcal{L}$ is the squared loss, μ is a probability measure on $[-1,1]^{d_0}$, and the data is given by $(\boldsymbol{x}, f(\boldsymbol{x}))$ for $\boldsymbol{x} \sim \mu$, then

$$\begin{aligned} \mathcal{R}(\Phi_n) &= \|\Phi_n - f\|^2_{L^2([-1,1]^{d_0},\mu)} \\ &= \int_{[-1,1]^{d_0}} |\Phi_n(\boldsymbol{x}) - f(\boldsymbol{x})|^2 d\mu(\boldsymbol{x}) \to 0 \qquad \text{for } n \to \infty. \end{aligned} \tag{13.3.5}$$

According to Proposition 13.12, for a given $\mathcal{A}$, and an activation function σ, it is possible that (13.3.5) holds, but $f \notin \mathcal{N}(\sigma; \mathcal{A}, \infty)$. The following result shows that in this situation, the weights of Φ_n diverge.

Proposition 13.14. *Let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma \colon \mathbb{R} \to \mathbb{R}$ be Lipschitz continuous with $C_\sigma \geq 1$, and $|\sigma(x)| \leq C_\sigma |x|$ for all $x \in \mathbb{R}$, and let μ be a measure on $[-1,1]^{d_0}$.*

Assume that there exists a sequence $\Phi_n \in \mathcal{N}(\sigma; \mathcal{A}, \infty)$ and $f \in L^2([-1,1]^{d_0}, \mu) \setminus \mathcal{N}(\sigma; \mathcal{A}, \infty)$ such that

$$\|\Phi_n - f\|^2_{L^2([-1,1]^{d_0},\mu)} \to 0. \tag{13.3.6}$$

Then

$$\limsup_{n\to\infty} \max\left\{ \|\boldsymbol{W}_n^{(\ell)}\|_\infty, \|\boldsymbol{b}_n^{(\ell)}\|_\infty \,\middle|\, \ell = 0, \dots L \right\} = \infty. \tag{13.3.7}$$

Proof. We assume towards a contradiction that the left-hand side of (13.3.7) is finite. As a result, there exists $C > 0$ such that $\Phi_n \in \mathcal{N}(\sigma; \mathcal{A}, C)$ for all $n \in \mathbb{N}$.

By Proposition 13.1, we conclude that $\mathcal{N}(\sigma; \mathcal{A}, C)$ is the image of a compact set under a continuous map and hence is itself a compact set in $L^2([-1,1]^{d_0}, \mu)$. In particular, we have that $\mathcal{N}(\sigma; \mathcal{A}, C)$ is closed. Hence, (13.3.6) implies $f \in \mathcal{N}(\sigma; \mathcal{A}, C)$. This gives a contradiction. □

Proposition 13.14 can be extended to all f for which there is no best approximation in $\mathcal{N}(\sigma; \mathcal{A}, \infty)$, see Exercise 13.18. The results imply that for functions we wish to learn that lack a best approximation within a neural network set, we must expect the weights of the approximating neural networks to grow to infinity. This can be undesirable because, as we will see in the following sections on generalization, a bounded parameter space facilitates many generalization bounds.

Bibliography and further reading

The properties of neural network sets were first studied with a focus on the continuous approximation property in [111, 113, 112] and [69]. The results in [111, 112, 113] already use the non-convexity

of sets of shallow neural networks. The results on convexity and closedness presented in this chapter follow mostly the arguments of [173]. Similar results were also derived for other norms in [139].

Exercises

Exercise 13.15. Prove Proposition 13.5.

Exercise 13.16. Extend Proposition 13.12 to $\mathcal{A} = (d_0, d_1, 1)$ for arbitrary d_0, $d_1 \in \mathbb{N}$, $d_1 \geq 2$.

Exercise 13.17. Use Proposition 3.16, to extend Proposition 13.12 to arbitrary depth.

Exercise 13.18. Extend Proposition 13.14 to functions f for which there is no best-approximation in $\mathcal{N}(\sigma; \mathcal{A}, \infty)$. To do this, replace (13.3.6) by

$$\|\Phi_n - f\|_{L^2}^2 \to \inf_{\Phi \in \mathcal{N}(\sigma;\mathcal{A},\infty)} \|\Phi - f\|_{L^2}^2.$$

Chapter 14

Generalization properties of deep neural networks

As discussed in the introduction in Section 1.2, we generally learn based on a finite data set. For example, given data $(x_i, y_i)_{i=1}^m$, we try to find a network Φ that satisfies $\Phi(x_i) = y_i$ for $i = 1, \dots, m$. The field of generalization is concerned with how well such Φ performs on *unseen* data, which refers to any x outside of training data $\{x_1, \dots, x_m\}$. In this chapter we discuss generalization through the use of covering numbers.

In Sections 14.1 and 14.2 we revisit and formalize the general setup of learning and empirical risk minimization in a general context. Although some notions introduced in these sections have already appeared in the previous chapters, we reintroduce them here for a more coherent presentation. In Sections 14.3-14.5, we first discuss the concepts of generalization bounds and covering numbers, and then apply these arguments specifically to neural networks. In Section 14.6 we explore the so-called "approximation-complexity trade-off", and finally in Sections 14.7-14.8 we introduce the "VC dimension" and give some implications for classes of neural networks.

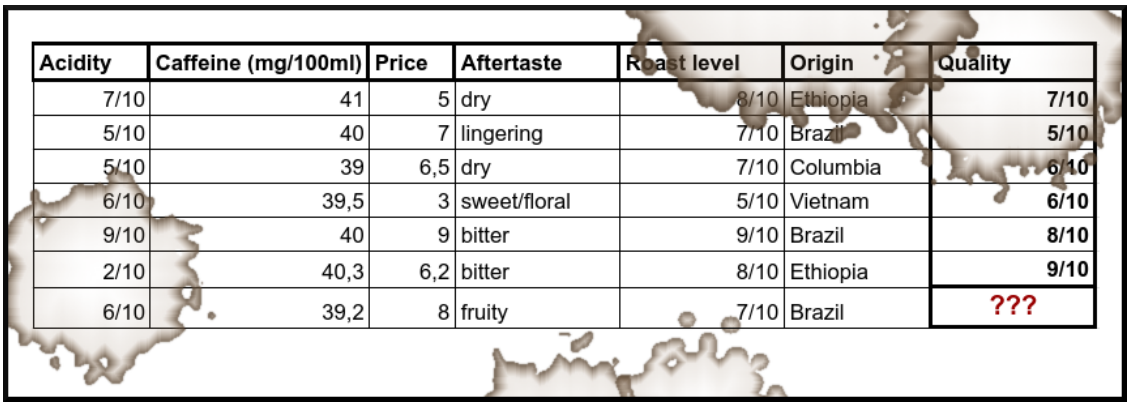
14.1 Learning setup

A general learning problem [148, 212, 43] requires a **feature space** X and a **label space** Y, which we assume throughout to be measurable spaces. We observe joint data pairs $(x_i, y_i)_{i=1}^m \subseteq X \times Y$, and aim to identify a connection between the x and y variables. Specifically, we assume a relationship between features x and labels y modeled by a probability distribution $\mathcal{D}$ over $X \times Y$, that generated the observed data $(x_i, y_i)_{i=1}^m$. While this distribution is unknown, our goal is to extract information from it, so that we can make possibly good predictions of y for a given x. Importantly, the relationship between x and y need not be deterministic.

To make these concepts more concrete, we next present an example that will serve as the running example throughout this chapter. This example is of high relevance for many mathematicians, as ensuring a steady supply of high-quality coffee is essential for maximizing the output of our mathematical work.

Example 14.1 (Coffee Quality)**.** Our goal is to determine the quality of different coffees. To this end we model the quality as a number in

$$Y = \Big\{ \frac{0}{10}, \dots, \frac{10}{10} \Big\},$$



Acidity	Caffeine (mg/100ml)	Price	Aftertaste	Roast level	Origin	Quality
7/10	41	5	dry	8/10	Ethiopia	**7/10**
5/10	40	7	lingering	7/10	Brazil	**5/10**
5/10	39	6,5	dry	7/10	Columbia	**6/10**
6/10	39,5	3	sweet/floral	5/10	Vietnam	**6/10**
9/10	40	9	bitter	9/10	Brazil	**8/10**
2/10	40,3	6,2	bitter	8/10	Ethiopia	**9/10**
6/10	39,2	8	fruity	7/10	Brazil	**???**

Figure 14.1: Collection of coffee data. The last row lacks a "Quality" label. Our aim is to predict the label without the need for an (expensive) taste test.

with higher numbers indicating better quality. Let us assume that our subjective assessment of quality of coffee is related to six features: "Acidity", "Caffeine content", "Price", "Aftertaste", "Roast level", and "Origin". The feature space X thus corresponds to the set of six-tuples describing these attributes, which can be either numeric or categorical (see Figure 14.1).

We aim to understand the relationship between elements of X and elements of Y, but we can neither afford, nor do we have the time to taste all the coffees in the world. Instead, we can sample some coffees, taste them, and grow our database accordingly as depicted in Figure 14.1. This way we obtain samples of pairs in $X \times Y$. The distribution $\mathcal{D}$ from which they are drawn depends on various external factors. For instance, we might have avoided particularly cheap coffees, believing them to be inferior. As a result they do not occur in our database. Moreover, if a colleague contributes to our database, he might have tried the same brand and arrived at a different rating. In this case, the quality label is not deterministic anymore.

Based on our database, we wish to predict the quality of an untasted coffee. Before proceeding, we first formalize what it means to be a "good" prediction.

Characterizing how good a predictor is requires a notion of discrepancy in the label space. This is the purpose of the so-called **loss function**, which is a measurable mapping $\mathcal{L}\colon Y \times Y \to \mathbb{R}_+$.

Definition 14.2. Let $\mathcal{L}\colon Y \times Y \to \mathbb{R}_+$ be a loss function and let $\mathcal{D}$ be a distribution on $X \times Y$. For a measurable function $h\colon X \to Y$ we call

$$\mathcal{R}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathcal{L}(h(x), y)\right]$$

the **(population) risk of** h.

Based on the risk, we can now formalize what we consider a good predictor. The best predictor is one such that its risk is as close as possible to the smallest that any function can achieve. More precisely, we would like a risk that is close to the so-called **Bayes risk**

$$R^* \coloneqq \inf_{h\colon X\to Y} \mathcal{R}(h), \tag{14.1.1}$$

where the infimum is taken over all h such that $h : X \to Y$ is measurable.

Example 14.3 (Loss functions)**.** The choice of a loss function $\mathcal{L}$ usually depends on the application. For a regression problem, i.e., a learning problem where Y is a non-discrete subset of a Euclidean space, a common choice is the square loss $\mathcal{L}_2(\boldsymbol{y}, \boldsymbol{y}') = \|\boldsymbol{y} - \boldsymbol{y}'\|^2$.

For binary classification problems, i.e. when Y is a discrete set of cardinality two, the "$0-1$ loss"

$$\mathcal{L}_{0-1}(y, y') = \begin{cases} 1 & y \neq y' \\ 0 & y = y' \end{cases}$$

is a common choice.

Another frequently used loss for binary classification, especially when we want to predict probabilities (i.e., if $Y = [0, 1]$ but all labels are binary), is the binary cross-entropy loss

$$\mathcal{L}_{\text{ce}}(y, y') = -(y \log(y') + (1 - y) \log(1 - y')).$$

In contrast to the $0-1$ loss, the cross-entropy loss is differentiable, which is desirable in deep learning as we saw in Chapter 10.

In the coffee quality prediction problem, the quality is given as a fraction of the form $k/10$ for $k = 0, \ldots, 10$. While this is a discrete set, it makes sense to more heavily penalize predictions that are wrong by a larger amount. For example, predicting $4/10$ instead of $8/10$ should produce a higher loss than predicting $7/10$. Hence, we would not use the $0-1$ loss but, for example, the square loss.

How do we find a function $h\colon X \to Y$ with a risk that is as close as possible to the Bayes risk? We will introduce a procedure to tackle this task in the next section.

14.2 Empirical risk minimization

Finding a minimizer of the risk constitutes a considerable challenge. First, we cannot search through all measurable functions. Therefore, we need to restrict ourselves to a specific set $\mathcal{H} \subseteq \{h : X \to Y\}$ called the **hypothesis set**. In the following, this set will be some set of neural networks. Second, we are faced with the problem that we cannot evaluate $\mathcal{R}(h)$ for non-trivial loss functions, because the distribution $\mathcal{D}$ is unknown. To approximate the risk, we will assume access to an i.i.d. sample of m observations drawn from $\mathcal{D}$. This is precisely the situation described in the coffee quality example of Figure 14.1, where $m = 6$ coffees were sampled.[1] So for a given hypothesis h we can check how well it performs on our sampled data. We call the error on the sample the **empirical risk**.

Definition 14.4. Let $m \in \mathbb{N}$, let $\mathcal{L}\colon Y \times Y \to \mathbb{R}$ be a loss function and let $S = (x_i, y_i)_{i=1}^m \in (X \times Y)^m$ be a sample. For $h\colon X \to Y$, we call

$$\widehat{\mathcal{R}}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(h(x_i), y_i)$$

the **empirical risk** of h.

[1] In practice, the assumption of independence of the samples is often unclear and typically not satisfied. For instance, the selection of the six previously tested coffees might be influenced by external factors such as personal preferences or availability at the local store, which introduce bias into the dataset.

If the sample S is drawn i.i.d. according to $\mathcal{D}$, then we immediately see from the linearity of the expected value that $\widehat{\mathcal{R}}_S(h)$ is an unbiased estimator of $\mathcal{R}(h)$, i.e., $\mathbb{E}_{S\sim\mathcal{D}^m}[\widehat{\mathcal{R}}_S(h)] = \mathcal{R}(h)$. Moreover, the weak law of large numbers states that the sample mean of an i.i.d. sequence of integrable random variables converges to the expected value in probability. Hence, there is some hope that, at least for large $m \in \mathbb{N}$, minimizing the empirical risk instead of the population risk might lead to a good hypothesis. We formalize this approach in the next definition.

Definition 14.5. Let $\mathcal{H} \subseteq \{h\colon X \to Y\}$ be a hypothesis set. Let $m \in \mathbb{N}$, let $\mathcal{L}\colon Y \times Y \to \mathbb{R}$ be a loss function and let $S = (x_i, y_i)_{i=1}^m \in (X \times Y)^m$ be a sample. We call a function h_S such that

$$\widehat{\mathcal{R}}_S(h_S) = \inf_{h\in\mathcal{H}} \widehat{\mathcal{R}}_S(h) \tag{14.2.1}$$

an **empirical risk minimizer**.

From a generalization perspective, deep learning is empirical risk minimization over sets of neural networks. The question we want to address next is how effective this approach is at producing hypotheses that achieve a risk close to the Bayes risk.

Let $\mathcal{H}$ be some hypothesis set, such that an empirical risk minimizer h_S exists for all $S \in (X \times Y)^m$; see Exercise 14.25 for an explanation of why this is a reasonable assumption. Moreover, let $h^* \in \mathcal{H}$ be arbitrary. Then

$$\begin{aligned}\mathcal{R}(h_S) - R^* &= \mathcal{R}(h_S) - \widehat{\mathcal{R}}_S(h_S) + \widehat{\mathcal{R}}_S(h_S) - R^* \\ &\leq |\mathcal{R}(h_S) - \widehat{\mathcal{R}}_S(h_S)| + \widehat{\mathcal{R}}_S(h^*) - R^* \\ &\leq 2\sup_{h\in\mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| + \mathcal{R}(h^*) - R^*,\end{aligned} \tag{14.2.2}$$

where in the first inequality we used that h_S is the empirical risk minimizer. By taking the infimum over all h^*, we conclude that

$$\begin{aligned}\mathcal{R}(h_S) - R^* &\leq 2\sup_{h\in\mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| + \inf_{h\in\mathcal{H}} \mathcal{R}(h) - R^* \\ &=: 2\varepsilon_{\text{gen}} + \varepsilon_{\text{approx}}.\end{aligned} \tag{14.2.3}$$

Similarly, considering only (14.2.2), yields that

$$\begin{aligned}\mathcal{R}(h_S) &\leq \sup_{h\in\mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| + \inf_{h\in\mathcal{H}} \widehat{\mathcal{R}}_S(h) \\ &=: \varepsilon_{\text{gen}} + \varepsilon_{\text{int}}.\end{aligned} \tag{14.2.4}$$

How to choose $\mathcal{H}$ to reduce the **approximation error** $\varepsilon_{\text{approx}}$ or the **interpolation error** ε_{int} was discussed at length in the previous chapters. The final piece is to figure out how to bound the **generalization error** $\sup_{h\in\mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)|$. This will be discussed in the sections below.

14.3 Generalization bounds

We have seen that one aspect of successful learning is to bound the generalization error ε_{gen} in (14.2.3). Let us first formally describe this problem.

Definition 14.6 (Generalization bound)**.** Let $\mathcal{H} \subseteq \{h\colon X \to Y\}$ be a hypothesis set, and let $\mathcal{L}\colon Y \times Y \to \mathbb{R}$ be a loss function. Let $\kappa\colon (0,1) \times \mathbb{N} \to \mathbb{R}_+$ be such that for every $\delta \in (0,1)$ holds $\kappa(\delta, m) \to 0$ for $m \to \infty$. We call κ a **generalization bound for** $\mathcal{H}$ if for every distribution $\mathcal{D}$ on $X \times Y$, every $m \in \mathbb{N}$ and every $\delta \in (0,1)$, it holds with probability at least $1-\delta$ over the random sample $S \sim \mathcal{D}^m$ that

$$\sup_{h \in \mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| \leq \kappa(\delta, m).$$

Remark 14.7. For a generalization bound κ it holds that

$$\mathbb{P}\left[\left|\mathcal{R}(h_S) - \widehat{\mathcal{R}}_S(h_S)\right| \leq \varepsilon\right] \geq 1 - \delta$$

as soon as m is so large that $\kappa(\delta, m) \leq \varepsilon$. If there exists an empirical risk minimizer h_S such that $\widehat{\mathcal{R}}_S(h_S) = 0$, then with high probability the empirical risk minimizer will also have a small risk $\mathcal{R}(h_S)$. Empirical risk minimization is often referred to as a "PAC" algorithm, which stands for *probably (δ) approximately correct (ε).*

Definition 14.6 requires the upper bound κ on the discrepancy between the empirical risk and the risk to be independent from the distribution $\mathcal{D}$. Why should this be possible? After all, we could have an underlying distribution that is not uniform and hence, certain data points could appear very rarely in the sample. As a result, it should be very hard to produce a correct prediction for such points. At first sight, this suggests that non-uniform distributions should be much more challenging than uniform distributions. This intuition is incorrect, as the following argument based on Example 14.1 demonstrates.

Example 14.8 (Generalization in the coffee quality problem)**.** In Example 14.1, the underlying distribution describes both our process of choosing coffees and the relation between the attributes and the quality. Suppose we do not enjoy drinking coffee that costs less than 1€. Consequently, we do not have a single sample of such coffee in the dataset, and therefore we have no chance about learning the quality of cheap coffees.

However, the absence of coffee samples costing less than 1€ in our dataset is due to our *general* avoidance of such coffee. As a result, we run a low risk of incorrectly classifying the quality of a coffee that is cheaper than 1€, since it is unlikely that we will choose such a coffee in the future.

To establish generalization bounds, we use stochastic tools that guarantee that the empirical risk converges to the true risk as the sample size increases. This is typically achieved through concentration inequalities. One of the simplest and most well-known is Hoeffding's inequality, see Theorem A.24. We will now apply Hoeffding's inequality to obtain a first generalization bound. This generalization bound is well-known and can be found in many textbooks on machine learning, e.g., [148, 212]. Although the result does not yet encompass neural networks, it forms the basis for a similar result applicable to neural networks, as we discuss subsequently.

Proposition 14.9 (Finite hypothesis set)**.** *Let $\mathcal{H} \subseteq \{h\colon X \mapsto Y\}$ be a finite hypothesis set. Let $\mathcal{L}\colon Y \times Y \to \mathbb{R}$ be such that $\mathcal{L}(Y \times Y) \subseteq [c_1, c_2]$ with $c_2 - c_1 = C > 0$.*

Then, for every $m \in \mathbb{N}$ and every distribution $\mathcal{D}$ on $X \times Y$ it holds with probability at least $1-\delta$ over the sample $S \sim \mathcal{D}^m$ that

$$\sup_{h\in\mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| \leq C\sqrt{\frac{\log(|\mathcal{H}|) + \log(2/\delta)}{2m}}.$$

Proof. Let $\mathcal{H} = \{h_1, \dots, h_n\}$. Then it holds by a union bound that

$$\mathbb{P}\left[\exists h_i \in \mathcal{H}\colon |\mathcal{R}(h_i) - \widehat{\mathcal{R}}_S(h_i)| > \varepsilon\right] \leq \sum_{i=1}^{n} \mathbb{P}\left[|\mathcal{R}(h_i) - \widehat{\mathcal{R}}_S(h_i)| > \varepsilon\right].$$

Note that $\widehat{\mathcal{R}}_S(h_i)$ is the mean of independent random variables which take their values almost surely in $[0, C]$. Additionally, $\mathcal{R}(h_i)$ is the expectation of $\widehat{\mathcal{R}}_S(h_i)$. The proof can therefore be finished by applying Theorem A.24. This will be addressed in Exercise 14.26. □

Consider now a *non-finite* set of neural networks $\mathcal{H}$, and assume that it can be covered by a *finite* set of (small) balls. Applying Proposition 14.9 to the centers of these balls, then allows to derive a similar bound as in the proposition for $\mathcal{H}$. This intuitive argument will be made rigorous in the following section.

14.4 Generalization bounds from covering numbers

To derive a generalization bound for classes of neural networks, we start by introducing the notion of covering numbers.

Definition 14.10. Let A be a relatively compact subset of a metric space (X, d). For $\varepsilon > 0$, we call

$$\mathcal{G}(A, \varepsilon, (X, d)) := \min\left\{ m \in \mathbb{N} \,\middle|\, \exists\, (x_i)_{i=1}^m \subseteq X \text{ s.t. } \bigcup_{i=1}^m B_\varepsilon(x_i) \supset A \right\},$$

where $B_\varepsilon(x) = \{z \in X \,|\, d(z, x) \leq \varepsilon\}$, the ε-covering number of A in X. In case X or d are clear from context, we also write $\mathcal{G}(A, \varepsilon, d)$ or $\mathcal{G}(A, \varepsilon, X)$ instead of $\mathcal{G}(A, \varepsilon, (X, d))$.

A visualization of Definition 14.10 is given in Figure 14.2. As we will see, it is possible to upper bound the ε-covering numbers of neural networks as a subset of $L^\infty([0,1]^d)$, assuming the weights are confined to a fixed bounded set. The precise estimates are postponed to Section 14.5. Before that, let us show how a finite covering number facilitates a generalization bound. We only consider Euclidean feature spaces X in the following result. A more general version could be easily derived.

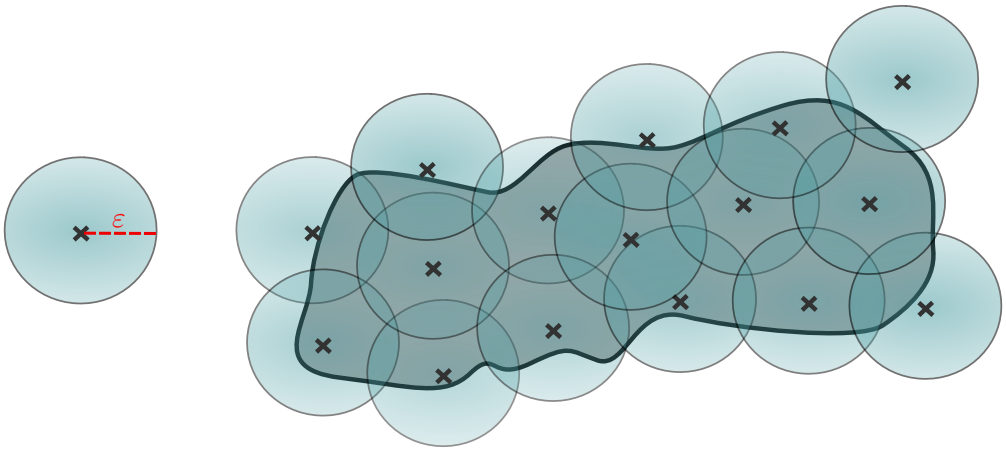


Figure 14.2: Illustration of the concept of covering numbers of Definition 14.10. The shaded set $A \subseteq \mathbb{R}^2$ is covered by sixteen Euclidean balls of radius ε. Therefore, $\mathcal{G}(A, \varepsilon, \mathbb{R}^2) \leq 16$.

Theorem 14.11. *Let C_Y, $C_{\mathcal{L}} > 0$ and $\alpha > 0$. Let $Y \subseteq [-C_Y, C_Y]$, $X \subseteq \mathbb{R}^d$ for some $d \in \mathbb{N}$, and $\mathcal{H} \subseteq \{h \colon X \to Y\}$. Further, let $\mathcal{L} \colon Y \times Y \to \mathbb{R}$ be $C_{\mathcal{L}}$-Lipschitz.*

Then, for every distribution $\mathcal{D}$ on $X \times Y$ and every $m \in \mathbb{N}$ it holds with probability at least $1 - \delta$ over the sample $S \sim \mathcal{D}^m$ that for all $h \in \mathcal{H}$

$$|\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| \leq 4 C_Y C_{\mathcal{L}} \sqrt{\frac{\log(\mathcal{G}(\mathcal{H}, m^{-\alpha}, L^\infty(X))) + \log(2/\delta)}{m}} + \frac{2 C_{\mathcal{L}}}{m^\alpha}.$$

Proof. Let

$$M = \mathcal{G}(\mathcal{H}, m^{-\alpha}, L^\infty(X)) \tag{14.4.1}$$

and let $\mathcal{H}_M = (h_i)_{i=1}^M \subseteq \mathcal{H}$ be such that for every $h \in \mathcal{H}$ there exists $h_i \in \mathcal{H}_M$ with $\|h - h_i\|_{L^\infty(X)} \leq 1/m^\alpha$. The existence of $\mathcal{H}_M$ follows by Definition 14.10.

Fix for the moment such $h \in \mathcal{H}$ and $h_i \in \mathcal{H}_M$. By the reverse and normal triangle inequalities, we have

$$|\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| - |\mathcal{R}(h_i) - \widehat{\mathcal{R}}_S(h_i)| \leq |\mathcal{R}(h) - \mathcal{R}(h_i)| + |\widehat{\mathcal{R}}_S(h) - \widehat{\mathcal{R}}_S(h_i)|.$$

Moreover, from the monotonicity of the expected value and the Lipschitz property of $\mathcal{L}$ it follows that

$$\begin{aligned} |\mathcal{R}(h) - \mathcal{R}(h_i)| &\leq \mathbb{E}|\mathcal{L}(h(x), y) - \mathcal{L}(h_i(x), y)| \\ &\leq C_{\mathcal{L}} \mathbb{E}|h(x) - h_i(x)| \leq \frac{C_{\mathcal{L}}}{m^\alpha}. \end{aligned}$$

A similar estimate yields $|\widehat{\mathcal{R}}_S(h) - \widehat{\mathcal{R}}_S(h_i)| \leq C_{\mathcal{L}}/m^\alpha$.

We thus conclude that for every $\varepsilon > 0$

$$\begin{aligned}&\mathbb{P}_{S\sim\mathcal{D}^m}\left[\exists h \in \mathcal{H}\colon |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| \geq \varepsilon\right] \\ &\quad \leq \mathbb{P}_{S\sim\mathcal{D}^m}\left[\exists h_i \in \mathcal{H}_M\colon |\mathcal{R}(h_i) - \widehat{\mathcal{R}}_S(h_i)| \geq \varepsilon - \frac{2C_{\mathcal{L}}}{m^\alpha}\right]. \end{aligned} \tag{14.4.2}$$

From Proposition 14.9, we know that for $\varepsilon > 0$ and $\delta \in (0,1)$

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left[\exists h_i \in H_M\colon |\mathcal{R}(h_i) - \widehat{\mathcal{R}}_S(h_i)| \geq \varepsilon - \frac{2C_{\mathcal{L}}}{m^\alpha}\right] \leq \delta \tag{14.4.3}$$

as long as

$$\varepsilon - \frac{2C_{\mathcal{L}}}{m^\alpha} > C\sqrt{\frac{\log(M) + \log(2/\delta)}{2m}},$$

where C is such that $\mathcal{L}(Y \times Y) \subseteq [c_1, c_2]$ with $c_2 - c_1 \leq C$. By the Lipschitz property of $\mathcal{L}$ we can choose $C = 2\sqrt{2}C_{\mathcal{L}}C_Y$.

Therefore, the definition of M in (14.4.1) together with (14.4.2) and (14.4.3) give that with probability at least $1-\delta$ it holds for all $h \in \mathcal{H}$

$$|\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| \leq 2\sqrt{2}C_{\mathcal{L}}C_Y\sqrt{\frac{\log(\mathcal{G}(\mathcal{H}, m^{-\alpha}, L^\infty)) + \log(2/\delta)}{2m}} + \frac{2C_{\mathcal{L}}}{m^\alpha}.$$

This concludes the proof. □

14.5 Covering numbers of deep neural networks

We have seen in Theorem 14.11, estimating L^∞-covering numbers is crucial for understanding the generalization error. How can we determine these covering numbers? The set of neural networks of a fixed architecture can be a quite complex set (see Chapter 13), so it is not immediately clear how to cover it with balls, let alone know the number of required balls. The following lemma suggest a simpler approach.

Lemma 14.12. *Let X_1, X_2 be two metric spaces and let $f\colon X_1 \to X_2$ be Lipschitz continuous with Lipschitz constant C_{Lip}. For every relatively compact $A \subseteq X_1$ it holds that for all $\varepsilon > 0$*

$$\mathcal{G}(f(A), C_{\mathrm{Lip}}\varepsilon, X_2) \leq \mathcal{G}(A, \varepsilon, X_1).$$

The proof of Lemma 14.12 is left as an exercise. If we can represent the set of neural networks as the image under the Lipschitz map of another set with known covering numbers, then Lemma 14.12 gives a direct way to bound the covering number of the neural network class.

Conveniently, we have already observed in Proposition 13.1, that the set of neural networks is the image of $\mathcal{PN}(\mathcal{A}, B)$ as in Definition 12.1 under the Lipschitz continuous realization map R_σ. It thus suffices to establish the ε-covering number of $\mathcal{PN}(\mathcal{A}, B)$ or equivalently of $[-B, B]^{n_{\mathcal{A}}}$. Then, using the Lipschitz property of R_σ that holds by Proposition 13.1, we can apply Lemma 14.12 to find the covering numbers of $\mathcal{N}(\sigma; \mathcal{A}, B)$. This idea is depicted in Figure 14.3.

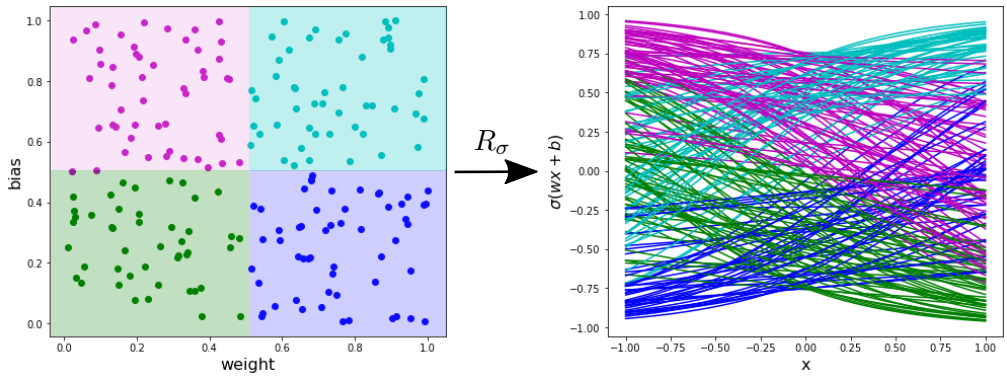


Figure 14.3: Illustration of the main idea to deduce covering numbers of neural network spaces. Points $\theta \in \mathbb{R}^2$ in parameter space in the left figure correspond to functions $R_\sigma(\theta)$ in the right figure (with matching colors). By Lemma 14.12, a covering of the parameter space on the left translates to a covering of the function space on the right.

Proposition 14.13. *Let B, $\varepsilon > 0$ and $q \in \mathbb{N}$. Then*

$$\mathcal{G}([-B,B]^q, \varepsilon, (\mathbb{R}^q, \|\cdot\|_\infty)) \leq \lceil B/\varepsilon \rceil^q.$$

Proof. We start with the one-dimensional case $q = 1$. We choose $k = \lfloor B/\varepsilon \rfloor$

$$x_0 = -B + \varepsilon \text{ and } x_j = x_{j-1} + 2\varepsilon \text{ for } j = 1, \dots, k-1.$$

It is clear that all points between $-B$ and x_{k-1} have distance at most ε to one of the x_j. Also, $x_{k-1} = -B + \varepsilon + 2(k-1)\varepsilon \geq B - \varepsilon$. We conclude that $\mathcal{G}([-B,B], \varepsilon, \mathbb{R}) \leq \lceil B/\varepsilon \rceil$. Set $X_k := \{x_0, \dots, x_{k-1}\}$.

For arbitrary q, we observe that for every $x \in [-B,B]^q$ there is an element in $X_k^q = \bigotimes_{j=1}^q X_k$ with $\|\cdot\|_\infty$ distance less than ε. Clearly, $|X_k^q| = \lceil B/\varepsilon \rceil^q$, which completes the proof. □

Having established a covering number for $[-B,B]^{n_\mathcal{A}}$ and hence $\mathcal{PN}(\mathcal{A}, B)$, we can now estimate the covering numbers of deep neural networks by combining Lemma 14.12 and Propositions 13.1 and 14.13 .

Theorem 14.14. *Let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma\colon \mathbb{R} \to \mathbb{R}$ be C_σ-Lipschitz continuous with $C_\sigma \geq 1$, let $|\sigma(x)| \leq C_\sigma |x|$ for all $x \in \mathbb{R}$, and let $B \geq 1$. Then*

$$\begin{aligned}\mathcal{G}(\mathcal{N}(\sigma; \mathcal{A}, B), \varepsilon, L^\infty([0,1]^{d_0})) &\leq \mathcal{G}([-B,B]^{n_\mathcal{A}}, \varepsilon/(2C_\sigma B d_{\max})^L, (\mathbb{R}^{n_\mathcal{A}}, \|\cdot\|_\infty)) \\ &\leq \lceil n_\mathcal{A}/\varepsilon \rceil^{n_\mathcal{A}} \lceil 2C_\sigma B d_{\max} \rceil^{n_\mathcal{A} L}.\end{aligned}$$

We end this section, by applying the previous theorem to the generalization bound of Theorem 14.11 with $\alpha = 1/2$. To simplify the analysis, we restrict the discussion to neural networks with range $[-1, 1]$. To this end, denote

$$\mathcal{N}^*(\sigma; \mathcal{A}, B) := \big\{\Phi \in \mathcal{N}(\sigma; \mathcal{A}, B)\big| \\ \Phi(\boldsymbol{x}) \in [-1, 1] \text{ for all } \boldsymbol{x} \in [0, 1]^{d_0}\big\}. \tag{14.5.1}$$

Since $\mathcal{N}^*(\sigma; \mathcal{A}, B) \subseteq \mathcal{N}(\sigma; \mathcal{A}, B)$ we can bound the covering numbers of $\mathcal{N}^*(\sigma; \mathcal{A}, B)$ by those of $\mathcal{N}(\sigma; \mathcal{A}, B)$. This yields the following result.

Theorem 14.15. *Let $C_{\mathcal{L}} > 0$ and let $\mathcal{L}\colon [-1, 1] \times [-1, 1] \to \mathbb{R}$ be $C_{\mathcal{L}}$-Lipschitz continuous. Further, let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma\colon \mathbb{R} \to \mathbb{R}$ be C_σ-Lipschitz continuous with $C_\sigma \geq 1$, and $|\sigma(x)| \leq C_\sigma |x|$ for all $x \in \mathbb{R}$, and let $B \geq 1$.*

Then, for every $m \in \mathbb{N}$, and every distribution $\mathcal{D}$ on $X \times [-1, 1]$ it holds with probability at least $1 - \delta$ over $S \sim \mathcal{D}^m$ that for all $\Phi \in \mathcal{N}^*(\sigma; \mathcal{A}, B)$

$$|\mathcal{R}(\Phi) - \widehat{\mathcal{R}}_S(\Phi)| \leq 4C_{\mathcal{L}} \sqrt{\frac{n_{\mathcal{A}} \log(\lceil n_{\mathcal{A}} \sqrt{m} \rceil) + L n_{\mathcal{A}} \log(\lceil 2C_\sigma B d_{\max} \rceil) + \log(2/\delta)}{m}} \\ + \frac{2C_{\mathcal{L}}}{\sqrt{m}}.$$

14.6 The approximation-complexity trade-off

We recall the decomposition of the error in (14.2.3)

$$\mathcal{R}(h_S) - R^* \leq 2\varepsilon_{\text{gen}} + \varepsilon_{\text{approx}},$$

where R_* is the Bayes risk defined in (14.1.1). We make the following observations about the approximation error $\varepsilon_{\text{approx}}$ and generalization error ε_{gen} in the context of neural network based learning:

- *Scaling of generalization error:* By Theorem 14.15, for a hypothesis class $\mathcal{H}$ of neural networks with $n_{\mathcal{A}}$ weights and L layers, and for sample of size $m \in \mathbb{N}$, the generalization error ε_{gen} essentially scales like

$$\varepsilon_{\text{gen}} = \mathcal{O}(\sqrt{(n_{\mathcal{A}} \log(n_{\mathcal{A}} m) + L n_{\mathcal{A}} \log(n_{\mathcal{A}}))/m}) \qquad \text{as } m \to \infty.$$

- *Scaling of approximation error:* Assume there exists h^* such that $\mathcal{R}(h^*) = R^*$, and let the loss function $\mathcal{L}$ be Lipschitz continuous in the first coordinate. Then

$$\varepsilon_{\text{approx}} = \inf_{h \in \mathcal{H}} \mathcal{R}(h) - \mathcal{R}(h^*) = \inf_{h \in \mathcal{H}} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(h(x), y) - \mathcal{L}(h^*(x), y)] \\ \leq C \inf_{h \in \mathcal{H}} \|h - h^*\|_{L^\infty},$$

for some constant $C > 0$. We have seen in Chapters 5 and 7 that if we choose $\mathcal{H}$ as a set of neural networks with size $n_{\mathcal{A}}$ and L layers, then, for appropriate activation functions, $\inf_{h\in\mathcal{H}} \|h - h^*\|_{L^\infty}$ behaves like ${n_{\mathcal{A}}}^{-r}$ if, e.g., h^* is a d-dimensional s-Hölder regular function and $r = s/d$ (Theorem 5.22), or $h^* \in C^{k,s}([0,1]^d)$ and $r < (k+s)/d$ (Theorem 7.7).

By these considerations, we conclude that for an empirical risk minimizer Φ_S from a set of neural networks with $n_{\mathcal{A}}$ weights and L layers, it holds that

$$\mathcal{R}(\Phi_S) - R^* \leq \mathcal{O}(\sqrt{(n_{\mathcal{A}} \log(m) + L n_{\mathcal{A}} \log(n_{\mathcal{A}}))/m}) + \mathcal{O}({n_{\mathcal{A}}}^{-r}), \tag{14.6.1}$$

for $m \to \infty$ and for some r depending on the regularity of h^*. Note that, enlarging the neural network set, i.e., increasing $n_{\mathcal{A}}$ has two effects: The term associated to approximation decreases, and the term associated to generalization increases. This trade-off is known as **approximation-complexity trade-off**. The situation is depicted in Figure 14.4. The figure and (14.6.1) suggest that, the perfect model, achieves the optimal trade-off between approximation and generalization error. Using this notion, we can also separate all models into three classes:

- *Underfitting*: If the approximation error decays faster than the estimation error increases.
- *Optimal*: If the sum of approximation error and generalization error is at a minimum.
- *Overfitting:* If the approximation error decays slower than the estimation error increases.

In Chapter 15, we will see that deep learning often operates in the regime where the number of parameters $n_{\mathcal{A}}$ exceeds the optimal trade-off point. For certain architectures used in practice, $n_{\mathcal{A}}$ can be so large that the theory of the approximation-complexity trade-off suggests that learning should be impossible. However, we emphasize, that the present analysis only provides upper bounds. It does not prove that learning is impossible or even impractical in the overparameterized regime. Moreover, in Chapter 11 we have already seen indications that learning in the overparametrized regime need not necessarily lead to large generalization errors.

14.7 PAC learning from VC dimension

In addition to covering numbers, there are several other tools to analyze the generalization capacity of hypothesis sets. In the context of classification problems, one of the most important is the so-called Vapnik–Chervonenkis (VC) dimension.

14.7.1 Definition and examples

Let $\mathcal{H}$ be a hypothesis set of functions mapping from $\mathbb{R}^d$ to $\{0,1\}$. A set $S = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subseteq \mathbb{R}^d$ is said to be **shattered** by $\mathcal{H}$ if for every $(y_1, \dots, y_n) \in \{0,1\}^n$ there exists $h \in \mathcal{H}$ such that $h(\boldsymbol{x}_j) = y_j$ for all $j \in \mathbb{N}$.

The VC dimension quantifies the complexity of a function class via the number of points that can in principle be shattered.

Definition 14.16. The **VC dimension** of $\mathcal{H}$ is the cardinality of the largest set $S \subseteq \mathbb{R}^d$ that is shattered by $\mathcal{H}$. We denote the VC dimension by $\mathrm{VCdim}(\mathcal{H})$.

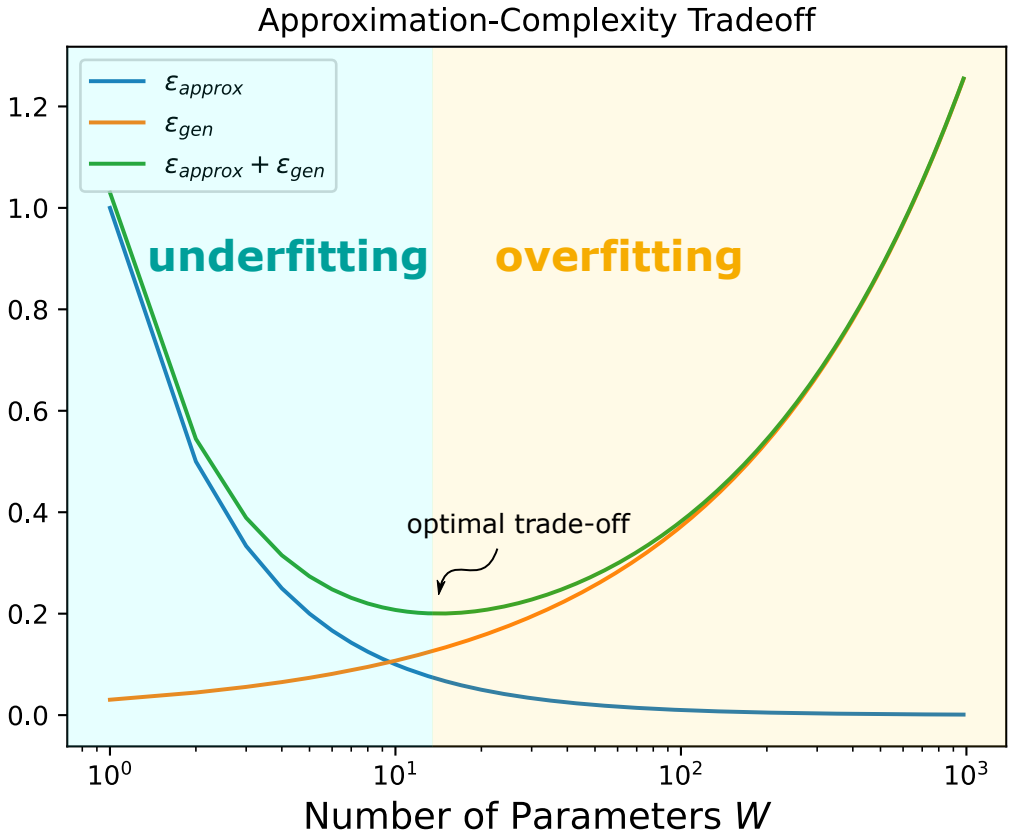


Figure 14.4: Illustration of the approximation-complexity-trade-off of Equation (14.6.1). Here we chose $r = 1$ and $m = 10.000$, also all implicit constants are assumed to be equal to 1.

Example 14.17 (Intervals)**.** Let $\mathcal{H} = \{\mathbb{1}_{[a,b]} \,|\, a, b \in \mathbb{R}\}$. It is clear that $\mathrm{VCdim}(\mathcal{H}) \geq 2$ since for $x_1 < x_2$ the functions

$$\mathbb{1}_{[x_1-2,x_1-1]}, \quad \mathbb{1}_{[x_1-1,x_1]}, \quad \mathbb{1}_{[x_1,x_2]}, \quad \mathbb{1}_{[x_2,x_2+1]},$$

are all different, when restricted to $S = (x_1, x_2)$.

On the other hand, if $x_1 < x_2 < x_3$ then, since $h^{-1}(\{1\})$ is an interval for all $h \in \mathcal{H}$ we have that $h(x_1) = 1 = h(x_3)$ implies $h(x_2) = 1$. Hence, no set of three elements can be shattered. Therefore, $\mathrm{VCdim}(\mathcal{H}) = 2$. The situation is depicted in Figure 14.5.

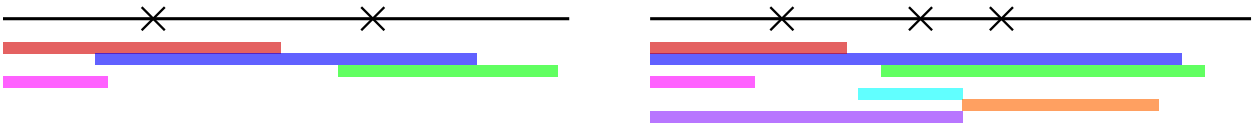


Figure 14.5: Different ways to classify two or three points. The colored-blocks correspond to intervals that produce different classifications of the points.

Example 14.18 (Half-spaces)**.** Let $\mathcal{H}_2 = \{\mathbb{1}_{[0,\infty)}(\langle \boldsymbol{w}, \cdot \rangle + b) \,|\, \boldsymbol{w} \in \mathbb{R}^2, b \in \mathbb{R}\}$ be a hypothesis set of rotated and shifted two-dimensional half-spaces. In Figure 14.6 we see that $\mathcal{H}_2$ shatters a set of

three points. More general, for $d \geq 2$ with

$$\mathcal{H}_d := \{\boldsymbol{x} \mapsto \mathbb{1}_{[0,\infty)}(\boldsymbol{w}^\top \boldsymbol{x} + b) \,|\, \boldsymbol{w} \in \mathbb{R}^d,\ b \in \mathbb{R}\}$$

the VC dimension of $\mathcal{H}_d$ equals $d + 1$.

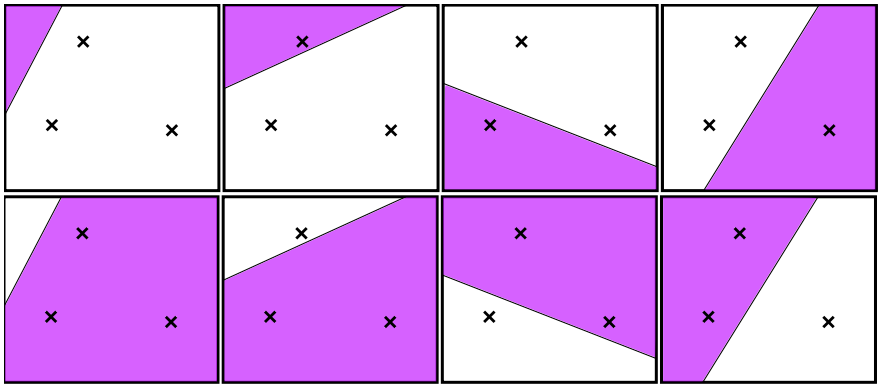


Figure 14.6: Different ways to classify three points by a half-space.

In the example above, the VC dimension coincides with the number of parameters. However, this is not true in general as the following example shows.

Example 14.19 (Infinite VC dimension)**.** Let for $x \in \mathbb{R}$

$$\mathcal{H} := \{x \mapsto \mathbb{1}_{[0,\infty)}(\sin(wx)) \,|\, w \in \mathbb{R}\}.$$

Then the VC dimension of $\mathcal{H}$ is infinite (Exercise 14.29).

14.7.2 Generalization based on VC dimension

In the following, we consider a classification problem. Denote by $\mathcal{D}$ the data-generating distribution on $\mathbb{R}^d \times \{0, 1\}$. Moreover, we let $\mathcal{H}$ be a set of functions from $\mathbb{R}^d \to \{0, 1\}$.

In the binary classification set-up, the natural choice of a loss function is the $0 - 1$ loss $\mathcal{L}_{0-1}(y, y') = \mathbb{1}_{y \neq y'}$. Thus, given a sample S, the empirical risk of a function $h \in \mathcal{H}$ is

$$\widehat{\mathcal{R}}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{h(\boldsymbol{x}_i) \neq y_i}.$$

Moreover, the risk can be written as

$$\mathcal{R}(h) = \mathbb{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}[h(\boldsymbol{x}) \neq y],$$

i.e., the probability under $(\boldsymbol{x}, y) \sim \mathcal{D}$ of h misclassifying the label y of $\boldsymbol{x}$.

We can now give a generalization bound in terms of the VC dimension of $\mathcal{H}$, see, e.g., [148, Corollary 3.19]:

Theorem 14.20. *Let $d, k \in \mathbb{N}$ and $\mathcal{H} \subseteq \{h\colon \mathbb{R}^d \to \{0,1\}\}$ have VC dimension k. Let $\mathcal{D}$ be a distribution on $\mathbb{R}^d \times \{0,1\}$. Then, for every $\delta > 0$ and $m \in \mathbb{N}$, it holds with probability at least $1-\delta$ over a sample $S \sim \mathcal{D}^m$ that for every $h \in \mathcal{H}$*

$$|\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| \leq \sqrt{\frac{2k\log(em/k)}{m}} + \sqrt{\frac{\log(1/\delta)}{2m}}. \tag{14.7.1}$$

In words, Theorem 14.20 tells us that if a hypothesis class has finite VC dimension, then a hypothesis with a small empirical risk will have a small risk if the number of samples is large. This shows that empirical risk minimization is a viable strategy in this scenario. Will this approach also work if the VC dimension is not bounded? No, in fact, in that case, no learning algorithm will succeed in reliably producing a hypothesis for which the risk is close to the best possible. We omit the technical proof of the following theorem from [148, Theorem 3.23].

Theorem 14.21. *Let $k \in \mathbb{N}$ and let $\mathcal{H} \subseteq \{h\colon X \to \{0,1\}\}$ be a hypothesis set with VC dimension k. Then, for every $m \in \mathbb{N}$ and every learning algorithm $\mathrm{A}\colon (X \times \{0,1\})^m \to \mathcal{H}$ there exists a distribution $\mathcal{D}$ on $X \times \{0,1\}$ such that*

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left[\mathcal{R}(\mathrm{A}(S)) - \inf_{h\in\mathcal{H}} \mathcal{R}(h) > \sqrt{\frac{k}{320m}}\right] \geq \frac{1}{64}.$$

Theorem 14.21 immediately implies the following statement for the generalization bound.

Corollary 14.22. *Let $k \in \mathbb{N}$ and let $\mathcal{H} \subseteq \{h\colon X \to \{0,1\}\}$ be a hypothesis set with VC dimension k. Then, for every $m \in \mathbb{N}$ there exists a distribution $\mathcal{D}$ on $X \times \{0,1\}$ such that*

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left[\sup_{h\in\mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| > \sqrt{\frac{k}{1280m}}\right] \geq \frac{1}{64}.$$

Proof. For a sample S, let $h_S \in \mathcal{H}$ be an empirical risk minimizer, i.e., $\widehat{\mathcal{R}}_S(h_S) = \min_{h\in\mathcal{H}} \widehat{\mathcal{R}}_S(h)$. Let $\mathcal{D}$ be the distribution of Theorem 14.21. Moreover, for $\delta > 0$, let $h_\delta \in \mathcal{H}$ be such that

$$\mathcal{R}(h_\delta) - \inf_{h\in\mathcal{H}} \mathcal{R}(h) < \delta.$$

Then, applying Theorem 14.21 with $\mathrm{A}(S) = h_S$ it holds that

$$\begin{aligned} 2 \sup_{h \in \mathcal{H}} |\mathcal{R}(h) - \widehat{\mathcal{R}}_S(h)| &\geq |\mathcal{R}(h_S) - \widehat{\mathcal{R}}_S(h_S)| + |\mathcal{R}(h_\delta) - \widehat{\mathcal{R}}_S(h_\delta)| \\ &\geq \mathcal{R}(h_S) - \widehat{\mathcal{R}}_S(h_S) + \widehat{\mathcal{R}}_S(h_\delta) - \mathcal{R}(h_\delta) \\ &\geq \mathcal{R}(h_S) - \mathcal{R}(h_\delta) \\ &> \mathcal{R}(h_S) - \inf_{h \in \mathcal{H}} \mathcal{R}(h) - \delta, \end{aligned}$$

where we used the definition of h_S in the third inequality. The proof is completed by applying Theorem 14.21 and using that δ was arbitrary. □

We have seen now, that we have a generalization bound scaling like $\mathcal{O}(1/\sqrt{m})$ for $m \to \infty$ if and only if the VC dimension of a hypothesis class is finite. In more quantitative terms, we require the VC dimension of a neural network to be smaller than m.

What does this imply for neural network functions? For ReLU neural networks there holds the following [3, Theorem 8.8].

Theorem 14.23. *Let $\mathcal{A} \in \mathbb{N}^{L+2}$, $L \in \mathbb{N}$ and set*

$$\mathcal{H} := \{\mathbb{1}_{[0,\infty)} \circ \Phi \,|\, \Phi \in \mathcal{N}(\sigma_{\mathrm{ReLU}}; \mathcal{A}, \infty)\}.$$

Then, there exists a constant $C > 0$ independent of L and $\mathcal{A}$ such that

$$\mathrm{VCdim}(\mathcal{H}) \leq C \cdot (n_{\mathcal{A}} L \log(n_{\mathcal{A}}) + n_{\mathcal{A}} L^2).$$

The bound (14.7.1) is meaningful if $m \gg k$. For ReLU neural networks as in Theorem 14.23, this means $m \gg n_{\mathcal{A}} L \log(n_{\mathcal{A}}) + n_{\mathcal{A}} L^2$. Fixing $L = 1$ this amounts to $m \gg n_{\mathcal{A}} \log(n_{\mathcal{A}})$ for a shallow neural network with $n_{\mathcal{A}}$ parameters. This condition is contrary to what we assumed in Chapter 11, where it was crucial that $n_{\mathcal{A}} \gg m$. If the VC dimension of the neural network sets scale like $\mathcal{O}(n_{\mathcal{A}} \log(n_{\mathcal{A}}))$, then Theorem 14.21 and Corollary 14.22 indicate that, at least for certain distributions, generalization should not be possible in this regime. We will discuss the resolution of this potential paradox in Chapter 15.

14.8 Lower bounds on achievable approximation rates

We conclude this chapter on the complexities and generalization bounds of neural networks by using the established VC dimension bound of Theorem 14.23 to deduce limitations to the approximation capacity of neural networks. The result described below was first given in [245].

Theorem 14.24. *Let k, $d \in \mathbb{N}$. Assume that for every $\varepsilon > 0$ there exists $L_\varepsilon \in \mathbb{N}$ and $\mathcal{A}_\varepsilon$ with L_ε layers and input dimension d such that*

$$\sup_{\|f\|_{C^k([0,1]^d)} \leq 1} \inf_{\Phi \in \mathcal{N}(\sigma_{\mathrm{ReLU}}; \mathcal{A}, \infty)} \|f - \Phi\|_{C^0([0,1]^d)} < \frac{\varepsilon}{2}.$$

Then there exists $C > 0$ solely depending on k and d, such that for all $\varepsilon \in (0,1)$

$$n_{\mathcal{A}_\varepsilon} L_\varepsilon \log(n_{\mathcal{A}_\varepsilon}) + n_{\mathcal{A}_\varepsilon} L_\varepsilon^2 \geq C\varepsilon^{-\frac{d}{k}}.$$

Proof. For $\boldsymbol{x} \in \mathbb{R}^d$ consider the "bump function"

$$\tilde{f}(\boldsymbol{x}) := \begin{cases} \exp\left(1 - \frac{1}{1-\|\boldsymbol{x}\|_2^2}\right) & \text{if } \|\boldsymbol{x}\|_2 < 1 \\ 0 & \text{otherwise,} \end{cases}$$

and its scaled version

$$\tilde{f}_\varepsilon(\boldsymbol{x}) \coloneqq \varepsilon f\left(2\varepsilon^{-1/k}\boldsymbol{x}\right),$$

for $\varepsilon \in (0,1)$. Then

$$\operatorname{supp}(\tilde{f}_\varepsilon) \subseteq \Big[-\frac{\varepsilon^{1/k}}{2}, \frac{\varepsilon^{1/k}}{2}\Big]^d$$

and

$$\|\tilde{f}_\varepsilon\|_{C^k} \leq 2^k \|\tilde{f}\|_{C^k} =: \tau_k > 0.$$

Consider the equispaced point set $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{N(\varepsilon)}\} = \varepsilon^{1/k}\mathbb{Z}^d \cap [0,1]^d$. The cardinality of this set is $N(\varepsilon) \simeq \varepsilon^{-d/k}$. Given $\boldsymbol{y} \in \{0,1\}^{N(\varepsilon)}$, let for $\boldsymbol{x} \in \mathbb{R}^d$

$$f_{\boldsymbol{y}}(\boldsymbol{x}) \coloneqq \tau_k^{-1} \sum_{j=1}^{N(\varepsilon)} y_j \tilde{f}_\varepsilon(\boldsymbol{x} - \boldsymbol{x}_j). \tag{14.8.1}$$

Then $f_{\boldsymbol{y}}(\boldsymbol{x}_j) = \tau_k^{-1}\varepsilon y_j$ for all $j = 1, \dots, N(\varepsilon)$ and $\|f_{\boldsymbol{y}}\|_{C^k} \leq 1$.

For every $\boldsymbol{y} \in \{0,1\}^{N(\varepsilon)}$ let $\Phi_{\boldsymbol{y}} \in \mathcal{N}(\sigma_{\mathrm{ReLU}}; \mathcal{A}_{\tau_k^{-1}\varepsilon}, \infty)$ be such that

$$\sup_{\boldsymbol{x} \in [0,1]^d} |f_{\boldsymbol{y}}(\boldsymbol{x}) - \Phi_{\boldsymbol{y}}(\boldsymbol{x})| < \frac{\varepsilon}{2\tau_k}.$$

Then

$$\mathbb{1}_{[0,\infty)}\Big(\Phi_y(\boldsymbol{x}_j) - \frac{\varepsilon}{2\tau_k}\Big) = y_j \qquad \text{for all } j = 1, \dots, N(\varepsilon).$$

Hence, the VC dimension of $\mathcal{N}(\sigma_{\mathrm{ReLU}}; \mathcal{A}_{\tau_k^{-1}\varepsilon}, \infty)$ is larger or equal to $N(\varepsilon)$. Theorem 14.23 thus implies

$$N(\varepsilon) \simeq \varepsilon^{-\frac{d}{k}} \leq C \cdot \Big(n_{\mathcal{A}_{\tau_k^{-1}\varepsilon}} L_{\tau_k^{-1}\varepsilon} \log(n_{\mathcal{A}_{\tau_k^{-1}\varepsilon}}) + n_{\mathcal{A}_{\tau_k^{-1}\varepsilon}} L^2_{\tau_k^{-1}\varepsilon}\Big)$$

or equivalently

$$\tau_k^{\frac{d}{k}} \varepsilon^{-\frac{d}{k}} \leq C \cdot \Big(n_{\mathcal{A}_{\tau_k^{-1}\varepsilon}} L_\varepsilon \log(n_{\mathcal{A}_{\tau_k^{-1}\varepsilon}}) + n_{\mathcal{A}_{\tau_k^{-1}\varepsilon}} L_\varepsilon^2\Big).$$

This completes the proof. □

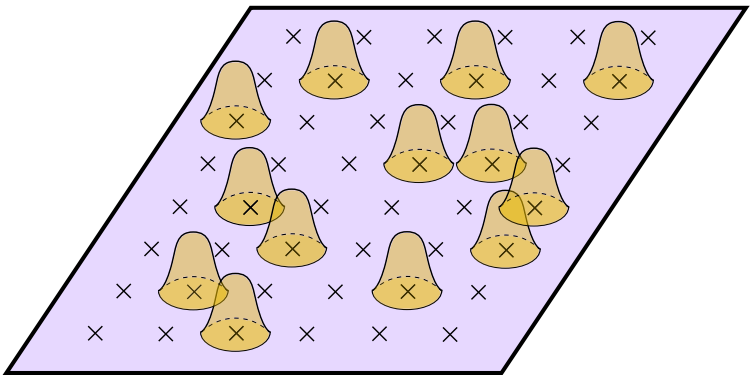


Figure 14.7: Illustration of $f_{\boldsymbol{y}}$ from Equation (14.8.1) on $[0,1]^2$.

To interpret Theorem 14.24, we consider two situations:

- In case the depth is allowed to increase at most logarithmically in ε, then reaching uniform error ε for all $f \in C^k([0,1]^d)$ with $\|f\|_{C^k([0,1]^d)} \leq 1$ requires

$$n_{\mathcal{A}_\varepsilon} \log(n_{\mathcal{A}_\varepsilon}) \log(\varepsilon) + n_{\mathcal{A}_\varepsilon} \log(\varepsilon)^2 \geq C\varepsilon^{-\frac{d}{k}}.$$

  In terms of the neural network size, this (necessary) condition becomes $n_{\mathcal{A}_\varepsilon} \geq C\varepsilon^{-d/k}/\log(\varepsilon)^2$. As we have shown in Chapter 7, in particular Theorem 7.7, up to log terms this condition is also sufficient. Hence, while the constructive proof of Theorem 7.7 might have seemed rather specific, under the assumption of the depth increasing at most logarithmically (which the construction in Chapter 7 satisfies), it was essentially optimal! The neural networks in this proof are shown to have size $O(\varepsilon^{-d/k})$ up to log terms.

- If we allow the depth L_ε to increase faster than logarithmically in ε, then the lower bound on the required neural network size improves. Fixing for example $\mathcal{A}_\varepsilon$ with L_ε layers such that $n_{\mathcal{A}_\varepsilon} \leq WL_\varepsilon$ for some fixed ε independent $W \in \mathbb{N}$, the (necessary) condition on the depth becomes

$$W \log(WL_\varepsilon) L_\varepsilon^2 + WL_\varepsilon^3 \geq C\varepsilon^{-\frac{d}{k}}$$

  and hence $L_\varepsilon \gtrsim \varepsilon^{-d/(3k)}$.

  We add that, for arbitrary depth the upper bound on the VC dimension of Theorem 14.23 can be improved to $n_{\mathcal{A}}^2$, [3, Theorem 8.6], and using this, would improve the just established lower bound to $L_\varepsilon \gtrsim \varepsilon^{-d/(2k)}$.

  For fixed width, this corresponds to neural networks of size $O(\varepsilon^{-d/(2k)})$, which would mean twice the convergence rate proven in Theorem 7.7. Indeed, it turns out that neural networks can achieve this rate in terms of the neural network size [246].

To sum up, in order to get error ε uniformly for all $\|f\|_{C^k([0,1]^d)} \leq 1$, the size of a ReLU neural network is required to increase at least like $O(\varepsilon^{-d/(2k)})$ as $\varepsilon \to 0$, i.e. the best possible attainable convergence rate is $2k/d$. It has been proven, that this rate is also achievable, and thus the bound is sharp. Achieving this rate requires neural network architectures that grow faster in depth than in width.

Bibliography and further reading

Classical statistical learning theory is based on the foundational work of Vapnik and Chervonenkis [233]. This led to the formulation of the probably approximately correct (PAC) learning model in [232], which is primarily utilized in this chapter. A streamlined mathematical introduction to statistical learning theory can be found in [43].

Since statistical learning theory is well-established, there exists a substantial amount of excellent expository work describing this theory. Some highly recommended books on the topic are [148, 212, 3]. The specific approach of characterizing learning via covering numbers has been discussed extensively in [3, Chapter 14]. Specific results for ReLU activation used in this chapter were derived in [204, 18]. The results of Section 14.8 describe some of the findings in [245, 246], and we also refer to [51] for general lower bounds (also applicable to neural networks) when approximating classes of Sobolev functions.

Exercises

Exercise 14.25. Let $\mathcal{H}$ be a set of neural networks with fixed architecture, where the weights are taken from a compact set. Moreover, assume that the activation function is continuous. Show that for every sample S there always exists an empirical risk minimizer h_S.

Exercise 14.26. Complete the proof of Proposition 14.9.

Exercise 14.27. Prove Lemma 14.12.

Exercise 14.28. Show that, the VC dimension of $\mathcal{H}$ of Example 14.18 is indeed 3, by demonstrating that no set of four points can be shattered by $\mathcal{H}$.

Exercise 14.29. Show that the VC dimension of

$$\mathcal{H} := \{x \mapsto \mathbb{1}_{[0,\infty)}(\sin(wx)) \,|\, w \in \mathbb{R}\}$$

is infinite.

Chapter 15

Generalization in the overparameterized regime

In the previous chapter, we discussed the theory of generalization for deep neural networks trained by minimizing the empirical risk. A key conclusion was that good generalization is possible as long as we choose an architecture that has a moderate number of neural network parameters relative to the number of training samples. Moreover, we saw in Section 14.6 that the best performance can be expected when the neural network size is chosen to balance the generalization and approximation errors, by minimizing their sum.

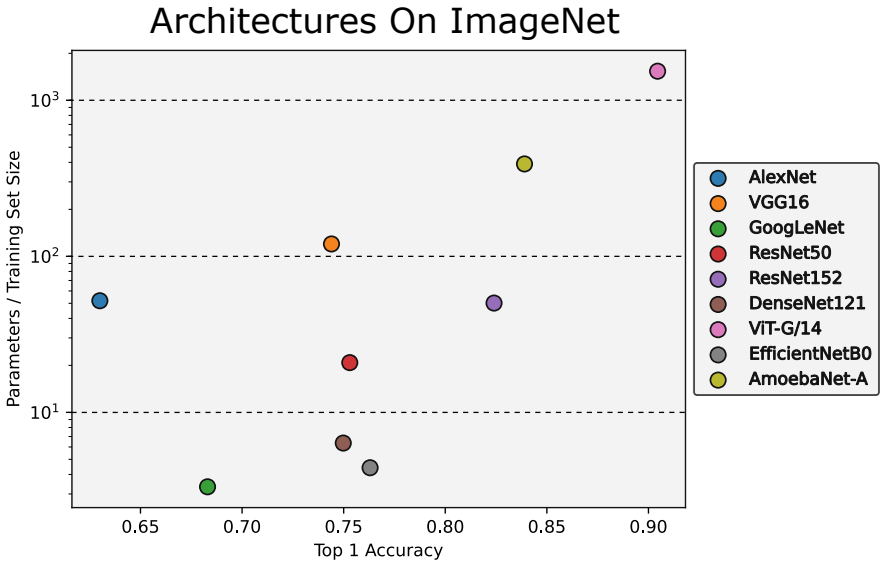


Figure 15.1: ImageNet Classification Competition: Final score on the test set in the Top 1 category vs. Parameters-to-Training-Samples Ratio. Note that all architectures have more parameters than training samples. Architectures include AlexNet [121], VGG16 [215], GoogLeNet [222], ResNet50/ResNet152 [87], DenseNet121 [96], ViT-G/14 [248], EfficientNetB0 [224], and AmoebaNet [189].

Surprisingly, successful neural network architectures do not necessarily follow these theoretical observations. Consider the neural network architectures in Figure 15.1. They represent some

of the most renowned image classification models, and all of them participated in the ImageNet Classification Competition [50]. The training set consisted of 1.2 million images. The x-axis shows the model performance, and the y-axis displays the ratio of the number of parameters to the size of the training set; notably, all architectures have a ratio larger than one, i.e. have more parameters than training samples. For the largest model, there are by a factor 1000 more neural network parameters than training samples.

Given that the practical application of deep learning appears to operate in a regime significantly different from the one analyzed in Chapter 14, we must ask: Why do these methods still work effectively?

15.1 The double descent phenomenon

The success of deep learning in a regime not covered by traditional statistical learning theory puzzled researchers for some time. In [14], an intriguing set of experiments was performed. These experiments indicate that while the risk follows the upper bound from Section 14.6 for neural network architectures that do not interpolate the data, the curve does not expand to infinity in the way that Figure 14.4 suggests. Instead, after surpassing the so-called "interpolation threshold", the risk starts to decrease again. This behavior, known as double descent, is illustrated in Figure 15.2.

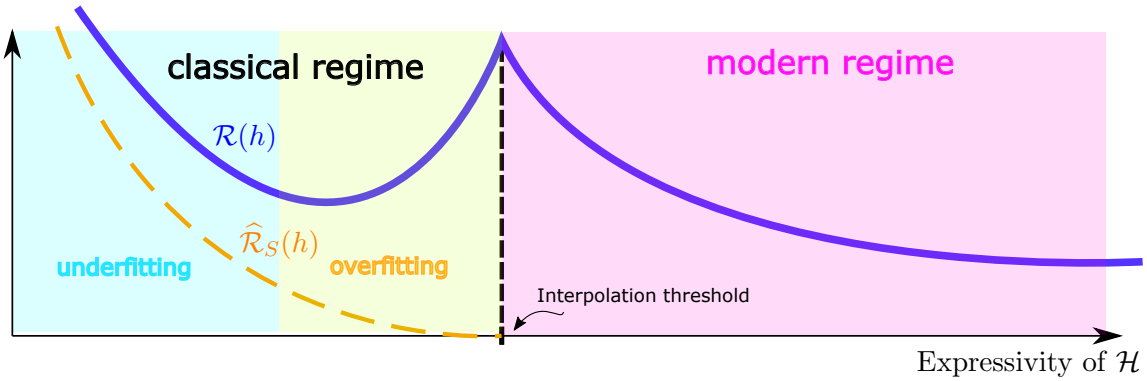


Figure 15.2: Illustration of the double descent phenomenon.

15.1.1 Least-squares regression revisited

To gain further insight, we consider least-squares (kernel) regression as introduced in Section 11.2. Consider a data sample $(\boldsymbol{x}_j, y_j)_{j=1}^m \subseteq \mathbb{R}^d \times \mathbb{R}$ generated by some ground-truth function f, i.e.

$$y_j = f(\boldsymbol{x}_j) \qquad \text{for } j = 1, \dots, m. \tag{15.1.1}$$

Let $\phi_j : \mathbb{R}^d \to \mathbb{R}$, $j \in \mathbb{N}$, be a sequence of *ansatz functions*. For $n \in \mathbb{N}$, we wish to fit a function $\boldsymbol{x} \mapsto \sum_{i=1}^n w_i \phi_i(\boldsymbol{x})$ to the data using linear least-squares. To this end, we introduce the feature map

$$\mathbb{R}^d \ni \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) := (\phi_1(\boldsymbol{x}), \dots, \phi_n(\boldsymbol{x}))^\top \in \mathbb{R}^n.$$

The goal is to determine coefficients $\boldsymbol{w} \in \mathbb{R}^n$ minimizing the empirical risk

$$\widehat{\mathcal{R}}_S(\boldsymbol{w}) = \frac{1}{m}\sum_{j=1}^{m}\Big(\sum_{i=1}^{n} w_i\phi_i(\boldsymbol{x}_j) - y_j\Big)^2 = \frac{1}{m}\sum_{j=1}^{m}(\langle\phi(\boldsymbol{x}_j), \boldsymbol{w}\rangle - y_j)^2.$$

With

$$\boldsymbol{A}_n := \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \dots & \phi_n(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_m) & \dots & \phi_n(\boldsymbol{x}_m) \end{pmatrix} = \begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \\ \vdots \\ \phi(\boldsymbol{x}_m)^\top \end{pmatrix} \in \mathbb{R}^{m\times n} \tag{15.1.2}$$

and $\boldsymbol{y} = (y_1, \dots, y_m)^\top$ it holds

$$\widehat{\mathcal{R}}_S(\boldsymbol{w}) = \frac{1}{m}\|\boldsymbol{A}_n\boldsymbol{w} - \boldsymbol{y}\|^2. \tag{15.1.3}$$

As discussed in Sections 11.1-11.2, a unique minimizer of (15.1.3) only exists if $\boldsymbol{A}_n$ has rank n. For a minimizer $\boldsymbol{w}_n$, the fitted function reads

$$f_n(x) := \sum_{j=1}^{n} w_{n,j}\phi_j(x). \tag{15.1.4}$$

We are interested in the behavior of the f_n as a function of n (the number of ansatz functions/parameters of our model), and distinguish between two cases:

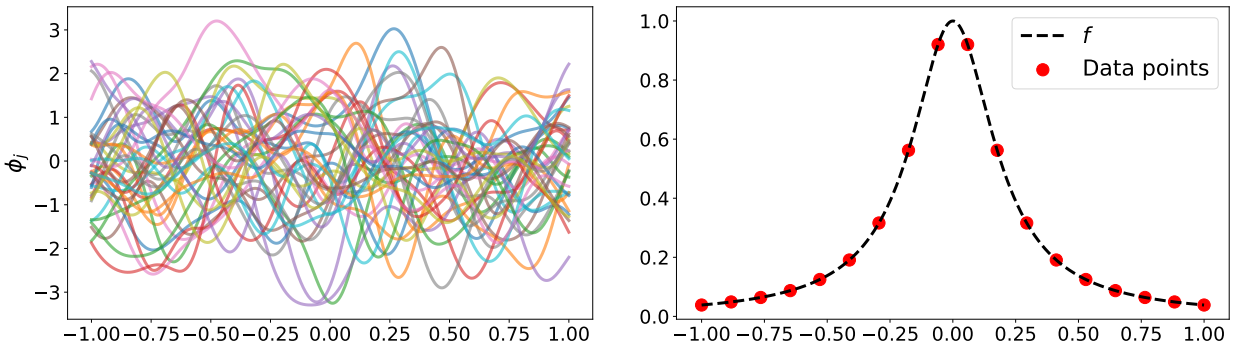
- *Underparameterized*: If $n < m$ we have fewer parameters n than training points m. For the least squares problem of minimizing $\widehat{\mathcal{R}}_S$, this means that there are more conditions m than free parameters n. Thus, in general, we cannot interpolate the data, and we have $\min_{\boldsymbol{w}\in\mathbb{R}^n}\widehat{\mathcal{R}}_S(\boldsymbol{w}) > 0$.
- *Overparameterized*: If $n \geq m$, then we have at least as many parameters n as training points m. If the $\boldsymbol{x}_j$ and the ϕ_j are such that $\boldsymbol{A}_n \in \mathbb{R}^{m\times n}$ has full rank m, then there exists $\boldsymbol{w}$ such that $\widehat{\mathcal{R}}_S(\boldsymbol{w}) = 0$. If $n > m$, then $\boldsymbol{A}_n$ necessarily has a nontrivial kernel, and there exist infinitely many parameters choices $\boldsymbol{w}$ that yield zero empirical risk $\widehat{\mathcal{R}}_S$. Some of them lead to better, and some lead to worse prediction functions f_n in (15.1.4).

In the overparameterized case, there exist many minimizers of $\widehat{\mathcal{R}}_S$. The training algorithm we use to compute a minimizer determines the type of prediction function f_n we obtain. To observe double descent, i.e. to achieve good generalization for large n, we need to choose the minimizer carefully. In the following, we consider the unique minimal 2-norm minimizer, which is defined as

$$\boldsymbol{w}_{n,*} = \Big(\operatorname{argmin}_{\{\boldsymbol{w}\in\mathbb{R}^n \,|\, \widehat{\mathcal{R}}_S(\boldsymbol{w})\leq\widehat{\mathcal{R}}_S(\boldsymbol{v})\ \forall \boldsymbol{v}\in\mathbb{R}^n\}} \|\boldsymbol{w}\|\Big) \in \mathbb{R}^n. \tag{15.1.5}$$

15.1.2 An example

Now let us consider a concrete example. In Figure 15.3 we plot a set of 40 ansatz functions $\phi_1, \dots, \phi_{40}$, which are drawn from a Gaussian process. Additionally, the figure shows a plot of the Runge function f, and $m = 18$ equispaced points which are used as the training data points. We then fit a function in $\operatorname{span}\{\phi_1, \dots, \phi_n\}$ via (15.1.5) and (15.1.4). The result is displayed in Figure 15.4:



(a) ansatz functions ϕ_j

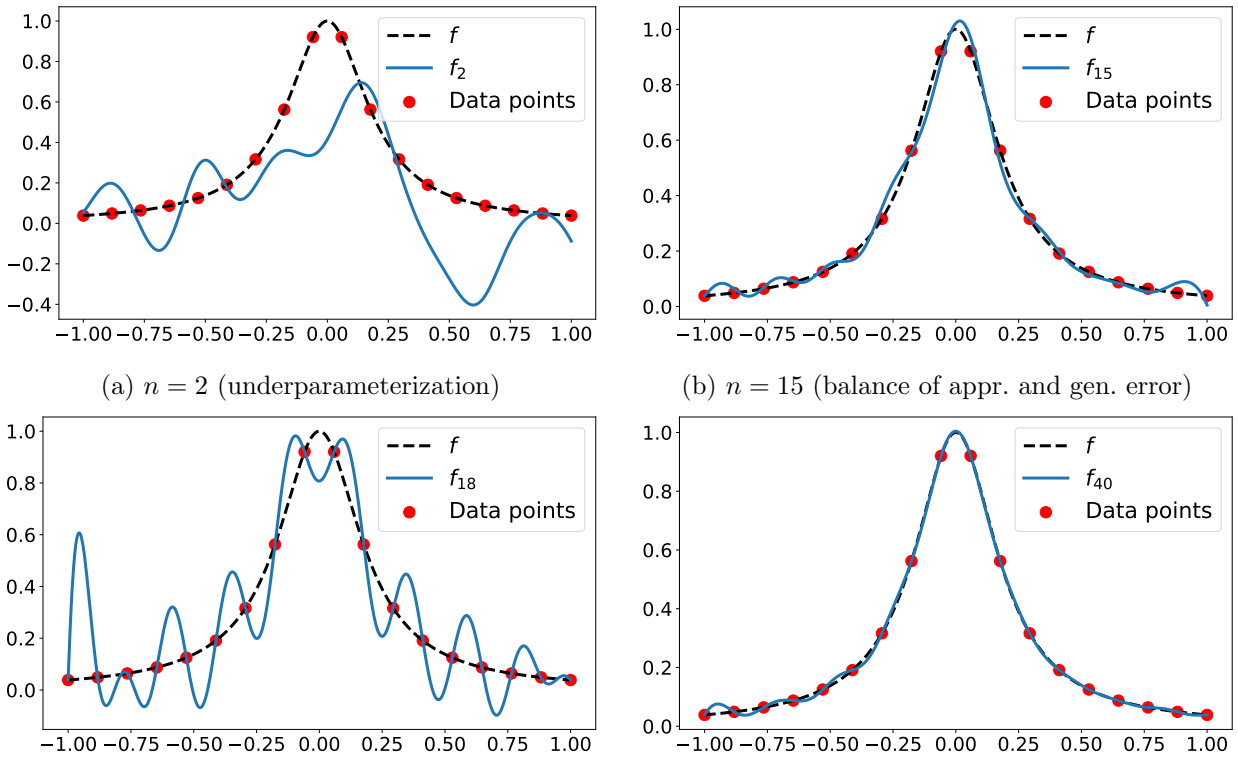
(b) Runge function f and data points

Figure 15.3: Ansatz functions $\phi_1, \dots, \phi_{40}$ drawn from a Gaussian process, along with the Runge function and 18 equispaced data points.

- $n = 2$: The model can only represent functions in $\operatorname{span}\{\phi_1, \phi_2\}$. It is not yet expressive enough to give a meaningful approximation of f.
- $n = 15$: The model has sufficient expressivity to capture the main characteristics of f. Since $n = 15 < 18 = m$, it is not yet able to interpolate the data. Thus it allows to strike a good balanced between the approximation and generalization error, which corresponds to the scenario discussed in Chapter 14.
- $n = 18$: We are at the interpolation threshold. The model is capable of interpolating the data, and there is a unique $\boldsymbol{w}$ such that $\widehat{\mathcal{R}}_S(\boldsymbol{w}) = 0$. Yet, in between data points the behavior of the predictor f_{18} seems erratic, and displays strong oscillations. This is referred to as **overfitting**, and is to be expected due to our analysis in Chapter 14; while the approximation error at the data points has improved compared to the case $n = 15$, the generalization error has gotten worse.
- $n = 40$: This is the overparameterized regime, where we have significantly more parameters than data points. Our prediction f_{40} interpolates the data and appears to be the best overall approximation to f so far, due to a "good" choice of minimizer of $\widehat{\mathcal{R}}_S$, namely (15.1.5). We also note that, while quite good, the fit is not perfect. We cannot expect significant improvement in performance by further increasing n, since at this point the main limiting factor is the amount of available data. Also see Figure 15.5 (a).

Figure 15.5 (a) displays the error $\|f - f_n\|_{L^2([-1,1])}$ over n. We observe the characteristic double descent curve, where the error initially decreases, after peaking at the interpolation threshold, which is marked by the dashed red line. Afterwards, in the overparameterized regime, it starts to decrease again. Figure 15.5 (b) displays $\|\boldsymbol{w}_{n,*}\|$. Note how the Euclidean norm of the coefficient vector also peaks at the interpolation threshold.

We emphasize that the precise nature of the convergence curves depends strongly on various factors, such as the distribution and number of training points m, the ground truth f, and the choice of ansatz functions ϕ_j (e.g., the specific kernel used to generate the ϕ_j in Figure 15.3 (a)). In the present setting we achieve a good approximation of f for $n = 15 < 18 = m$ corresponding to the regime where the approximation and interpolation errors are balanced. However, as Figure 15.5



(a) $n = 2$ (underparameterization)

(b) $n = 15$ (balance of appr. and gen. error)

(c) $n = 18$ (interpolation threshold)

(d) $n = 40$ (overparameterization)

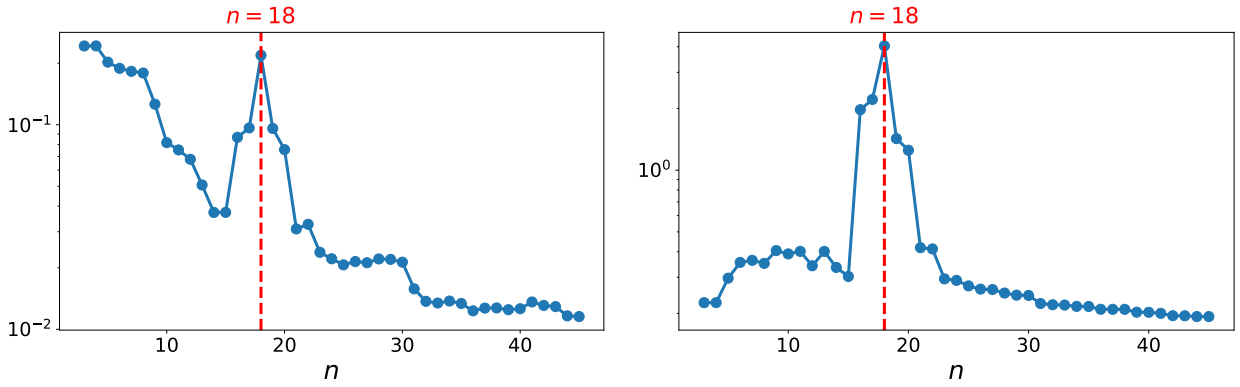
Figure 15.4: Fit of the $m = 18$ red data points using the ansatz functions $\phi_1, \ldots, \phi_n$ from Figure 15.3, employing equations (15.1.5) and (15.1.4) for different numbers of ansatz functions n.

(a) shows, it can be difficult to determine a suitable value of $n < m$ a priori, and the acceptable range of n values can be quite narrow. For overparametrization ($n \gg m$), the precise choice of n is less critical, potentially making the algorithm more stable in this regime. We encourage the reader to conduct similar experiments and explore different settings to get a better feeling for the double descent phenomenon.

15.2 Size of weights

In Figure 15.5, we observed that the norm of the coefficients $\|\boldsymbol{w}_{n,*}\|$ exhibits similar behavior to the L^2-error, peaking at the interpolation threshold $n = 18$. In machine learning, large weights are usually undesirable, as they are associated with large derivatives or oscillatory behavior. This is evident in the example shown in Figure 15.4 for $n = 18$. Assuming that the data in (15.1.1) was generated by a "smooth" function f, e.g. a function with moderate Lipschitz constant, these large derivatives of the prediction function could lead to poor generalization. Such a smoothness assumption about f may or may not be satisfied. However, if f is not smooth, there is little hope of accurately recovering f from limited data (see the discussion in Section 9.2).

The next result gives an explanation for the observed behavior of $\|\boldsymbol{w}_{n,*}\|$.



(a) $\|f - f_n\|_{L^2([-1,1])}$ (b) $\|\boldsymbol{w}_{n,*}\|$

Figure 15.5: The L^2-error for the fitted functions in Figure 15.4, and the ℓ^2-norm of the corresponding coefficient vector $\boldsymbol{w}_{n,*}$ defined in (15.1.5).

Proposition 15.1. *Assume that $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$ and the $(\phi_j)_{j\in\mathbb{N}}$ are such that $\boldsymbol{A}_n$ in (15.1.2) has full rank n for all $n \le m$. Given $\boldsymbol{y} \in \mathbb{R}^m$, denote by $\boldsymbol{w}_{n,*}(\boldsymbol{y})$ the vector in (15.1.5). Then*

$$n \mapsto \sup_{\|\boldsymbol{y}\|=1} \|\boldsymbol{w}_{n,*}(\boldsymbol{y})\| \quad \textit{is monotonically} \quad \begin{cases} \textit{increasing} & \textit{for } n < m, \\ \textit{decreasing} & \textit{for } n \ge m. \end{cases}$$

Proof. We start with the case $n \ge m$. By assumption $\boldsymbol{A}_m$ has full rank m, and thus $\boldsymbol{A}_n$ has rank m for all $n \ge m$, see (15.1.2). In particular, there exists $\boldsymbol{w}_n \in \mathbb{R}^n$ such that $\boldsymbol{A}_n \boldsymbol{w}_n = \boldsymbol{y}$. Now fix $\boldsymbol{y} \in \mathbb{R}^m$ and let $\boldsymbol{w}_n$ be any such vector. Then $\boldsymbol{w}_{n+1} := (\boldsymbol{w}_n, 0) \in \mathbb{R}^{n+1}$ satisfies $\boldsymbol{A}_{n+1}\boldsymbol{w}_{n+1} = \boldsymbol{y}$ and $\|\boldsymbol{w}_{n+1}\| = \|\boldsymbol{w}_n\|$. Thus necessarily $\|\boldsymbol{w}_{n+1,*}\| \le \|\boldsymbol{w}_{n,*}\|$ for the minimal norm solutions defined in (15.1.5). Since this holds for every $\boldsymbol{y}$, we obtain the statement for $n \ge m$.

Now let $n < m$. Recall that the minimal norm solution can be written through the pseudo inverse

$$\boldsymbol{w}_{n,*}(\boldsymbol{y}) = \boldsymbol{A}_n^\dagger \boldsymbol{y},$$

see for instance Exercise 11.32. Here,

$$\boldsymbol{A}_n^\dagger = \boldsymbol{V}_n \begin{pmatrix} \sigma_{n,1}^{-1} & & 0 & \\ & \ddots & & \ddots \\ & & \sigma_{n,n}^{-1} & & 0 \end{pmatrix} \boldsymbol{U}_n^\top \in \mathbb{R}^{n\times m}$$

where $\boldsymbol{A}_n = \boldsymbol{U}_n \boldsymbol{\Sigma}_n \boldsymbol{V}_n^\top$ is the singular value decomposition of $\boldsymbol{A}_n$, and

$$\boldsymbol{\Sigma}_n = \begin{pmatrix} \sigma_{n,1} & & & \\ & \ddots & & \\ & & \sigma_{n,n} & \\ 0 & & & \\ & \ddots & & \\ & & & 0 \end{pmatrix} \in \mathbb{R}^{m \times n}$$

contains the singular values $\sigma_{n,1} \geq \cdots \geq \sigma_{n,n} > 0$ of $\boldsymbol{A}_n \in \mathbb{R}^{m \times n}$ ordered by decreasing size. Since $\boldsymbol{V}_n \in \mathbb{R}^{n \times n}$ and $\boldsymbol{U}_n \in \mathbb{R}^{m \times m}$ are orthogonal matrices, we have

$$\sup_{\|\boldsymbol{y}\|=1} \|\boldsymbol{w}_{n,*}(\boldsymbol{y})\| = \sup_{\|\boldsymbol{y}\|=1} \|\boldsymbol{A}_n^\dagger \boldsymbol{y}\| = \sigma_{n,n}^{-1}.$$

Finally, since the minimal singular value $\sigma_{n,n}$ of $\boldsymbol{A}_n$ can be written as

$$\sigma_{n,n} = \inf_{\substack{\boldsymbol{x} \in \mathbb{R}^n \\ \|\boldsymbol{x}\|=1}} \|\boldsymbol{A}_n \boldsymbol{x}\| \geq \inf_{\substack{\boldsymbol{x} \in \mathbb{R}^{n+1} \\ \|\boldsymbol{x}\|=1}} \|\boldsymbol{A}_{n+1} \boldsymbol{x}\| = \sigma_{n+1,n+1},$$

we observe that $n \mapsto \sigma_{n,n}$ is monotonically decreasing for $n \leq m$. This concludes the proof. □

15.3 Theoretical justification

Let us now examine one possible explanation of the double descent phenomenon for neural networks. While there are many alternative arguments available in the literature (see the bibliography section), the explanation presented here is based on a simplification of the ideas in [12].

The key assumption underlying our analysis is that large overparameterized neural networks tend to be Lipschitz continuous with a Lipschitz constant independent of the size. This is a consequence of neural networks typically having relatively small weights. To motivate this, let us consider the class of neural networks $\mathcal{N}(\sigma; \mathcal{A}, B)$ for an architecture $\mathcal{A}$ of depth $d \in \mathbb{N}$ and width $L \in \mathbb{N}$. If σ is C_σ-Lipschitz continuous such that $B \leq c_B \cdot (dC_\sigma)^{-1}$ for some $c_B > 0$, then by Lemma 13.2

$$\mathcal{N}(\sigma; \mathcal{A}, B) \subseteq \mathrm{Lip}_{c_B^L}(\mathbb{R}^{d_0}), \tag{15.3.1}$$

An assumption of the type $B \leq c_B \cdot (dC_\sigma)^{-1}$, i.e. a scaling of the weights by the reciprocal $1/d$ of the width, is not unreasonable in practice: Standard initialization schemes such as LeCun [129] or He [86] initialization, use random weights with variance scaled inverse proportional to the input dimension of each layer. Moreover, as we saw in Chapter 11, for very wide neural networks, the weights do not move significantly from their initialization during training. Additionally, many training routines use regularization terms on the weights, thereby encouraging them the optimization routine to find small weights.

We study the generalization capacity of Lipschitz functions through the covering-number-based learning results of Chapter 14. The set of C-Lipschitz functions on a compact d-dimensional Euclidean domain $\mathrm{Lip}_C(\Omega)$ has covering numbers bounded according to

$$\log(\mathcal{G}(\mathrm{Lip}_C(\Omega), \varepsilon, L^\infty)) \leq C_{\mathrm{cov}} \cdot \left(\frac{C}{\varepsilon}\right)^d \quad \text{for all } \varepsilon > 0 \tag{15.3.2}$$

for some constant C_{cov} independent of $\varepsilon > 0$. A proof can be found in [75, Lemma 7], see also [230].

As a result of these considerations, we can identify two regimes:

- *Standard regime:* For small neural network size $n_{\mathcal{A}}$, we consider neural networks as a set parameterized by $n_{\mathcal{A}}$ parameters. As we have seen before, this yields a bound on the generalization error that scales linearly with $n_{\mathcal{A}}$. As long as $n_{\mathcal{A}}$ is small in comparison to the number of samples, we can expect good generalization by Theorem 14.15.
- *Overparameterized regime:* For large neural network size $n_{\mathcal{A}}$ and small weights, we consider neural networks as a subset of $\mathrm{Lip}_C(\Omega)$ for a constant $C > 0$. This set has a covering number bound that is independent of the number of parameters $n_{\mathcal{A}}$.

Choosing the better of the two generalization bounds for each regime yields the following result. Recall that $\mathcal{N}^*(\sigma; \mathcal{A}, B)$ denotes all neural networks in $\mathcal{N}(\sigma; \mathcal{A}, B)$ with a range contained in $[-1, 1]$ (see (14.5.1)).

Theorem 15.2. *Let C, $C_{\mathcal{L}} > 0$ and let $\mathcal{L}\colon [-1,1] \times [-1,1] \to \mathbb{R}$ be $C_{\mathcal{L}}$-Lipschitz. Further, let $\mathcal{A} = (d_0, d_1, \dots, d_{L+1}) \in \mathbb{N}^{L+2}$, let $\sigma\colon \mathbb{R} \to \mathbb{R}$ be C_σ-Lipschitz continuous with $C_\sigma \geq 1$, and $|\sigma(x)| \leq C_\sigma |x|$ for all $x \in \mathbb{R}$, and let $B > 0$.*

Then, there exist c_1, $c_2 > 0$, such that for every $m \in \mathbb{N}$ and every distribution $\mathcal{D}$ on $[-1,1]^{d_0} \times [-1,1]$ it holds with probability at least $1 - \delta$ over $S \sim \mathcal{D}^m$ that for all $\Phi \in \mathcal{N}^*(\sigma; \mathcal{A}, B) \cap \mathrm{Lip}_C([-1,1]^{d_0})$

$$|\mathcal{R}(\Phi) - \widehat{\mathcal{R}}_S(\Phi)| \leq g(\mathcal{A}, C_\sigma, B, m) + 4C_{\mathcal{L}} \sqrt{\frac{\log(4/\delta)}{m}}, \tag{15.3.3}$$

where

$$g(\mathcal{A}, C_\sigma, B, m) = \min\left\{ c_1 \sqrt{\frac{n_{\mathcal{A}} \log(n_{\mathcal{A}} \lceil \sqrt{m} \rceil) + L n_{\mathcal{A}} \log(d_{\max})}{m}}, c_2 m^{-\frac{1}{2+d_0}} \right\}.$$

Proof. Applying Theorem 14.11 with $\alpha = 1/(2 + d_0)$ and (15.3.2), we obtain that with probability at least $1 - \delta/2$ it holds for all $\Phi \in \mathrm{Lip}_C([-1,1]^{d_0})$

$$\begin{aligned}
|\mathcal{R}(\Phi) - \widehat{\mathcal{R}}_S(\Phi)| &\leq 4C_{\mathcal{L}} \sqrt{\frac{C_{\mathrm{cov}}(m^\alpha C)^{d_0} + \log(4/\delta)}{m}} + \frac{2C_{\mathcal{L}}}{m^\alpha} \\
&\leq 4C_{\mathcal{L}} \sqrt{C_{\mathrm{cov}} C^{d_0} (m^{d_0/(d_0+2)-1})} + \frac{2C_{\mathcal{L}}}{m^\alpha} + 4C_{\mathcal{L}} \sqrt{\frac{\log(4/\delta)}{m}} \\
&= 4C_{\mathcal{L}} \sqrt{C_{\mathrm{cov}} C^{d_0} (m^{-2/(d_0+2)})} + \frac{2C_{\mathcal{L}}}{m^\alpha} + 4C_{\mathcal{L}} \sqrt{\frac{\log(4/\delta)}{m}} \\
&= \frac{(4C_{\mathcal{L}} \sqrt{C_{\mathrm{cov}} C^{d_0}} + 2C_{\mathcal{L}})}{m^\alpha} + 4C_{\mathcal{L}} \sqrt{\frac{\log(4/\delta)}{m}},
\end{aligned}$$

where we used in the second inequality that $\sqrt{x+y} \leq \sqrt{x} + \sqrt{y}$ for all x, $y \geq 0$.

In addition, Theorem 14.15 yields that with probability at least $1-\delta/2$ it holds for all $\Phi \in \mathcal{N}^*(\sigma;\mathcal{A},B)$

$$\begin{aligned}|\mathcal{R}(\Phi)-\widehat{\mathcal{R}}_S(\Phi)| &\leq 4C_{\mathcal{L}}\sqrt{\frac{n_{\mathcal{A}}\log(\lceil n_{\mathcal{A}}\sqrt{m}\rceil)+Ln_{\mathcal{A}}\log(\lceil 2C_\sigma B d_{\max}\rceil)+\log(4/\delta)}{m}} \\ &\quad + \frac{2C_{\mathcal{L}}}{\sqrt{m}} \\ &\leq 6C_{\mathcal{L}}\sqrt{\frac{n_{\mathcal{A}}\log(\lceil n_{\mathcal{A}}\sqrt{m}\rceil)+Ln_{\mathcal{A}}\log(\lceil 2C_\sigma B d_{\max}\rceil)}{m}} \\ &\quad + 4C_{\mathcal{L}}\sqrt{\frac{\log(4/\delta))}{m}}.\end{aligned}$$

Then, for $\Phi \in \mathcal{N}^*(\sigma;\mathcal{A},B)\cap \mathrm{Lip}_C([-1,1]^{d_0})$ the minimum of both upper bounds holds with probability at least $1-\delta$. □

The two regimes in Theorem 15.2 correspond to the two terms comprising the minimum in the definition of $g(\mathcal{A},C_\sigma,B,m)$. The first term increases with $n_{\mathcal{A}}$ while the second is constant. In the first regime, where the first term is smaller, the generalization gap $|\mathcal{R}(\Phi)-\widehat{\mathcal{R}}_S(\Phi)|$ increases with $n_{\mathcal{A}}$.

In the second regime, where the second term is smaller, the generalization gap is constant with $n_{\mathcal{A}}$. Moreover, it is reasonable to assume that the empirical risk $\widehat{\mathcal{R}}_S$ will decrease with increasing number of parameters $n_{\mathcal{A}}$.

By (15.3.3) we can bound the risk by

$$\mathcal{R}(\Phi) \leq \widehat{\mathcal{R}}_S + g(\mathcal{A},C_\sigma,B,m) + 4C_{\mathcal{L}}\sqrt{\frac{\log(4/\delta)}{m}}.$$

In the second regime, this upper bound is monotonically decreasing. In the first regime it may both decrease and increase. In some cases, this behavior can lead to an upper bound on the risk resembling the curve of Figure 15.2. The following section describes a specific scenario where this is the case.

Remark 15.3. Theorem 15.2 assumes C-Lipschitz continuity of the neural networks. As we saw in Sections 15.1.2 and 15.2, this assumption may not hold near the interpolation threshold. Hence, Theorem 15.2 likely gives a too optimistic upper bound near the interpolation threshold.

15.4 Double descent for neural network learning

Now let us understand the double descent phenomenon in the context of Theorem 15.2. We make a couple of simplifying assumptions to obtain a formula for an upper bound on the risk. First, we assume that the data $S=(\boldsymbol{x}_i,y_i)_{i=1}^m \in \mathbb{R}^{d_0}\times\mathbb{R}$ stem from a C_M-Lipschitz continuous function. In addition, we fix a depth $L\in\mathbb{N}$ and consider, for $d\in\mathbb{N}$, architectures of the form $(\sigma_{\mathrm{ReLU}};\mathcal{A}_d)$, where

$$\mathcal{A}_d = (d_0,d,\ldots,d,1).$$

For this architecture the number of parameters is bounded by

$$n_{\mathcal{A}_d} = (d_0+1)d + (L-1)(d+1)d + d + 1.$$

To derive an upper bound on the risk, we start by upper bounding the empirical risk and then applying Theorem 15.2 to establish an upper bound on the generalization gap. In combination, these estimates provide an upper bound on the risk. We will then observe that this upper bound follows the double descent curve in Figure 15.2.

15.4.1 Upper bound on empirical risk

We establish an upper bound on $\widehat{\mathcal{R}}_S(\Phi)$ for $\Phi \in \mathcal{N}^*(\sigma_{\mathrm{ReLU}}; \mathcal{A}_d, B) \cap \mathrm{Lip}_{C_M}([-1,1]^{d_0})$. For $B \geq C_M$, we can apply Theorem 9.6, and conclude that with a neural network of sufficient depth we can interpolate m points from a C_M-Lipschitz function with a neural network in $\mathrm{Lip}_{C_M}([-1,1]^{d_0})$, if $n_{\mathcal{A}} \geq c_{\mathrm{int}} \log(m) d_0 m$. To simplify the exposition, we assume $c_{\mathrm{int}} = 1$ in the following. Thus, $\widehat{\mathcal{R}}_S(\Phi) = 0$ as soon as $n_{\mathcal{A}} \geq \log(m) d_0 m$.

In addition, depending on smoothness properties of the data, the interpolation error may decay with some rate, by one of the results in Chapters 5, 7, or 8. For simplicity, we choose that $\widehat{\mathcal{R}}_S(\Phi) = O(n_{\mathcal{A}}^{-1})$ for $n_{\mathcal{A}}$ significantly smaller than $\log(m) d_0 m$. If we combine these two assumptions, we can make the following Ansatz for the empirical risk of $\Phi_{\mathcal{A}_d} \in \mathcal{N}^*(\sigma_{\mathrm{ReLU}}; \mathcal{A}_d, B) \cap \mathrm{Lip}_{C_M}([-1,1]^{d_0})$:

$$\widehat{\mathcal{R}}_S(\Phi_{\mathcal{A}_d}) \leq \widetilde{\mathcal{R}}_S(\Phi_{\mathcal{A}_d}) \coloneqq C_{\mathrm{approx}} \max\left\{0, n_{\mathcal{A}_d}^{-1} - (\log(m) d_0 m)^{-1}\right\} \tag{15.4.1}$$

for a constant $C_{\mathrm{approx}} > 0$. Note that, we can interpolate the sample S already with $d_0 m$ parameters by Theorem 9.3. However, it is not guaranteed that this can be done using C_M-Lipschitz neural networks.

15.4.2 Upper bound on generalization gap

We complement the bound on the empirical risk by an upper bound on the risk. Invoking the notation of Theorem 15.2, we have that,

$$g(\mathcal{A}_d, C_{\sigma_{\mathrm{ReLU}}}, B, m) = \min\left\{\kappa_{\mathrm{NN}}(\mathcal{A}_d, m; c_1), \kappa_{\mathrm{Lip}}(\mathcal{A}_d, m; c_2)\right\},$$

where

$$\begin{aligned} \kappa_{\mathrm{NN}}(\mathcal{A}_d, m; c_1) &\coloneqq c_1 \sqrt{\frac{n_{\mathcal{A}_d} \log(\lceil n_{\mathcal{A}} \sqrt{m} \rceil) + L n_{\mathcal{A}_d} \log(d)}{m}}, \\ \kappa_{\mathrm{Lip}}(\mathcal{A}_d, m; c_2) &\coloneqq c_2 m^{-\frac{1}{2+d_0}} \end{aligned} \tag{15.4.2}$$

for some constants c_1, $c_2 > 0$.

15.4.3 Upper bound on risk

Next, we combine (15.4.1) and (15.4.2) to obtain an upper bound on the risk $\mathcal{R}(\Phi_{\mathcal{A}_d})$. Specifically, we define

$$\begin{aligned} \widetilde{\mathcal{R}}(\Phi_{\mathcal{A}_d}) \coloneqq& \widetilde{\mathcal{R}}_S(\Phi_{\mathcal{A}_d}) + \min\left\{\kappa_{\mathrm{NN}}(\mathcal{A}_d, m; c_1), \kappa_{\mathrm{Lip}}(\mathcal{A}_d, m; c_2)\right\} \\ &+ 4 C_{\mathcal{L}} \sqrt{\frac{\log(4/\delta)}{m}}. \end{aligned} \tag{15.4.3}$$

We depict in Figure 15.6 the upper bound on the risk given by (15.4.3) (excluding the terms that do not depend on the architecture). The upper bound clearly resembles the double descent phenomenon of Figure 15.2. Note that the Lipschitz interpolation point is slightly behind this threshold, which is when we assume our empirical risk to be 0. To produce the plot, we chose $L = 5$, $c_1 = 1.2 \cdot 10^{-4}$, $c_2 = 6.5 \cdot 10^{-3}$, $m = 10.000$, $d_0 = 6$, $C_{\text{approx}} = 30$. We mention that the double descent phenomenon is not visible for all choices of parameters. Moreover, in our model, the fact that the peak coincides with the interpolation threshold is due to the choice of constants and does not emerge from the model. Other models of double descent explain the location of the peak more accurately [143, 83]. We note that, as observed in Remark 15.3, the peak close to the interpolation threshold that we see in Figure 15.6 would likely be more pronounced in practical scenarios.

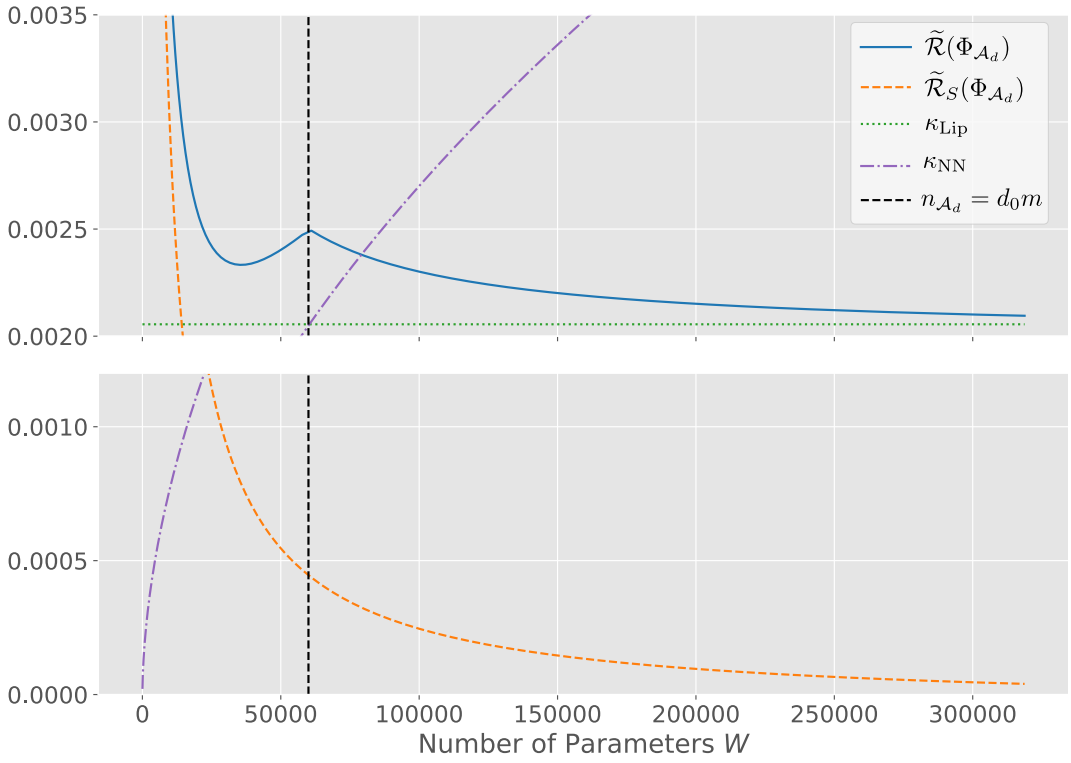


Figure 15.6: Upper bound on $\mathcal{R}(\Phi_{\mathcal{A}_d})$ derived in (15.4.3). For better visibility the part corresponding to y-values between 0.0012 and 0.0022 is not shown. The vertical dashed line indicates the interpolation threshold according to Theorem 9.3.

Bibliography and further reading

The discussion on kernel regression and the effect of the number of parameters on the norm of the weights was already given in [14]. Similar analyses, with more complex ansatz systems and more precise asymptotic estimates, are found in [143, 83]. Our results in Section 15.3 are inspired by [12]; see also [161].

For a detailed account of further arguments justifying the surprisingly good generalization

capabilities of overparameterized neural networks, we refer to [19, Section 2]. Here, we only briefly mention two additional directions of inquiry. First, if the learning algorithm introduces a form of robustness, this can be leveraged to yield generalization bounds [6, 244, 24, 179]. Second, for very overparameterized neural networks, it was stipulated in [106] that neural networks become linear kernel interpolators based on the neural tangent kernel of Section 11.5.2. Thus, for large neural networks, generalization can be studied through kernel regression [106, 131, 15, 135].

Exercises

Exercise 15.4. Let $f : [-1, 1] \to \mathbb{R}$ be a continuous function, and let $-1 \leq x_1 < \cdots < x_m \leq 1$ for some fixed $m \in \mathbb{N}$. As in Section 15.1.2, we wish to approximate f by a least squares approximation. To this end we use the Fourier ansatz functions

$$b_0(x) := \frac{1}{2} \qquad \text{and} \qquad b_j(x) := \begin{cases} \sin(\lceil \frac{j}{2} \rceil \pi x) & j \geq 1 \text{ is odd} \\ \cos(\lceil \frac{j}{2} \rceil \pi x) & j \geq 1 \text{ is even.} \end{cases} \tag{15.4.4}$$

With the empirical risk

$$\widehat{\mathcal{R}}_S(\boldsymbol{w}) = \frac{1}{m} \sum_{j=1}^{m} \Big(\sum_{i=0}^{n} w_i b_i(x_j) - y_j \Big)^2,$$

denote by $\boldsymbol{w}_*^n \in \mathbb{R}^{n+1}$ the minimal norm minimizer of $\widehat{\mathcal{R}}_S$, and set $f_n(x) := \sum_{i=0}^{n} w_{*,i}^n b_i(x)$.

Show that in this case generalization fails in the overparametrized regime: for sufficiently large $n \gg m$, f_n is *not* necessarily a good approximation to f. What does f_n converge to as $n \to \infty$?

Exercise 15.5. Consider the setting of Exercise 15.4. We adapt the ansatz functions in (15.4.4) by rescaling them via

$$\tilde{b}_j := c_j b_j.$$

Choose real numbers $c_j \in \mathbb{R}$, such that the corresponding minimal norm least squares solution avoids the phenomenon encountered in Exercise 15.4.

Hint: Should ansatz functions corresponding to large frequencies be scaled by large or small numbers to avoid overfitting?

Exercise 15.6. Prove (15.3.2) for $d = 1$.

Chapter 16

Robustness and adversarial examples

How sensitive is the output of a neural network to small changes in its input? Real-world observations of trained neural networks often reveal that even barely noticeable modifications of the input can lead to drastic variations in the network's predictions. This intriguing behavior was first documented in the context of image classification in [223].

Figure 16.1 illustrates this concept. The left panel shows a picture of a panda that the neural network correctly classifies as a panda. By adding an almost imperceptible amount of noise to the image, we obtain the modified image in the right panel. To a human, there is no visible difference, but the neural network classifies the perturbed image as a wombat. This phenomenon, where a correctly classified image is misclassified after a slight perturbation, is termed an *adversarial example*.

In practice, such behavior is highly undesirable. It indicates that our learning algorithm might not be very reliable and poses a potential security risk, as malicious actors could exploit it to trick the algorithm. In this chapter, we describe the basic mathematical principles behind adversarial examples and investigate simple conditions under which they might or might not occur. For simplicity, we restrict ourselves to a binary classification problem but note that the main ideas remain valid in more general situations.

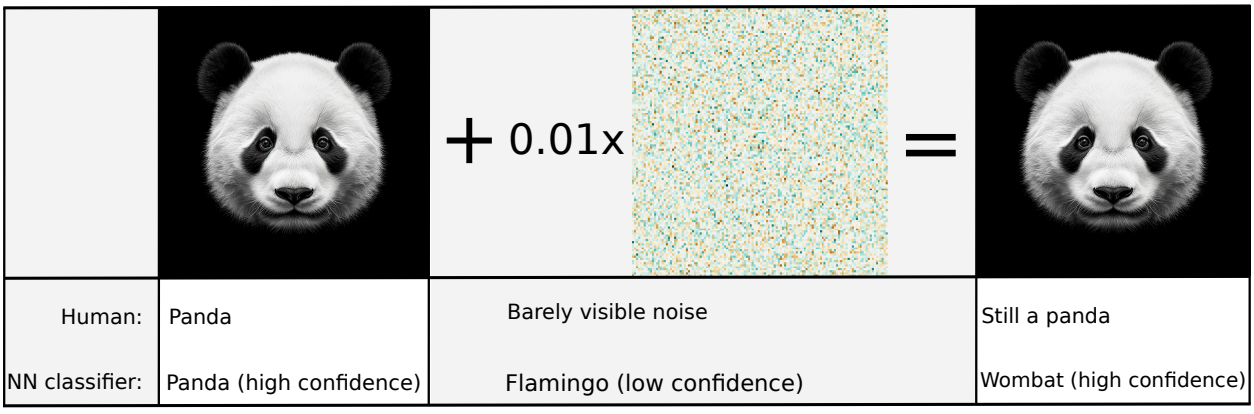


Figure 16.1: Sketch of an adversarial example.

16.1 Adversarial examples

Let us start by formalizing the notion of an adversarial example. We consider the problem of assigning a label $y \in \{-1, 1\}$ to a vector $\boldsymbol{x} \in \mathbb{R}^d$. It is assumed that the relation between $\boldsymbol{x}$ and y is described by a distribution $\mathcal{D}$ on $\mathbb{R}^d \times \{-1, 1\}$. In particular, for a given $\boldsymbol{x}$, both values -1 and 1 could have positive probability, i.e. the label is not necessarily deterministic. Additionally, we let

$$D_{\boldsymbol{x}} := \{\boldsymbol{x} \in \mathbb{R}^d \,|\, \exists y \text{ s.t. } (\boldsymbol{x}, y) \in \operatorname{supp}(\mathcal{D})\}, \tag{16.1.1}$$

and refer to $D_{\boldsymbol{x}}$ as the **feature support**.

Throughout this chapter we denote by

$$g\colon \mathbb{R}^d \to \{-1, 0, 1\}$$

a fixed so-called *ground-truth classifier*, satisfying[1]

$$\mathbb{P}[y = g(\boldsymbol{x})|\boldsymbol{x}] \geq \mathbb{P}[y = -g(\boldsymbol{x})|\boldsymbol{x}] \qquad \text{for all } \boldsymbol{x} \in D_{\boldsymbol{x}}. \tag{16.1.2}$$

Note that we allow g to take the value 0, which is to be understood as an additional label corresponding to nonrelevant or nonsensical input data $\boldsymbol{x}$. We will refer to $g^{-1}(0)$ as the **nonrelevant class**. The ground truth g is interpreted as how a human would classify the data, as the following example illustrates.

Example 16.1. We wish to classify whether an image shows a panda ($y = 1$) or a wombat ($y = -1$). Consider again Figure 16.1, and denote the three images by $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$. The first image $\boldsymbol{x}_1$ is a photograph of a panda. Together with a label y, it can be interpreted as a draw $(\boldsymbol{x}_1, y)$ from $\mathcal{D}$, i.e. $\boldsymbol{x}_1 \in D_{\boldsymbol{x}}$ and $g(\boldsymbol{x}_1) = 1$. The second image $\boldsymbol{x}_2$ displays noise and corresponds to nonrelevant data as it shows neither a panda nor a wombat. In particular, $\boldsymbol{x}_2 \in D_{\boldsymbol{x}}^c$ and $g(\boldsymbol{x}_2) = 0$. The third (perturbed) image $\boldsymbol{x}_3$ also belongs to $D_{\boldsymbol{x}}^c$, as it is not a photograph but a noise corrupted version of $\boldsymbol{x}_1$. Nonetheless, it is *not* nonrelevant, as a human would classify it as a panda. Thus $g(\boldsymbol{x}_3) = 1$.

Additional to the ground truth g, we denote by

$$h\colon \mathbb{R}^d \to \{-1, 1\}$$

some trained classifier.

Definition 16.2. Let $g\colon \mathbb{R}^d \to \{-1, 0, 1\}$ be the ground-truth classifier, let $h\colon \mathbb{R}^d \to \{-1, 1\}$ be a classifier, and let $\|\cdot\|_*$ be a norm on $\mathbb{R}^d$. For $\boldsymbol{x} \in \mathbb{R}^d$ and $\delta > 0$, we call $\boldsymbol{x}' \in \mathbb{R}^d$ an **adversarial example** to $\boldsymbol{x} \in \mathbb{R}^d$ with perturbation δ, if and only if

(i) $\|\boldsymbol{x}' - \boldsymbol{x}\|_* \leq \delta$,

(ii) $g(\boldsymbol{x})g(\boldsymbol{x}') > 0$,

(iii) $h(\boldsymbol{x}) = g(\boldsymbol{x})$ and $h(\boldsymbol{x}') \neq g(\boldsymbol{x}')$.

[1] To be more precise, the conditional distribution of $y|\boldsymbol{x}$ is only well-defined almost everywhere w.r.t. the marginal distribution of $\boldsymbol{x}$. Thus (16.1.2) can only be assumed to hold for *almost every* $\boldsymbol{x} \in D_{\boldsymbol{x}}$ *w.r.t. to the marginal distribution of* $\boldsymbol{x}$.

In words, $\boldsymbol{x}'$ is an adversarial example to $\boldsymbol{x}$ with perturbation δ, if (i) the distance of $\boldsymbol{x}$ and $\boldsymbol{x}'$ is at most δ, (ii) $\boldsymbol{x}$ and $\boldsymbol{x}'$ belong to the same (not nonrelevant) class according to the ground truth classifier, and (iii) the classifier h correctly classifies $\boldsymbol{x}$ but misclassifies $\boldsymbol{x}'$.

Remark 16.3. We emphasize that the concept of a ground-truth classifier g differs from a minimizer of the Bayes risk (14.1.1) for two reasons. First, we allow for an additional label 0 corresponding to the nonrelevant class, which does not exist for the data generating distribution $\mathcal{D}$. Second, g should correctly classify points *outside of* $D_{\boldsymbol{x}}$; small perturbations of images as we find them in adversarial examples, are not regular images in $D_{\boldsymbol{x}}$. Nonetheless, a human classifier can still classify these images, and g models this property of human classification.

16.2 Bayes classifier

At first sight, an adversarial example seems to be no more than a misclassified sample. Naturally, these exist if the model does not generalize well. In this section we present a more nuanced view from [218].

To avoid edge cases, we assume in the following that for all $\boldsymbol{x} \in D_{\boldsymbol{x}}$

$$\text{either} \quad \mathbb{P}[y=1|\boldsymbol{x}] > \mathbb{P}[y=-1|\boldsymbol{x}] \quad \text{or} \quad \mathbb{P}[y=1|\boldsymbol{x}] < \mathbb{P}[y=-1|\boldsymbol{x}] \tag{16.2.1}$$

so that (16.1.2) uniquely defines $g(\boldsymbol{x})$ for $\boldsymbol{x} \in D_{\boldsymbol{x}}$. We say that the distribution **exhausts the domain** if $D_{\boldsymbol{x}} \cup g^{-1}(0) = \mathbb{R}^d$. This means that every point is either in the feature support $D_{\boldsymbol{x}}$ or it belongs to the nonrelevant class. Moreover, we say that h is a **Bayes classifier** if

$$\mathbb{P}[h(\boldsymbol{x})|\boldsymbol{x}] \geq \mathbb{P}[-h(\boldsymbol{x})|\boldsymbol{x}] \qquad \text{for all } \boldsymbol{x} \in D_{\boldsymbol{x}}.$$

By (16.1.2), the ground truth g is a Bayes classifier, and (16.2.1) ensures that h coincides with g on $D_{\boldsymbol{x}}$ if h is a Bayes classifier. It is easy to see that a Bayes classifier minimizes the Bayes risk.

With these two notions, we now distinguish between four cases.

(i) *Bayes classifier/exhaustive distribution:* If h is a Bayes classifier and the data exhausts the domain, then there are *no adversarial examples*. This is because every $\boldsymbol{x} \in \mathbb{R}^d$ either belongs to the nonrelevant class or is classified the same by h and g.

(ii) *Bayes classifier/non-exhaustive distribution:* If h is a Bayes classifier and the distribution does not exhaust the domain, then *adversarial examples can exist*. Even though the learned classifier h coincides with the ground truth g on the feature support, adversarial examples can be constructed for data points on the complement of $D_{\boldsymbol{x}} \cup g^{-1}(0)$, which is not empty.

(iii) *Not a Bayes classifier/exhaustive distribution:* The set $D_{\boldsymbol{x}}$ can be covered by the four subdomains

$$\begin{aligned} C_1 &= h^{-1}(1) \cap g^{-1}(1), \quad F_1 = h^{-1}(-1) \cap g^{-1}(1), \\ C_{-1} &= h^{-1}(-1) \cap g^{-1}(-1), \quad F_{-1} = h^{-1}(1) \cap g^{-1}(-1). \end{aligned} \tag{16.2.2}$$

If $\operatorname{dist}(C_1 \cap D_{\boldsymbol{x}}, F_1 \cap D_{\boldsymbol{x}})$ or $\operatorname{dist}(C_{-1} \cap D_{\boldsymbol{x}}, F_{-1} \cap D_{\boldsymbol{x}})$ is smaller than δ, then there exist points $\boldsymbol{x}$, $\boldsymbol{x}' \in D_{\boldsymbol{x}}$ such that $\boldsymbol{x}'$ is an adversarial example to x with perturbation δ. Hence, *adversarial examples in the feature support can exist*. This is, however, not guaranteed to happen. For example, $D_{\boldsymbol{x}}$ does not need to be connected if $g^{-1}(0) \neq \emptyset$, see Exercise 16.18. Hence, even for classifiers that have incorrect predictions on the data, adversarial examples *do not need to exist*.

(iv) *Not a Bayes classifier/non-exhaustive distribution:* In this case *everything is possible*. Data points and their associated adversarial examples can appear in the feature support of the distribution and adversarial examples to elements in the feature support of the distribution can be created by leaving the feature support of the distribution. We will see examples in the following section.

16.3 Affine classifiers

For linear classifiers, a simple argument outlined in [223] and [73] showcases that the high-dimensionality of the input, common in image classification problems, is a potential cause for the existence of adversarial examples.

A linear classifier is a map of the form

$$\boldsymbol{x} \mapsto \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x}) \qquad \text{where } \boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^d.$$

Let

$$\boldsymbol{x}' := \boldsymbol{x} - 2|\boldsymbol{w}^\top \boldsymbol{x}| \frac{\operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x}) \operatorname{sign}(\boldsymbol{w})}{\|\boldsymbol{w}\|_1}$$

where $\operatorname{sign}(\boldsymbol{w})$ is understood coordinate-wise. Then $\|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \leq 2|\boldsymbol{w}^\top \boldsymbol{x}|/\|\boldsymbol{w}\|_1$ and it is not hard to see that $\operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x}') \neq \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x})$.

For high-dimensional vectors $\boldsymbol{w}$, $\boldsymbol{x}$ chosen at random but possibly dependent such that $\boldsymbol{w}$ is uniformly distributed on a $d-1$ dimensional sphere, it holds with high probability that

$$\frac{|\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|_1} \leq \frac{\|\boldsymbol{x}\| \|\boldsymbol{w}\|}{\|\boldsymbol{w}\|_1} \ll \|\boldsymbol{x}\|.$$

This can be seen by noting that for every $c > 0$

$$\mu(\{\boldsymbol{w} \in \mathbb{R}^d \mid \|\boldsymbol{w}\|_1 > c, \|\boldsymbol{w}\| \leq 1\}) \to 1 \text{ for } d \to \infty, \tag{16.3.1}$$

where μ is the uniform probability measure on the d-dimensional Euclidean unit ball, see Exercise 16.17. Thus, if $\boldsymbol{x}$ has a moderate Euclidean norm, the perturbation of $\boldsymbol{x}'$ is likely small for large dimensions.

Below we give a sufficient condition for the existence of adversarial examples, in case both h and the ground truth g are linear classifiers.

Theorem 16.4. *Let $\boldsymbol{w}$, $\overline{\boldsymbol{w}} \in \mathbb{R}^d$ be nonzero. For $\boldsymbol{x} \in \mathbb{R}^d$, let $h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x})$ be a classifier and let $g(\boldsymbol{x}) = \operatorname{sign}(\overline{\boldsymbol{w}}^\top x)$ be the ground-truth classifier.*

For every $\boldsymbol{x} \in \mathbb{R}^d$ with $h(\boldsymbol{x})g(\boldsymbol{x}) > 0$ and all $\varepsilon \in (0, |\boldsymbol{w}^\top \boldsymbol{x}|)$ such that

$$\frac{|\overline{\boldsymbol{w}}^\top \boldsymbol{x}|}{\|\overline{\boldsymbol{w}}\|} > \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|} \frac{|\boldsymbol{w}^\top \overline{\boldsymbol{w}}|}{\|\boldsymbol{w}\| \|\overline{\boldsymbol{w}}\|} \tag{16.3.2}$$

it holds that

$$\boldsymbol{x}' = \boldsymbol{x} - h(\boldsymbol{x}) \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|^2} \boldsymbol{w} \tag{16.3.3}$$

is an adversarial example to $\boldsymbol{x}$ with perturbation $\delta = (\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|)/\|\boldsymbol{w}\|$.

Before we present the proof, we give some interpretation of this result. First, note that $\{\boldsymbol{x} \in \mathbb{R}^d \,|\, \boldsymbol{w}^\top \boldsymbol{x} = 0\}$ is the decision boundary of h, meaning that points lying on opposite sides of this hyperplane, are classified differently by h. Due to $|\boldsymbol{w}^\top \overline{\boldsymbol{w}}| \leq \|\boldsymbol{w}\|\|\overline{\boldsymbol{w}}\|$, (16.3.2) implies that an adversarial example always exists whenever

$$\frac{|\overline{\boldsymbol{w}}^\top \boldsymbol{x}|}{\|\overline{\boldsymbol{w}}\|} > \frac{|\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|}. \tag{16.3.4}$$

The left term is the decision margin of $\boldsymbol{x}$ for g, i.e. the distance of $\boldsymbol{x}$ to the decision boundary of g. Similarly, the term on the right is the decision margin of $\boldsymbol{x}$ for h. Thus we conclude that adversarial examples exist if the decision margin of $\boldsymbol{x}$ for the ground truth g is larger than that for the classifier h.

Second, the term $(\boldsymbol{w}^\top \overline{\boldsymbol{w}})/(\|\boldsymbol{w}\|\|\overline{\boldsymbol{w}}\|)$ describes the alignment of the two classifiers. If the classifiers are not aligned, i.e., $\boldsymbol{w}$ and $\overline{\boldsymbol{w}}$ have a large angle between them, then adversarial examples exist even if the margin of the classifier is larger than that of the ground-truth classifier.

Finally, adversarial examples with small perturbation are possible if $|\boldsymbol{w}^\top \boldsymbol{x}| \ll \|\boldsymbol{w}\|$. The extreme case $\boldsymbol{w}^\top \boldsymbol{x} = 0$ means that $\boldsymbol{x}$ lies on the decision boundary of h, and if $|\boldsymbol{w}^\top \boldsymbol{x}| \ll \|\boldsymbol{w}\|$ then $\boldsymbol{x}$ is close to the decision boundary of h.

of Theorem 16.4. We verify that $\boldsymbol{x}'$ in (16.3.3) satisfies the conditions of an adversarial example in Definition 16.2. In the following we will use that due to $h(\boldsymbol{x})g(\boldsymbol{x}) > 0$

$$g(\boldsymbol{x}) = \operatorname{sign}(\overline{\boldsymbol{w}}^\top \boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x}) = h(\boldsymbol{x}) \neq 0. \tag{16.3.5}$$

First, it holds

$$\|\boldsymbol{x} - \boldsymbol{x}'\| = \left\| \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|^2} \boldsymbol{w} \right\| = \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|} = \delta.$$

Next we show $g(\boldsymbol{x})g(\boldsymbol{x}') > 0$, i.e. that $(\overline{\boldsymbol{w}}^\top \boldsymbol{x})(\overline{\boldsymbol{w}}^\top \boldsymbol{x}')$ is positive. Plugging in the definition of $\boldsymbol{x}'$, this term reads

$$\begin{aligned}
\overline{\boldsymbol{w}}^\top \boldsymbol{x} \left(\overline{\boldsymbol{w}}^\top \boldsymbol{x} - h(\boldsymbol{x}) \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|^2} \overline{\boldsymbol{w}}^\top \boldsymbol{w} \right) &= |\overline{\boldsymbol{w}}^\top \boldsymbol{x}|^2 - |\overline{\boldsymbol{w}}^\top \boldsymbol{x}| \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|^2} \overline{\boldsymbol{w}}^\top \boldsymbol{w} \\
&\geq |\overline{\boldsymbol{w}}^\top \boldsymbol{x}|^2 - |\overline{\boldsymbol{w}}^\top \boldsymbol{x}| \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|^2} |\overline{\boldsymbol{w}}^\top \boldsymbol{w}|,
\end{aligned} \tag{16.3.6}$$

where the equality holds because $h(\boldsymbol{x}) = g(\boldsymbol{x}) = \operatorname{sign}(\overline{\boldsymbol{w}}^\top \boldsymbol{x})$ by (16.3.5). Dividing the right-hand side of (16.3.6) by $|\overline{\boldsymbol{w}}^\top \boldsymbol{x}|\|\overline{\boldsymbol{w}}\|$, which is positive by (16.3.5), we obtain

$$\frac{|\overline{\boldsymbol{w}}^\top \boldsymbol{x}|}{\|\overline{\boldsymbol{w}}\|} - \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|} \frac{|\overline{\boldsymbol{w}}^\top \boldsymbol{w}|}{\|\boldsymbol{w}\|\|\overline{\boldsymbol{w}}\|}. \tag{16.3.7}$$

The term (16.3.7) is positive thanks to (16.3.2).

Finally, we check that $0 \neq h(\boldsymbol{x}') \neq h(\boldsymbol{x})$, i.e. $(\boldsymbol{w}^\top \boldsymbol{x})(\boldsymbol{w}^\top \boldsymbol{x}') < 0$. We have that

$$\begin{aligned}
(\boldsymbol{w}^\top \boldsymbol{x})(\boldsymbol{w}^\top \boldsymbol{x}') &= |\boldsymbol{w}^\top \boldsymbol{x}|^2 - \boldsymbol{w}^\top \boldsymbol{x} h(\boldsymbol{x}) \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|^2} \boldsymbol{w}^\top \boldsymbol{w} \\
&= |\boldsymbol{w}^\top \boldsymbol{x}|^2 - |\boldsymbol{w}^\top \boldsymbol{x}|(\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|) < 0,
\end{aligned}$$

where we used that $h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x})$. This completes the proof. □

Theorem 16.4 readily implies the following proposition for *affine* classifiers.

Proposition 16.5. *Let $\boldsymbol{w}$, $\overline{\boldsymbol{w}} \in \mathbb{R}^d$ and b, $\bar{b} \in \mathbb{R}$. For $\boldsymbol{x} \in \mathbb{R}^d$ let $h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$ be a classifier and let $g(\boldsymbol{x}) = \operatorname{sign}(\overline{\boldsymbol{w}}^\top \boldsymbol{x} + \bar{b})$ be the ground-truth classifier.*

For every $\boldsymbol{x} \in \mathbb{R}^d$ with $\overline{\boldsymbol{w}}^\top \boldsymbol{x} \neq 0$, $h(\boldsymbol{x})g(\boldsymbol{x}) > 0$, and all $\varepsilon \in (0, |\boldsymbol{w}^\top \boldsymbol{x} + b|)$ such that

$$\frac{|\overline{\boldsymbol{w}}^\top \boldsymbol{x} + \bar{b}|^2}{\|\overline{\boldsymbol{w}}\|^2 + b^2} > \frac{(\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x} + b|)^2}{\|\boldsymbol{w}\|^2 + b^2} \frac{(\boldsymbol{w}^\top \overline{\boldsymbol{w}} + b\bar{b})^2}{(\|\boldsymbol{w}\|^2 + b^2)(\|\overline{\boldsymbol{w}}\|^2 + \bar{b}^2)}$$

it holds that

$$\boldsymbol{x}' = \boldsymbol{x} - h(\boldsymbol{x}) \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x} + b|}{\|\boldsymbol{w}\|^2} \boldsymbol{w}$$

is an adversarial example with perturbation $\delta = (\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x} + b|)/\|\boldsymbol{w}\|$ to $\boldsymbol{x}$.

The proof is left to the reader, see Exercise 16.19.

Let us now study two cases of linear classifiers, which allow for different types of adversarial examples. In the following two examples, the ground-truth classifier $g : \mathbb{R}^d \to \{-1, 1\}$ is given by $g(\boldsymbol{x}) = \operatorname{sign}(\overline{\boldsymbol{w}}^\top \boldsymbol{x})$ for $\overline{\boldsymbol{w}} \in \mathbb{R}^d$ with $\|\overline{\boldsymbol{w}}\| = 1$.

For the first example, we construct a Bayes classifier h admitting adversarial examples in the complement of the feature support. This corresponds to case (ii) in Section 16.2.

Example 16.6. Let $\mathcal{D}$ be the uniform distribution on

$$\{(\lambda \overline{\boldsymbol{w}}, g(\lambda \overline{\boldsymbol{w}})) \,|\, \lambda \in [-1, 1] \setminus \{0\}\} \subseteq \mathbb{R}^d \times \{-1, 1\}.$$

The feature support equals

$$D_{\boldsymbol{x}} = \{\lambda \overline{\boldsymbol{w}} \,|\, \lambda \in [-1, 1] \setminus \{0\}\} \subseteq \operatorname{span}\{\overline{\boldsymbol{w}}\}.$$

Next fix $\alpha \in (0, 1)$ and set $\boldsymbol{w} := \alpha \overline{\boldsymbol{w}} + (1 - \alpha)\boldsymbol{v}$ for some $\boldsymbol{v} \in \overline{\boldsymbol{w}}^\perp$ with $\|\boldsymbol{v}\| = 1$, so that $\|\boldsymbol{w}\| = 1$. We let $h(\boldsymbol{x}) := \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x})$. We now show that every $\boldsymbol{x} \in D_{\boldsymbol{x}}$ satisfies the assumptions of Theorem 16.4, and therefore admits an adversarial example.

Note that $h(\boldsymbol{x}) = g(\boldsymbol{x})$ for every $\boldsymbol{x} \in D_{\boldsymbol{x}}$. Hence h is a Bayes classifier. Now fix $\boldsymbol{x} \in D_{\boldsymbol{x}}$. Then $|\boldsymbol{w}^\top \boldsymbol{x}| \leq \alpha |\overline{\boldsymbol{w}}^\top \boldsymbol{x}|$, so that (16.3.2) is satisfied. Furthermore, for every $\varepsilon > 0$ it holds that

$$\delta := \frac{\varepsilon + |\boldsymbol{w}^\top \boldsymbol{x}|}{\|\boldsymbol{w}\|} \leq \varepsilon + \alpha.$$

Hence, for $\varepsilon < |\boldsymbol{w}^\top \boldsymbol{x}|$ it holds by Theorem 16.4 that there exists an adversarial example with perturbation less than $\varepsilon + \alpha$. For small α, the situation is depicted in the upper panel of Figure 16.2.

For the second example, we construct a distribution with global feature support and a classifier which is not a Bayes classifier. This corresponds to case (iv) in Section 16.2.

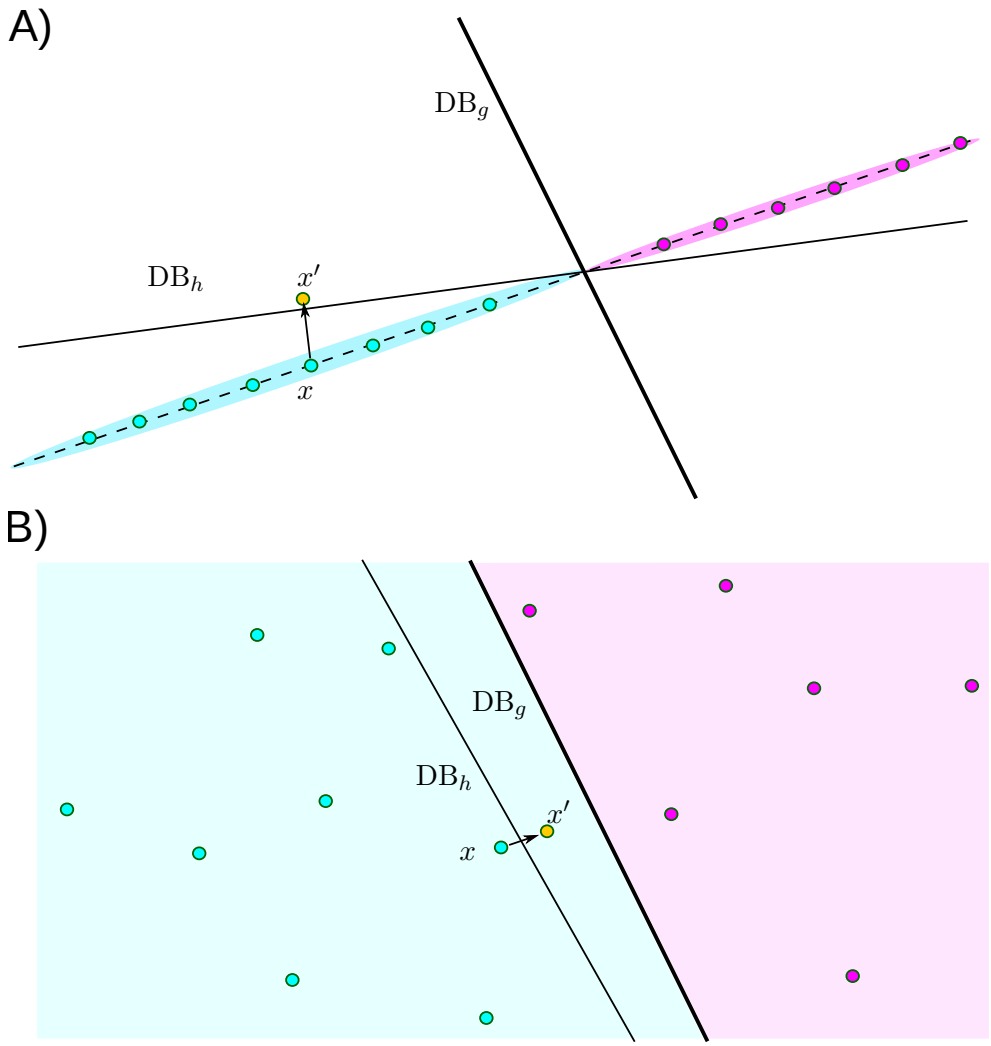


Figure 16.2: Illustration of the two types of adversarial examples in Examples 16.6 and 16.7. In panel A) the feature support $D_{\boldsymbol{x}}$ corresponds to the dashed line. We depict the two decision boundaries $\text{DB}_h = \{\boldsymbol{x} \,|\, \boldsymbol{w}^\top \boldsymbol{x} = 0\}$ of $h(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x})$ and $\text{DB}_g = \{\boldsymbol{x} \,|\, \overline{\boldsymbol{w}}^\top \boldsymbol{x} = 0\}$ $g(\boldsymbol{x}) = \text{sign}(\overline{\boldsymbol{w}}^\top \boldsymbol{x})$. Both h and g perfectly classify every data point in $D_{\boldsymbol{x}}$. One data point $\boldsymbol{x}$ is shifted outside of the support of the distribution in a way to change its label according to h. This creates an adversarial example $\boldsymbol{x}'$. In panel B) the data distribution is globally supported. However, h and g do not coincide. Thus the decision boundaries DB_h and DB_g do not coincide. Moving data points across DB_h can create adversarial examples, as depicted by $\boldsymbol{x}$ and $\boldsymbol{x}'$.

Example 16.7. Let $\mathcal{D}_{\boldsymbol{x}}$ be a distribution on $\mathbb{R}^d$ with positive Lebesgue density everywhere outside the decision boundary $\mathrm{DB}_g = \{\boldsymbol{x} \,|\, \overline{\boldsymbol{w}}^\top \boldsymbol{x} = 0\}$ of g. We define $\mathcal{D}$ to be the distribution of $(X, g(X))$ for $X \sim \mathcal{D}_{\boldsymbol{x}}$. In addition, let $\boldsymbol{w} \notin \{\pm\overline{\boldsymbol{w}}\}$, $\|\boldsymbol{w}\| = 1$ and $h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x})$. We exclude $\boldsymbol{w} = -\overline{\boldsymbol{w}}$ because, in this case, every prediction of h is wrong. Thus no adversarial examples are possible.

By construction the feature support is given by $D_{\boldsymbol{x}} = \mathbb{R}^d$. Moreover, $h^{-1}(\{\pm 1\})$ and $g^{-1}(\{\pm 1\})$ are half spaces, which implies that, in the notation of (16.2.2) that

$$\operatorname{dist}(C_{\pm 1} \cap D_{\boldsymbol{x}}, F_{\pm 1} \cap D_{\boldsymbol{x}}) = \operatorname{dist}(C_{\pm 1}, F_{\pm 1}) = 0.$$

Hence, for every $\delta > 0$ there is a positive probability of observing $\boldsymbol{x}$ to which an adversarial example with perturbation δ exists.

The situation is depicted in the lower panel of Figure 16.2.

16.4 ReLU neural networks

So far we discussed classification by affine classifiers. A binary classifier based on a ReLU neural network is a function $\mathbb{R}^d \ni \boldsymbol{x} \mapsto \operatorname{sign}(\Phi(\boldsymbol{x}))$, where Φ is a ReLU neural network. As noted in [223], the arguments for affine classifiers, see Proposition 16.5, can be applied to the affine pieces of Φ, to show existence of adversarial examples.

Consider a ground-truth classifier $g\colon \mathbb{R}^d \to \{-1, 0, 1\}$. For each $\boldsymbol{x} \in \mathbb{R}^d$ we define the geometric margin of g at $\boldsymbol{x}$ as

$$\mu_g(\boldsymbol{x}) := \operatorname{dist}(\boldsymbol{x}, g^{-1}(\{g(\boldsymbol{x})\})^c), \tag{16.4.1}$$

i.e., as the distance of $\boldsymbol{x}$ to the closest element that is classified differently from $\boldsymbol{x}$ or the infimum over all distances to elements from other classes if no closest element exists. Additionally, we denote the distance of $\boldsymbol{x}$ to the closest adjacent affine piece by

$$\nu_\Phi(\boldsymbol{x}) := \operatorname{dist}(\boldsymbol{x}, A^c_{\Phi,\boldsymbol{x}}), \tag{16.4.2}$$

where $A_{\Phi,\boldsymbol{x}}$ is the largest connected region on which Φ is affine and which contains $\boldsymbol{x}$. We have the following theorem.

Theorem 16.8. *Let $\Phi\colon \mathbb{R}^d \to \mathbb{R}$ and for $\boldsymbol{x} \in \mathbb{R}^d$ let $h(\boldsymbol{x}) = \operatorname{sign}(\Phi(\boldsymbol{x}))$. Denote by $g\colon \mathbb{R}^d \to \{-1, 0, 1\}$ the ground-truth classifier. Let $\boldsymbol{x} \in \mathbb{R}^d$ and $\varepsilon > 0$ be such that $\nu_\Phi(\boldsymbol{x}) > 0$, $g(\boldsymbol{x}) \neq 0$, $\nabla\Phi(\boldsymbol{x}) \neq 0$ and*

$$\mu_g(\boldsymbol{x}), \nu_\Phi(\boldsymbol{x}) > \frac{\varepsilon + |\Phi(\boldsymbol{x})|}{\|\nabla\Phi(\boldsymbol{x})\|}.$$

Then

$$\boldsymbol{x}' := \boldsymbol{x} - h(\boldsymbol{x}) \frac{\varepsilon + |\Phi(\boldsymbol{x})|}{\|\nabla\Phi(\boldsymbol{x})\|^2} \nabla\Phi(\boldsymbol{x})$$

is an adversarial example to $\boldsymbol{x}$ with perturbation $\delta = (\varepsilon + |\Phi(\boldsymbol{x})|)/\|\nabla\Phi(\boldsymbol{x})\|$.

Proof. We show that $\boldsymbol{x}'$ satisfies the properties in Definition 16.2.

By construction $\|\boldsymbol{x} - \boldsymbol{x}'\| \leq \delta$. Since $\mu_g(\boldsymbol{x}) > \delta$ it follows that $g(\boldsymbol{x}) = g(\boldsymbol{x}')$. Moreover, by assumption $g(\boldsymbol{x}) \neq 0$, and thus $g(\boldsymbol{x})g(\boldsymbol{x}') > 0$.

It only remains to show that $h(\boldsymbol{x}') \neq h(\boldsymbol{x})$. Since $\delta < \nu_\Phi(\boldsymbol{x})$, we have that $\Phi(\boldsymbol{x}) = \nabla\Phi(\boldsymbol{x})^\top \boldsymbol{x} + b$ and $\Phi(\boldsymbol{x}') = \nabla\Phi(\boldsymbol{x})^\top \boldsymbol{x}' + b$ for some $b \in \mathbb{R}$. Therefore,

$$\begin{aligned}\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}') = \nabla\Phi(\boldsymbol{x})^\top(\boldsymbol{x} - \boldsymbol{x}') &= \nabla\Phi(\boldsymbol{x})^\top \left(h(\boldsymbol{x})\frac{\varepsilon + |\Phi(\boldsymbol{x})|}{\|\nabla\Phi(\boldsymbol{x})\|^2}\nabla\Phi(\boldsymbol{x})\right) \\ &= h(\boldsymbol{x})(\varepsilon + |\Phi(\boldsymbol{x})|).\end{aligned}$$

Since $h(\boldsymbol{x})|\Phi(\boldsymbol{x})| = \Phi(\boldsymbol{x})$ it follows that $\Phi(\boldsymbol{x}') = -h(\boldsymbol{x})\varepsilon$. Hence, $h(\boldsymbol{x}') = -h(\boldsymbol{x})$, which completes the proof. □

Remark 16.9. We look at the key parameters in Theorem 16.8 to understand which factors facilitate adversarial examples.

- *The geometric margin of the ground-truth classifier $\mu_g(\boldsymbol{x})$:* To make the construction possible, we need to be sufficiently far away from points that belong to a different class than $\boldsymbol{x}$ or to the nonrelevant class.
- *The distance to the next affine piece $\nu_\Phi(\boldsymbol{x})$:* Since we are looking for an adversarial example within the same affine piece as $\boldsymbol{x}$, we need this piece to be sufficiently large.
- *The perturbation δ:* The perturbation is given by $(\varepsilon + |\Phi(\boldsymbol{x})|)/\|\nabla\Phi(\boldsymbol{x})\|$, which depends on the classification margin $|\Phi(\boldsymbol{x})|$ of the ReLU classifier and its sensitivity to inputs $\|\nabla\Phi(\boldsymbol{x})\|$. For adversarial examples to be possible, we either want a small classification margin of Φ or a high sensitivity of Φ to its inputs.

16.5 Robustness

Having established that adversarial examples can arise in various ways under mild assumptions, we now turn our attention to conditions that prevent their existence.

16.5.1 Global Lipschitz regularity

We have repeatedly observed in the previous sections that a large value of $\|\boldsymbol{w}\|$ for linear classifiers $\operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x})$, or $\|\nabla\Phi(\boldsymbol{x})\|$ for ReLU classifiers $\operatorname{sign}(\Phi(\boldsymbol{x}))$, facilitates the occurrence of adversarial examples. Naturally, both these values are upper bounded by the Lipschitz constant of the classifier's inner functions $\boldsymbol{x} \mapsto \boldsymbol{w}^\top \boldsymbol{x}$ and $\boldsymbol{x} \mapsto \Phi(\boldsymbol{x})$. Consequently, it was stipulated early on that bounding the Lipschitz constant of the inner functions could be an effective measure against adversarial examples [223].

We have the following result for general classifiers of the form $\boldsymbol{x} \mapsto \operatorname{sign}(\Phi(\boldsymbol{x}))$.

Proposition 16.10. *Let $\Phi\colon \mathbb{R}^d \to \mathbb{R}$ be C_L-Lipschitz with $C_L > 0$, and let $s > 0$. Let $h(\boldsymbol{x}) = \operatorname{sign}(\Phi(\boldsymbol{x}))$ be a classifier, and let $g\colon \mathbb{R}^d \to \{-1, 0, 1\}$ be a ground-truth classifier. Moreover, let $\boldsymbol{x} \in \mathbb{R}^d$ be such that*

$$\Phi(\boldsymbol{x})g(\boldsymbol{x}) \geq s. \tag{16.5.1}$$

Then there does not exist an adversarial example to $\boldsymbol{x}$ of perturbation $\delta < s/C_L$.

Proof. Let $\boldsymbol{x} \in \mathbb{R}^d$ satisfy (16.5.1) and assume that $\|\boldsymbol{x}' - \boldsymbol{x}\| \leq \delta$. The Lipschitz continuity of Φ implies

$$|\Phi(\boldsymbol{x}') - \Phi(\boldsymbol{x})| < s.$$

Since $|\Phi(\boldsymbol{x})| \geq s$ we conclude that $\Phi(\boldsymbol{x}')$ has the same sign as $\Phi(\boldsymbol{x})$ which shows that $\boldsymbol{x}'$ cannot be an adversarial example to $\boldsymbol{x}$. □

Remark 16.11. As we have seen in Lemma 13.2, we can bound the Lipschitz constant of ReLU neural networks by restricting the magnitude and number of their weights and the number of layers.

There has been some criticism to results of this form, see, e.g., [99], since an assumption on the Lipschitz constant may potentially restrict the capabilities of the neural network too much. We next present a result that shows under which assumptions on the training set, there exists a neural network that classifies the training set correctly, but does not allow for adversarial examples within the training set.

Theorem 16.12. *Let $m \in \mathbb{N}$, let $g\colon \mathbb{R}^d \to \{-1, 0, 1\}$ be a ground-truth classifier, and let $(\boldsymbol{x}_i, g(\boldsymbol{x}_i))_{i=1}^m \in (\mathbb{R}^d \times \{-1, 1\})^m$. Assume that*

$$\sup_{i \neq j} \frac{|g(\boldsymbol{x}_i) - g(\boldsymbol{x}_j)|}{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|} =: \widetilde{M} > 0.$$

Then there exists a ReLU neural network Φ with $\operatorname{depth}(\Phi) = O(\log(m))$ and $\operatorname{width}(\Phi) = O(dm)$ such that for all $i = 1, \dots, m$

$$\operatorname{sign}(\Phi(\boldsymbol{x}_i)) = g(\boldsymbol{x}_i)$$

and there is no adversarial example of perturbation $\delta = 1/\widetilde{M}$ to $\boldsymbol{x}_i$.

Proof. The result follows directly from Theorem 9.6 and Proposition 16.10. The reader is invited to complete the argument in Exercise 16.20. □

16.5.2 Local regularity

One issue with upper bounds involving global Lipschitz constants such as those in Proposition 16.10, is that these bounds may be quite large for deep neural networks. For example, the upper

bound given in Lemma 13.2 is

$$\|\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')\|_\infty \leq C_\sigma^L \cdot (B d_{\max})^{L+1} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty$$

which grows exponentially with the depth of the neural network. However, in practice this bound may be pessimistic, and locally the neural network might have significantly smaller gradients than the global Lipschitz constant.

Because of this, it is reasonable to study results preventing adversarial examples under *local* Lipschitz bounds. Such a result together with an algorithm providing bounds on the local Lipschitz constant was proposed in [88]. We state the theorem adapted to our set-up.

Theorem 16.13. *Let $h\colon \mathbb{R}^d \to \{-1, 1\}$ be a classifier of the form $h(\boldsymbol{x}) = \operatorname{sign}(\Phi(\boldsymbol{x}))$ and let $g\colon \mathbb{R}^d \to \{-1, 0, 1\}$ be the ground-truth classifier. Let $\boldsymbol{x} \in \mathbb{R}^d$ satisfy $g(\boldsymbol{x}) \neq 0$, and set*

$$\alpha := \max_{R>0} \min \left\{ \Phi(\boldsymbol{x}) g(\boldsymbol{x}) \Big/ \sup_{\substack{\|\boldsymbol{y}-\boldsymbol{x}\|_\infty \leq R \\ \boldsymbol{y} \neq \boldsymbol{x}}} \frac{|\Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_\infty}, R \right\}, \tag{16.5.2}$$

where the minimum is understood to be R in case the supremum is zero. Then there are no adversarial examples to $\boldsymbol{x}$ with perturbation $\delta < \alpha$.

Proof. Let $\boldsymbol{x} \in \mathbb{R}^d$ be as in the statement of the theorem. Assume, towards a contradiction, that for $0 < \delta < \alpha$ satisfying (16.5.2), there exists an adversarial example $\boldsymbol{x}'$ to $\boldsymbol{x}$ with perturbation δ.

If the supremum in (16.5.2) is zero, then Φ is constant on a ball of radius R around $\boldsymbol{x}$. In particular for $\|\boldsymbol{x}' - \boldsymbol{x}\| \leq \delta < R$ holds $h(\boldsymbol{x}') = h(\boldsymbol{x})$ and $\boldsymbol{x}'$ cannot be an adversarial example.

Now assume the supremum in (16.5.2) is not zero. It holds by (16.5.2), that

$$\delta < \Phi(\boldsymbol{x}) g(\boldsymbol{x}) \Big/ \sup_{\substack{\|\boldsymbol{y}-\boldsymbol{x}\|_\infty \leq R \\ \boldsymbol{y} \neq \boldsymbol{x}}} \frac{|\Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_\infty}. \tag{16.5.3}$$

Moreover,

$$\begin{aligned} |\Phi(\boldsymbol{x}') - \Phi(\boldsymbol{x})| &\leq \sup_{\substack{\|\boldsymbol{y}-\boldsymbol{x}\|_\infty \leq R \\ \boldsymbol{y} \neq \boldsymbol{x}}} \frac{|\Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_\infty} \|\boldsymbol{x} - \boldsymbol{x}'\|_\infty \\ &\leq \sup_{\substack{\|\boldsymbol{y}-\boldsymbol{x}\|_\infty \leq R \\ \boldsymbol{y} \neq \boldsymbol{x}}} \frac{|\Phi(\boldsymbol{y}) - \Phi(\boldsymbol{x})|}{\|\boldsymbol{x} - \boldsymbol{y}\|_\infty} \delta < \Phi(\boldsymbol{x}) g(\boldsymbol{x}), \end{aligned}$$

where we applied (16.5.3) in the last line. It follows that

$$\begin{aligned} g(\boldsymbol{x})\Phi(\boldsymbol{x}') &= g(\boldsymbol{x})\Phi(\boldsymbol{x}) + g(\boldsymbol{x})(\Phi(\boldsymbol{x}') - \Phi(\boldsymbol{x})) \\ &\geq g(\boldsymbol{x})\Phi(\boldsymbol{x}) - |\Phi(\boldsymbol{x}') - \Phi(\boldsymbol{x})| > 0. \end{aligned}$$

This rules out $\boldsymbol{x}'$ as an adversarial example. □

The supremum in (16.5.2) is bounded by the Lipschitz constant of Φ on $B_R(\boldsymbol{x})$. Thus Theorem 16.13 depends only on the local Lipschitz constant of Φ. One obvious criticism of this result is that the computation of (16.5.2) is potentially prohibitive. We next show a different result, for which the assumptions can immediately be checked by applying a simple algorithm that we present subsequently.

To state the following proposition, for a continuous function $\Phi : \mathbb{R}^d \to \mathbb{R}$ and $\delta > 0$ we define for $\boldsymbol{x} \in \mathbb{R}^d$ and $\delta > 0$

$$z^{\delta,\max} := \max\{\Phi(\boldsymbol{y}) \,|\, \|\boldsymbol{y} - \boldsymbol{x}\|_\infty \leq \delta\} \tag{16.5.4}$$

$$z^{\delta,\min} := \min\{\Phi(\boldsymbol{y}) \,|\, \|\boldsymbol{y} - \boldsymbol{x}\|_\infty \leq \delta\}. \tag{16.5.5}$$

Proposition 16.14. *Let $h\colon \mathbb{R}^d \to \{-1, 1\}$ be a classifier of the form $h(\boldsymbol{x}) = \operatorname{sign}(\Phi(\boldsymbol{x}))$ and $g\colon \mathbb{R}^d \to \{-1, 0, 1\}$, let $\boldsymbol{x}$ be such that $h(\boldsymbol{x}) = g(\boldsymbol{x})$. Then $\boldsymbol{x}$ does not have an adversarial example of perturbation δ if $z^{\delta,\max} z^{\delta,\min} > 0$.*

Proof. The proof is immediate, since $z^{\delta,\max} z^{\delta,\min} > 0$ implies that all points in a δ neighborhood of $\boldsymbol{x}$ are classified the same. □

To apply (16.14), we only have to compute $z^{\delta,\max}$ and $z^{\delta,\min}$. It turns out that if Φ is a neural network, then $z^{\delta,\max}$, $z^{\delta,\min}$ can be approximated by a computation similar to a forward pass of Φ. Denote by $|\boldsymbol{A}|$ the matrix obtained by taking the absolute value of each entry of the matrix $\boldsymbol{A}$. Additionally, we define

$$\boldsymbol{A}^+ = (|\boldsymbol{A}| + \boldsymbol{A})/2 \text{ and } \boldsymbol{A}^- = (|\boldsymbol{A}| - \boldsymbol{A})/2.$$

The idea behind the Algorithm 2 is common in the area of neural network verification, see, e.g., [66, 61, 7, 238].

Remark 16.15. Up to constants, Algorithm 2 has the same computational complexity as a forward pass, also see Algorithm 1. In addition, in contrast to upper bounds based on estimating the global Lipschitz constant of Φ via its weights, the upper bounds found via Algorithm 2 include the effect of the activation function σ. For example, if σ is the ReLU, then we may often end up in a situation, where $\delta^{(\ell),\mathrm{up}}$ or $\delta^{(\ell),\mathrm{low}}$ can have many entries that are 0. If an entry of $\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + \boldsymbol{b}^{(\ell)}$ is nonpositive, then it is guaranteed that the associated entry in $\delta^{(\ell),\mathrm{low}}$ will be zero. Similarly, if $\boldsymbol{W}^{(\ell)}$ has only few positive entries, then most of the entries of $\delta^{(\ell),\mathrm{up}}$ are not propagated to $\delta^{(\ell+1),\mathrm{up}}$.

Next, we prove that Algorithm 2 indeed produces sensible output.

Proposition 16.16. *Let Φ be a neural network with weight matrices $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1} \times d_\ell}$ and bias vectors $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ for $\ell = 0, \ldots, L$, and a monotonically increasing activation function σ.*

Let $\boldsymbol{x} \in \mathbb{R}^d$. Then the output of Algorithm 2 satisfies

$$\boldsymbol{x}^{L+1} + \delta^{(L+1),\mathrm{up}} > z^{\delta,\max} \text{ and } \boldsymbol{x}^{L+1} - \delta^{(L+1),\mathrm{low}} < z^{\delta,\min}.$$

Algorithm 2 Compute $\Phi(\boldsymbol{x})$, $z^{\delta,\max}$ and $z^{\delta,\min}$ for a given neural network

Input: weight matrices $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}\times d_\ell}$ and bias vectors $\boldsymbol{b}^{(\ell)} \in \mathbb{R}^{d_{\ell+1}}$ for $\ell = 0,\dots,L$ with $d_{L+1} = 1$, monotonous activation function σ, input vector $\boldsymbol{x} \in \mathbb{R}^{d_0}$, neighborhood size $\delta > 0$
Output: Bounds for $z^{\delta,\max}$ and $z^{\delta,\min}$

$\boldsymbol{x}^{(0)} = \boldsymbol{x}$
$\delta^{(0),\mathrm{up}} = \delta\mathbb{1} \in \mathbb{R}^{d_0}$
$\delta^{(0),\mathrm{low}} = \delta\mathbb{1} \in \mathbb{R}^{d_0}$
for $\ell = 0,\dots,L-1$ **do**
  $\boldsymbol{x}^{(\ell+1)} = \sigma(\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + \boldsymbol{b}^{(\ell)})$
  $\delta^{(\ell+1),\mathrm{up}} = \sigma(\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + (\boldsymbol{W}^{(\ell)})^+\delta^{(\ell),\mathrm{up}} + (\boldsymbol{W}^{(\ell)})^-\delta^{(\ell),\mathrm{low}} + \boldsymbol{b}^{(\ell)}) - \boldsymbol{x}^{(\ell+1)}$
  $\delta^{(\ell+1),\mathrm{low}} = \boldsymbol{x}^{(\ell+1)} - \sigma(\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} - (\boldsymbol{W}^{(\ell)})^+\delta^{(\ell),\mathrm{low}} - (\boldsymbol{W}^{(\ell)})^-\delta^{(\ell),\mathrm{up}} + \boldsymbol{b}^{(\ell)})$
end for
$\boldsymbol{x}^{(L+1)} = \boldsymbol{W}^{(L)}\boldsymbol{x}^{(L)} + \boldsymbol{b}^{(L)}$
$\delta^{(L+1),\mathrm{up}} = (\boldsymbol{W}^{(L)})^+\delta^{(L),\mathrm{up}} + (\boldsymbol{W}^{(L)})^-\delta^{(L),\mathrm{low}}$
$\delta^{(L+1),\mathrm{low}} = (\boldsymbol{W}^{(L)})^+\delta^{(L),\mathrm{low}} + (\boldsymbol{W}^{(L)})^-\delta^{(L),\mathrm{up}}$
return $\boldsymbol{x}^{(L+1)}$, $\boldsymbol{x}^{(L+1)} + \delta^{(L+1),\mathrm{up}}$, $\boldsymbol{x}^{(L+1)} - \delta^{(L+1),\mathrm{low}}$

Proof. Fix $\boldsymbol{y}$, $\boldsymbol{x} \in \mathbb{R}^d$ with $\|\boldsymbol{y}-\boldsymbol{x}\|_\infty \le \delta$ and let $\boldsymbol{y}^{(\ell)}$, $\boldsymbol{x}^{(\ell)}$ for $\ell = 0,\dots,L+1$ be as in Algorithm 2 applied to $\boldsymbol{y}$, $\boldsymbol{x}$, respectively. Moreover, let $\delta^{\ell,\mathrm{up}}$, $\delta^{\ell,\mathrm{low}}$ for $\ell = 0,\dots,L+1$ be as in Algorithm 2 applied to $\boldsymbol{x}$. We will prove by induction over $\ell = 0,\dots,L+1$ that

$$\boldsymbol{y}^{(\ell)} - \boldsymbol{x}^{(\ell)} \le \delta^{\ell,\mathrm{up}} \qquad \text{and} \qquad \boldsymbol{x}^{(\ell)} - \boldsymbol{y}^{(\ell)} \le \delta^{\ell,\mathrm{low}}, \tag{16.5.6}$$

where the inequalities are understood entry-wise for vectors. Since $\boldsymbol{y}$ was arbitrary this then proves the result.

The case $\ell = 0$ follows immediately from $\|\boldsymbol{y}-\boldsymbol{x}\|_\infty \le \delta$. Assume now, that the statement was shown for $\ell < L$. We have that

$$\begin{aligned}\boldsymbol{y}^{(\ell+1)} - \boldsymbol{x}^{(\ell+1)} - \delta^{\ell+1,\mathrm{up}} =& \sigma(\boldsymbol{W}^{(\ell)}\boldsymbol{y}^{(\ell)} + \boldsymbol{b}^{(\ell)}) \\ &- \sigma\big(\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + (\boldsymbol{W}^{(\ell)})^+\delta^{(\ell),\mathrm{up}} + (\boldsymbol{W}^{(\ell)})^-\delta^{(\ell),\mathrm{low}} + \boldsymbol{b}^{(\ell)}\big).\end{aligned}$$

The monotonicity of σ implies that

$$\boldsymbol{y}^{(\ell+1)} - \boldsymbol{x}^{(\ell+1)} \le \delta^{\ell+1,\mathrm{up}}$$

if

$$\boldsymbol{W}^{(\ell)}\boldsymbol{y}^{(\ell)} \le \boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} + (\boldsymbol{W}^{(\ell)})^+\delta^{(\ell),\mathrm{up}} + (\boldsymbol{W}^{(\ell)})^-\delta^{(\ell),\mathrm{low}}. \tag{16.5.7}$$

To prove (16.5.7), we observe that

$$\begin{aligned}\boldsymbol{W}^{(\ell)}(\boldsymbol{y}^{(\ell)} - \boldsymbol{x}^{(\ell)}) &= (\boldsymbol{W}^{(\ell)})^+(\boldsymbol{y}^{(\ell)} - \boldsymbol{x}^{(\ell)}) - (\boldsymbol{W}^{(\ell)})^-(\boldsymbol{y}^{(\ell)} - \boldsymbol{x}^{(\ell)}) \\ &= (\boldsymbol{W}^{(\ell)})^+(\boldsymbol{y}^{(\ell)} - \boldsymbol{x}^{(\ell)}) + (\boldsymbol{W}^{(\ell)})^-(\boldsymbol{x}^{(\ell)} - \boldsymbol{y}^{(\ell)}) \\ &\le (\boldsymbol{W}^{(\ell)})^+\delta^{(\ell),\mathrm{up}} + (\boldsymbol{W}^{(\ell)})^-\delta^{(\ell),\mathrm{low}},\end{aligned}$$

where we used the induction assumption in the last line. This shows the first estimate in (16.5.6). Similarly,

$$\begin{aligned}&\boldsymbol{x}^{(\ell+1)} - \boldsymbol{y}^{(\ell+1)} - \delta^{\ell+1,\mathrm{low}}\\ &= \sigma(\boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} - (\boldsymbol{W}^{(\ell)})^{+}\delta^{(\ell),\mathrm{low}} - (\boldsymbol{W}^{(\ell)})^{-}\delta^{(\ell),\mathrm{up}} + \boldsymbol{b}^{(\ell)}) - \sigma(\boldsymbol{W}^{(\ell)}\boldsymbol{y}^{(\ell)} + \boldsymbol{b}^{(\ell)}).\end{aligned}$$

Hence, $\boldsymbol{x}^{(\ell+1)} - \boldsymbol{y}^{(\ell+1)} \leq \delta^{\ell+1,\mathrm{low}}$ if

$$\boldsymbol{W}^{(\ell)}\boldsymbol{y}^{(\ell)} \geq \boldsymbol{W}^{(\ell)}\boldsymbol{x}^{(\ell)} - (\boldsymbol{W}^{(\ell)})^{+}\delta^{(\ell),\mathrm{low}} - (\boldsymbol{W}^{(\ell)})^{-}\delta^{(\ell),\mathrm{up}}. \tag{16.5.8}$$

To prove (16.5.8), we observe that

$$\begin{aligned}\boldsymbol{W}^{(\ell)}(\boldsymbol{x}^{(\ell)} - \boldsymbol{y}^{(\ell)}) &= (\boldsymbol{W}^{(\ell)})^{+}(\boldsymbol{x}^{(\ell)} - \boldsymbol{y}^{(\ell)}) - (\boldsymbol{W}^{(\ell)})^{-}(\boldsymbol{x}^{(\ell)} - \boldsymbol{y}^{(\ell)})\\ &= (\boldsymbol{W}^{(\ell)})^{+}(\boldsymbol{x}^{(\ell)} - \boldsymbol{y}^{(\ell)}) + (\boldsymbol{W}^{(\ell)})^{-}(\boldsymbol{y}^{(\ell)} - \boldsymbol{x}^{(\ell)})\\ &\leq (\boldsymbol{W}^{(\ell)})^{+}\delta^{(\ell),\mathrm{low}} + (\boldsymbol{W}^{(\ell)})^{-}\delta^{(\ell),\mathrm{up}},\end{aligned}$$

where we used the induction assumption in the last line. This completes the proof of (16.5.6) for all $\ell \leq L$.

The case $\ell = L + 1$ follows by the same argument, but replacing σ by the identity. □

Bibliography and further reading

This chapter begins with the foundational paper [223], but it should be remarked that adversarial examples for non-deep-learning models in machine learning were studied earlier in [98].

The results in this chapter are inspired by various results in the literature, though they may not be found in precisely the same form. The overall setup is inspired by [223]. The explanation based on the high-dimensionality of the data given in Section 16.3 was first formulated in [223] and [73]. The formalism reviewed in Section 16.2 is inspired by [218]. The results on robustness via local Lipschitz properties are due to [88]. Algorithm 2 is covered by results in the area of network verifiability [66, 61, 7, 238]. For a more comprehensive overview of modern approaches, we refer to the survey article [193].

Important directions not discussed in this chapter are the transferability of adversarial examples, defense mechanisms, and alternative adversarial operations. Transferability refers to the phenomenon that adversarial examples for one model often also fool other models, [170, 153]. Defense mechanisms, i.e., techniques for specifically training a neural network to prevent adversarial examples, include for example the Fast Gradient Sign Method of [73], and more sophisticated recent approaches such as [32]. Finally, adversarial examples can be generated not only through additive perturbations, but also through smooth transformations of images, as demonstrated in [1, 243].

Exercises

Exercise 16.17. Prove (16.3.1) by comparing the volume of the d-dimensional Euclidean unit ball with the volume of the d-dimensional 1-ball of radius c for a given $c > 0$.

Exercise 16.18. Fix $\delta > 0$. For a pair of classifiers h and g such that $C_1 \cup C_{-1} = \emptyset$ in (16.2.2), there trivially cannot exist any adversarial examples. Construct an example, of h, g, $\mathcal{D}$ such that C_1, $C_{-1} \neq \emptyset$, h is not a Bayes classifier, and g is such that no adversarial examples with a perturbation δ exist.

Is this also possible if $g^{-1}(0) = \emptyset$?

Exercise 16.19. Prove Proposition 16.5.

Hint: Repeat the proof of Theorem 16.4. In the first part set $\boldsymbol{x}^{(\text{ext})} = (\boldsymbol{x}, 1)$, $\boldsymbol{w}^{(\text{ext})} = (\boldsymbol{w}, b)$ and $\overline{\boldsymbol{w}}^{(\text{ext})} = (\overline{\boldsymbol{w}}, \overline{b})$. Then show that $h(\boldsymbol{x}') \neq h(\boldsymbol{x})$ by plugging in the definition of $\boldsymbol{x}'$.

Exercise 16.20. Complete the proof of Theorem 16.12.

Appendix A

Probability theory

This appendix provides some basic notions and results in probability theory required in the main text. It is intended as a revision for a reader already familiar with these concepts. For more details and proofs, we refer for example to the standard textbook [117].

A.1 Sigma-algebras, topologies, and measures

Let Ω be a set, and denote by 2^Ω the powerset of Ω.

Definition A.1. A subset $\mathfrak{A} \subseteq 2^\Omega$ is called a **sigma-algebra**[1] on Ω if it satisfies

(i) $\Omega \in \mathfrak{A}$,

(ii) $A^c \in \mathfrak{A}$ whenever $A \in \mathfrak{A}$,

(iii) $\bigcup_{i\in\mathbb{N}} A_i \in \mathfrak{A}$ whenever $A_i \in \mathfrak{A}$ for all $i \in \mathbb{N}$.

For a sigma-algebra $\mathfrak{A}$ on Ω, the tuple $(\Omega, \mathfrak{A})$ is also referred to as a **measurable space**. For a measurable space, a subset $A \subseteq \Omega$ is called **measurable**, if $A \in \mathfrak{A}$. Measurable sets are also called **events**.

Another key system of subsets of Ω is that of a topology.

Definition A.2. A subset $\mathfrak{A} \subseteq 2^\Omega$ is called a **topology** on Ω if it satisfies

(i) $\emptyset, \Omega \in \mathfrak{T}$,

(ii) $\bigcap_{j=1}^n O_j \in \mathfrak{T}$ whenever $n \in \mathbb{N}$ and $O_1, \dots, O_n \in \mathfrak{T}$,

(iii) $\bigcup_{i\in I} O_i \in \mathfrak{T}$ whenever for an index set I holds $O_i \in \mathfrak{T}$ for all $i \in I$.

If $\mathfrak{T}$ is a topology on Ω, we call $(\Omega, \mathfrak{T})$ a **topological space**, and a set $O \subseteq \Omega$ is called **open** if and only if $O \in \mathfrak{T}$.

Remark A.3. The two notions differ in that a topology allows for unions of *arbitrary* (possibly uncountably many) sets, but only for *finite* intersection, whereas a sigma-algebra allows for countable unions and intersections.

Example A.4. Let $d \in \mathbb{N}$ and denote by $B_\varepsilon(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^d \,|\, \|\boldsymbol{y} - \boldsymbol{x}\| < \varepsilon\}$ the set of points whose Euclidean distance to $\boldsymbol{x}$ is less than ε. Then for every $A \subseteq \mathbb{R}^d$, the smallest topology on A containing $A \cap B_\varepsilon(\boldsymbol{x})$ for all $\varepsilon > 0$, $\boldsymbol{x} \in \mathbb{R}^d$, is called the **Euclidean topology** on A.

If $(\Omega, \mathfrak{T})$ is a topological space, then the **Borel sigma-algebra** refers to the smallest sigma-algebra on Ω containing all open sets, i.e. all elements of $\mathfrak{T}$. Throughout this book, subsets of $\mathbb{R}^d$ are always understood to be equipped with the Euclidean topology and the Borel sigma-algebra. The Borel sigma-algebra on $\mathbb{R}^d$ is denoted by $\mathfrak{B}_d$.

We can now introduce measures.

Definition A.5. Let $(\Omega, \mathfrak{A})$ be a measurable space. A mapping $\mu : \mathfrak{A} \to [0, \infty]$ is called a **measure** if it satisfies

(i) $\mu(\emptyset) = 0$,

(ii) for every sequence $(A_i)_{i\in\mathbb{N}} \subseteq \mathfrak{A}$ such that $A_i \cap A_j = \emptyset$ whenever $i \neq j$, it holds

$$\mu\Big(\bigcup_{i\in\mathbb{N}} A_i\Big) = \sum_{i\in\mathbb{N}} \mu(A_i).$$

We say that the measure is **finite** if $\mu(\Omega) < \infty$, and it is **sigma-finite** if there exists a sequence $(A_i)_{i\in\mathbb{N}} \subseteq \mathfrak{A}$ such that $\Omega = \bigcup_{i\in\mathbb{N}} A_i$ and $\mu(A_i) < 1$ for all $i \in \mathbb{N}$. In case $\mu(\Omega) = 1$, the measure is called a **probability measure**.

Example A.6. One can show that there exists a unique measure λ on $(\mathbb{R}^d, \mathfrak{B}_d)$, such that for all sets of the type $\times_{j=1}^d [a_i, b_i)$ with $-\infty < a_i \leq b_i < \infty$ holds

$$\lambda(\times_{i=1}^d [a_i, b_i)) = \prod_{i=1}^d (b_i - a_i).$$

This measure is called the **Lebesgue measure**.

If μ is a measure on the measurable space $(\Omega, \mathfrak{A})$, then the triplet $(\Omega, \mathfrak{A}, \mu)$ is called a **measure space**. In case μ is a probability measure, it is called a **probability space**.

Let $(\Omega, \mathfrak{A}, \mu)$ be a measure space. A subset $N \subseteq \Omega$ is called a **null-set**, if N is measurable and $\mu(N) = 0$. Moreover, an equality or inequality is said to hold μ-**almost everywhere** or μ-**almost surely**, if it is satisfied on the complement of a null-set. In case μ is clear from context, we simply write "almost everywhere" or "almost surely" instead. Usually this refers to the Lebesgue measure.

A.2 Random variables

A.2.1 Measurability of functions

To define random variables, we first need to recall the measurability of functions.

Definition A.7. Let $(\Omega_1, \mathfrak{A}_1)$ and $(\Omega_2, \mathfrak{A}_2)$ be two measurable spaces. A function $f : \Omega_1 \to \Omega_2$ is called **measurable** if

$$f^{-1}(A_2) := \{\omega \in \Omega_1 \mid f(\omega) \in A_2\} \in \mathfrak{A}_1 \qquad \text{for all } A_2 \in \mathfrak{A}_2.$$

A mapping $X : \Omega_1 \to \Omega_2$ is called a Ω_2**-valued random variable** if it is measurable.

Remark A.8. We again point out the parallels to topological spaces: A function $f : \Omega_1 \to \Omega_2$ between two topological spaces $(\Omega_1, \mathfrak{T}_1)$ and $(\Omega_2, \mathfrak{T}_2)$ is called **continuous** if $f^{-1}(O_2) \in \mathfrak{T}_1$ for all $O_2 \in \mathfrak{T}_2$.

Let Ω_1 be a set and let $(\Omega_2, \mathfrak{A}_2)$ be a measurable space. For $X : \Omega_1 \to \Omega_2$, we can ask for the smallest sigma-algebra $\mathfrak{A}_X$ on Ω_1, such that X is measurable as a mapping from $(\Omega_1, \mathfrak{A}_X)$ to $(\Omega_2, \mathfrak{A}_2)$. Clearly, for every sigma-algebra $\mathfrak{A}_1$ on Ω_1, X is measurable as a mapping from $(\Omega_1, \mathfrak{A}_1)$ to $(\Omega_2, \mathfrak{A}_2)$ if and only if every $A \in \mathfrak{A}_X$ belongs to $\mathfrak{A}_1$; or in other words, $\mathfrak{A}_X$ is a sub sigma-algebra of $\mathfrak{A}_1$. It is easy to check that $\mathfrak{A}_X$ is given through the following definition.

Definition A.9. Let $X : \Omega_1 \to \Omega_2$ be a random variable. Then

$$\mathfrak{A}_X := \{X^{-1}(A_2) \mid A_2 \in \mathfrak{A}_2\} \subseteq 2^{\Omega_1}$$

is the **sigma-algebra induced by** X on Ω_1.

A.2.2 Distribution and expectation

Now let $(\Omega_1, \mathfrak{A}_1, \mathbb{P})$ be a probability space, and let $(\Omega_2, \mathfrak{A}_2)$ be a measurable space. Then X naturally induces a measure on $(\Omega_2, \mathfrak{A}_2)$ via

$$\mathbb{P}_X[A_2] := \mathbb{P}[X^{-1}(A_2)] \qquad \text{for all } A_2 \in \mathfrak{A}_2.$$

Note that due to the measurability of X it holds $X^{-1}(A_2) \in \mathfrak{A}_1$, so that $\mathbb{P}_X$ is well-defined.

Definition A.10. The measure $\mathbb{P}_X$ is called the **distribution** of X. If $(\Omega_2, \mathfrak{A}_2) = (\mathbb{R}^d, \mathfrak{B}_d)$, and there exists a function $f_X : \mathbb{R}^d \to \mathbb{R}$ such that

$$\mathbb{P}[A] = \int_A f_X(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \qquad \text{for all } A \in \mathfrak{B}_d,$$

then f_X is called the **(Lebesgue) density** of X.

Remark A.11. The term distribution is often used without specifying an underlying probability space and random variable. In this case, "distribution" stands interchangeably for "probability

measure". For example, μ *is a distribution on* Ω_2 states that μ is a probability measure on the measurable space $(\Omega_2, \mathfrak{A}_2)$. In this case, there always exists a probability space $(\Omega_1, \mathfrak{A}_1, \mathbb{P})$ and a random variable $X : \Omega_1 \to \Omega_2$ such that $\mathbb{P}_X = \mu$; namely $(\Omega_1, \mathfrak{A}_1, \mathbb{P}) = (\Omega_2, \mathfrak{A}_2, \mu)$ and $X(\omega) = \omega$.

Example A.12. Some important distributions include the following.

- **Bernoulli distribution**: A random variable $X : \Omega \to \{0, 1\}$ is Bernoulli distributed if there exists $p \in [0, 1]$ such that $\mathbb{P}[X = 1] = p$ and $\mathbb{P}[X = 0] = 1 - p$.
- **Uniform distribution**: A random variable $X : \Omega \to \mathbb{R}^d$ is uniformly distributed on a measurable set $A \in \mathfrak{B}_d$, if its density equals

  $$f_X(\boldsymbol{x}) = \frac{1}{|A|} \mathbb{1}_A(\boldsymbol{x})$$

  where $|A| < \infty$ is the Lebesgue measure of A.
- **Gaussian distribution**: A random variable $X : \Omega \to \mathbb{R}^d$ is Gaussian distributed with mean $\boldsymbol{m} \in \mathbb{R}^d$ and the regular covariance matrix $\boldsymbol{C} \in \mathbb{R}^{d \times d}$, if its density equals

  $$f_X(\boldsymbol{x}) = \frac{1}{(2\pi \det(\boldsymbol{C}))^{d/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m})^\top \boldsymbol{C}^{-1}(\boldsymbol{x} - \boldsymbol{m})\right).$$

  We denote this distribution by $\mathrm{N}(\boldsymbol{m}, \boldsymbol{C})$.

Let $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space, let $X : \Omega \to \mathbb{R}^d$ be an $\mathbb{R}^d$-valued random variable. We then call the Lebesgue integral

$$\mathbb{E}[X] := \int_\Omega X(\omega) \, \mathrm{d}\mathbb{P}(\omega) = \int_{\mathbb{R}^d} \boldsymbol{x} \, \mathrm{d}\mathbb{P}_X(\boldsymbol{x}) \tag{A.2.1}$$

the **expectation** of X. Moreover, for $k \in \mathbb{N}$ we say that X has **finite k-th moment** if $\mathbb{E}[\|X\|^k] < \infty$. Similarly, for a probability measure μ on $\mathbb{R}^d$ and $k \in \mathbb{N}$, we say that μ has finite k-th moment if

$$\int_{\mathbb{R}^d} \|\boldsymbol{x}\|^k \, \mathrm{d}\mu(\boldsymbol{x}) < \infty.$$

Furthermore, the matrix

$$\int_\Omega (X(\omega) - \mathbb{E}[X])(X(\omega) - \mathbb{E}[X])^\top \, \mathrm{d}\mathbb{P}(\omega) \in \mathbb{R}^{d \times d}$$

is the **covariance** of $X : \Omega \to \mathbb{R}^d$. For $d = 1$, it is called the **variance** of X and denoted by $\mathbb{V}[X]$.

Finally, we recall different variants of convergence for random variables.

Definition A.13. Let $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space, and let $X_j : \Omega \to \mathbb{R}^d$, $j \in \mathbb{N}$, be a sequence of random variables and let $X : \Omega \to \mathbb{R}^d$ also be a random variable. The sequence is said to

(i) **converge almost surely** to X, if

$$\mathbb{P}\left[\left\{\omega \in \Omega \,\middle|\, \lim_{j \to \infty} X_j(\omega) = X(\omega)\right\}\right] = 1,$$

(ii) **converge in probability** to X, if

$$\text{for all } \varepsilon > 0 : \quad \lim_{j\to\infty} \mathbb{P}\left[\{\omega \in \Omega \,|\, |X_j(\omega) - X(\omega)| > \varepsilon\}\right] = 0,$$

(iii) **converge weakly** to X, if for all bounded continuous functions $f : \mathbb{R}^d \to \mathbb{R}$ holds

$$\lim_{j\to\infty} \mathbb{E}[f \circ X_j] = \mathbb{E}[f \circ X].$$

The notions in Definition A.13 are ordered by decreasing strength, i.e. almost sure convergence implies convergence in probability, and convergence in probability implies weak convergence, see for example [117, Chapter 13]. Since $\mathbb{E}[f \circ X] = \int_{\mathbb{R}^d} f(x)\,\mathrm{d}\mathbb{P}_X(x)$, the notion of weak convergence only depends on the distribution $\mathbb{P}_X$ of X. We thus also say that a sequence of random variables converges weakly towards a measure μ.

A.3 Conditionals, marginals, and independence

In this section, we concentrate on $\mathbb{R}^d$-valued random variables, although the following concepts can be extended to more general spaces.

A.3.1 Joint and marginal distribution

Let again $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space, and let $X : \Omega \to \mathbb{R}^{d_X}$, $Y : \Omega \to \mathbb{R}^{d_Y}$ be two random variables. Then

$$Z := (X, Y) : \Omega \to \mathbb{R}^{d_X + d_Y}$$

is also a random variable. Its distribution $\mathbb{P}_Z$ is a measure on the measurable space $(\mathbb{R}^{d_X+d_Y}, \mathfrak{B}_{d_X+d_Y})$, and $\mathbb{P}_Z$ is referred to as the **joint distribution** of X and Y. On the other hand, $\mathbb{P}_X$, $\mathbb{P}_Y$ are called the **marginal distributions** of X, Y. Note that

$$\mathbb{P}_X[A] = \mathbb{P}_Z[A \times \mathbb{R}^{d_Y}] \qquad \text{for all } A \in \mathfrak{B}_{d_X},$$

and similarly for $\mathbb{P}_Y$. Thus the marginals $\mathbb{P}_X$, $\mathbb{P}_Y$, can be constructed from the joint distribution $\mathbb{P}_Z$. In turn, knowledge of the marginals is not sufficient to construct the joint distribution.

A.3.2 Independence

The concept of independence serves to formalize the situation, where knowledge of one random variable provides no information about another random variable. We first give the formal definition, and afterwards discuss the roll of a die as a simple example.

Definition A.14. Let $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space. Then two events $A,\ B \in \mathfrak{A}$ are called **independent** if

$$\mathbb{P}[A \cap B] = \mathbb{P}[A]\mathbb{P}[B].$$

Two random variables $X : \Omega \to \mathbb{R}^{d_X}$ and $Y : \Omega \to \mathbb{R}^{d_Y}$ are called **independent**, if

$$A,\ B \text{ are independent for all } A \in \mathfrak{A}_X,\ B \in \mathfrak{A}_Y.$$

Two random variables are thus independent, if and only if all events in their induced sigma-algebras are independent. This turns out to be equivalent to the joint distribution $\mathbb{P}_{(X,Y)}$ being equal to the product measure $\mathbb{P}_X \otimes \mathbb{P}_Y$; the latter is characterized as the unique measure μ on $\mathbb{R}^{d_X+d_Y}$ satisfying $\mu(A \times B) = \mathbb{P}_X[A]\mathbb{P}_Y[B]$ for all $A \in \mathfrak{B}_{d_x}$, $B \in \mathfrak{B}_{d_Y}$.

Example A.15. Let $\Omega = \{1, \dots, 6\}$ represent the outcomes of rolling a fair die, let $\mathfrak{A} = 2^\Omega$ be the sigma-algebra, and let $\mathbb{P}[\omega] = 1/6$ for all $\omega \in \Omega$. Consider the three random variables

$$X_1(\omega) = \begin{cases} 0 & \text{if } \omega \text{ is odd} \\ 1 & \text{if } \omega \text{ is even} \end{cases} \quad X_2(\omega) = \begin{cases} 0 & \text{if } \omega \leq 3 \\ 1 & \text{if } \omega \geq 4 \end{cases} \quad X_3(\omega) = \begin{cases} 0 & \text{if } \omega \in \{1,2\} \\ 1 & \text{if } \omega \in \{3,4\} \\ 2 & \text{if } \omega \in \{5,6\}. \end{cases}$$

These random variables can be interpreted as follows:

- X_1 indicates whether the roll yields an odd or even number.
- X_2 indicates whether the roll yields a number at most 3 or at least 4.
- X_3 categorizes the roll into one of the groups $\{1,2\}$, $\{3,4\}$ or $\{5,6\}$.

The induced sigma-algebras are

$$\begin{aligned} \mathfrak{A}_{X_1} &= \{\emptyset, \Omega, \{1,3,5\}, \{2,4,6\}\} \\ \mathfrak{A}_{X_2} &= \{\emptyset, \Omega, \{1,2,3\}, \{4,5,6\}\} \\ \mathfrak{A}_{X_3} &= \{\emptyset, \Omega, \{1,2\}, \{3,4\}, \{5,6\}, \{1,2,3,4\}, \{1,2,5,6\}, \{3,4,5,6\}\}. \end{aligned}$$

We leave it to the reader to formally check that X_1 and X_2 are not independent, but X_1 and X_3 are independent. This reflects the fact that, for example, knowing the outcome to be odd, makes it more likely that the number belongs to $\{1,2,3\}$ rather than $\{4,5,6\}$. However, this knowledge provides no information on the three categories $\{1,2\}$, $\{3,4\}$, and $\{5,6\}$.

If $X : \Omega \to \mathbb{R}$, $Y : \Omega \to \mathbb{R}$ are two independent random variables, then, due to $\mathbb{P}_{(X,Y)} = \mathbb{P}_X \otimes \mathbb{P}_Y$

$$\begin{aligned} \mathbb{E}[XY] &= \int_\Omega X(\omega)Y(\omega)\,\mathrm{d}\mathbb{P}(\omega) \\ &= \int_{\mathbb{R}^2} xy\,\mathrm{d}\mathbb{P}_{(X,Y)}(x,y) \\ &= \int_{\mathbb{R}} x\,\mathrm{d}\mathbb{P}_X(x) \int_{\mathbb{R}} y\,\mathrm{d}\mathbb{P}_X(y) \\ &= \mathbb{E}[X]\mathbb{E}[Y]. \end{aligned}$$

Using this observation, it is easy to see that for a sequence of independent $\mathbb{R}$-valued random variables $(X_i)_{i=1}^n$ with bounded second moments, there holds **Bienaymé's identity**

$$\mathbb{V}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{V}\left[X_i\right]. \tag{A.3.1}$$

A.3.3 Conditional distributions

Let $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space, and let A, $B \in \mathfrak{A}$ be two events. In case $\mathbb{P}[B] > 0$, we define

$$\mathbb{P}[A|B] := \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}, \tag{A.3.2}$$

and call $\mathbb{P}[A|B]$ the **conditional probability of A given B.**

Example A.16. Consider the setting of Example A.15. Let $A = \{\omega \in \Omega \,|\, X_1(\omega) = 0\}$ be the event that the outcome of the die roll was an odd number and let $B = \{\omega \in \Omega \,|\, X_2(\omega) = 0\}$ be the event that the outcome yielded a number at most 3. Then $\mathbb{P}[B] = 1/2$, and $\mathbb{P}[A \cap B] = 1/3$. Thus

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]} = \frac{1/3}{1/2} = \frac{2}{3}.$$

This reflects that, given we know the outcome to be at most 3, the probability of the number being odd, i.e. in $\{1, 3\}$, is larger than the probability of the number being even, i.e. equal to 2.

The conditional probability in (A.3.2) is only well-defined if $\mathbb{P}[B] > 0$. In practice, we often encounter the case where we would like to condition on an event of probability zero.

Example A.17. Consider the following procedure: We first draw a random number $p \in [0, 1]$ according to a uniform distribution on $[0, 1]$. Afterwards we draw a random number $X \in \{0, 1\}$ according to a p-Bernoulli distribution, i.e. $\mathbb{P}[X = 1] = p$ and $\mathbb{P}[X = 0] = 1 - p$. Then (p, X) is a joint random variable taking values in $[0, 1] \times \{0, 1\}$. What is $\mathbb{P}[X = 1|p = 0.5]$ in this case? Intuitively, it should be $1/2$, but note that $\mathbb{P}[p = 0.5] = 0$, so that (A.3.2) is not meaningful here.

Definition A.18 (regular conditional distribution)**.** Let $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space, and let $X : \Omega \to \mathbb{R}^{d_X}$ and $Y : \Omega \to \mathbb{R}^{d_Y}$ be two random variables. Let $\tau_{X|Y} : \mathfrak{B}_{d_X} \times \mathbb{R}^{d_Y} \to [0, 1]$ satisfy

(i) $y \mapsto \tau_{X|Y}(A, y) : \mathbb{R}^{d_Y} \to [0, 1]$ is measurable for every fixed $A \in \mathfrak{B}_{d_X}$,

(ii) $A \mapsto \tau_{X|Y}(A, y)$ is a probability measure on $(\mathbb{R}^{d_X}, \mathfrak{B}_{d_X})$ for every $y \in Y(\Omega)$,

(iii) for all $A \in \mathfrak{B}_{d_X}$ and all $B \in \mathfrak{B}_{d_Y}$ holds

$$\mathbb{P}[X \in A, Y \in B] = \int_B \tau_{X|Y}(A, y)\mathbb{P}_Y(y).$$

Then τ is called a **regular (version of the) conditional distribution of X given Y.** In this case, we denote

$$\mathbb{P}[X \in A|Y = y] := \tau_{X|Y}(A, y),$$

and refer to this measure as the conditional distribution of $X|Y = y$.

Definition A.18 provides a mathematically rigorous way of assigning a distribution to a random variable conditioned on an event that may have probability zero, as in Example A.17. Existence and uniqueness of these conditional distributions hold in the following sense, see for example [117, Chapter 8] or [201, Chapter 3] for the specific statement given here.

Theorem A.19. *Let $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space, and let $X : \Omega \to \mathbb{R}^{d_X}$, $Y : \Omega \to \mathbb{R}^{d_Y}$ be two random variables. Then there exists a regular version of the conditional distribution τ_1.*

Let τ_2 be another regular version of the conditional distribution. Then there exists a $\mathbb{P}_Y$-null set $N \subseteq \mathbb{R}^{d_Y}$, such that for all $y \in N^c \cap Y(\Omega)$, the two probability measures $\tau_1(\cdot, y)$ and $\tau_2(\cdot, y)$ coincide.

In particular, conditional distributions are only well-defined in a $\mathbb{P}_Y$-almost everywhere sense.

Definition A.20. Let $(\Omega, \mathfrak{A}, \mathbb{P})$ be a probability space, and let $X : \Omega \to \mathbb{R}^{d_X}$, $Y : \Omega \to \mathbb{R}^{d_Y}$, $Z : \Omega \to \mathbb{R}^{d_Z}$ be three random variables. We say that X and Z are **conditionally independent given** Y, if the two distributions $X|Y = y$ and $Z|Y = y$ are independent for $\mathbb{P}_Y$-almost every $y \in Y(\Omega)$.

A.4 Concentration inequalities

Let $X_i : \Omega \to \mathbb{R}$, $i \in \mathbb{N}$, be a sequence of random variables with finite first moments. The centered average over the first n terms

$$S_n := \frac{1}{n} \sum_{i=1}^{n} (X_i - \mathbb{E}[X_i]) \tag{A.4.1}$$

is another random variable, and by linearity of the expectation it holds $\mathbb{E}[S_n] = 0$. The sequence is said to satisfy the **strong law of large numbers** if

$$\mathbb{P}\Big[\limsup_{n \to \infty} |S_n| = 0 \Big] = 1.$$

This is for example the case if there exists $C < \infty$ such that $\mathbb{V}[X_i] \leq C$ for all $i \in \mathbb{N}$. Concentration inequalities provide bounds on the rate of this convergence.

We start with Markov's inequality.

Lemma A.21 (Markov's inequality)**.** *Let $X : \Omega \to \mathbb{R}$ be a random variable, and let $\varphi : [0, \infty) \to [0, \infty)$ be monotonically increasing. Then for all $\varepsilon > 0$*

$$\mathbb{P}[|X| \geq \varepsilon] \leq \frac{\mathbb{E}[\varphi(|X|)]}{\varphi(\varepsilon)}.$$

Proof. We have

$$\mathbb{P}[|X| \geq \varepsilon] = \int_{X^{-1}([\varepsilon,\infty))} 1 \,\mathrm{d}\mathbb{P}(\omega) \leq \int_{\Omega} \frac{\varphi(|X(\omega)|)}{\varphi(\varepsilon)} \,\mathrm{d}\mathbb{P}(\omega) = \frac{\mathbb{E}[\varphi(|X|)]}{\varphi(\varepsilon)},$$

which gives the claim. □

Applying Markov's inequality with $\varphi(x) := x^2$ to the random variable $X - \mathbb{E}[X]$ directly gives Chebyshev's inequality.

Lemma A.22 (Chebyshev's inequality)**.** *Let $X : \Omega \to \mathbb{R}$ be a random variable with finite variance. Then for all $\varepsilon > 0$*

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq \varepsilon] \leq \frac{\mathbb{V}[X]}{\varepsilon^2}.$$

From Chebyshev's inequality we obtain the next result, which is a quite general concentration inequality for random variables with finite variances.

Theorem A.23. *Let $X_1, \ldots, X_n$ be $n \in \mathbb{N}$ independent real-valued random variables such that for some $\varsigma > 0$ holds $\mathbb{E}[|X_i - \mu|^2] \leq \varsigma^2$ for all $i = 1, \ldots, n$. Denote*

$$\mu := \mathbb{E}\Big[\frac{1}{n}\sum_{j=1}^{n} X_j\Big]. \tag{A.4.2}$$

Then for all $\varepsilon > 0$

$$\mathbb{P}\left[\Big|\frac{1}{n}\sum_{j=1}^{n} X_i - \mu\Big| \geq \varepsilon\right] \leq \frac{\varsigma^2}{\varepsilon^2 n}.$$

Proof. Let $S_n = \sum_{j=1}^{n}(X_i - \mathbb{E}[X_i])/n = (\sum_{j=1}^{n} X_i)/n - \mu$. By Bienaymé's identity (A.3.1), it holds that

$$\mathbb{V}[S_n] = \frac{1}{n^2}\sum_{j=1}^{n} \mathbb{E}[(X_i - \mathbb{E}[X_i])^2] \leq \frac{\varsigma^2}{n}.$$

Since $\mathbb{E}[S_n] = 0$, Chebyshev's inequality applied to S_n gives the statement. □

If we have additional information about the random variables, then we can derive sharper bounds. In case of uniformly bounded random variables (rather than just bounded variance), Hoeffding's inequality, which we recall next, shows an exponential rate of concentration around the mean.

Theorem A.24 (Hoeffding's inequality)**.** *Let a, $b \in \mathbb{R}$. Let $X_1, \dots, X_n$ be $n \in \mathbb{N}$ independent random real-valued variables such that $a \leq X_i \leq b$ almost surely for all $i = 1, \dots, n$, and let μ be as in* (A.4.2)*. Then, for every $\varepsilon > 0$*

$$\mathbb{P}\left[\left|\frac{1}{n}\sum_{j=1}^{n} X_j - \mu\right| > \varepsilon\right] \leq 2e^{-\frac{2n\varepsilon^2}{(b-a)^2}}.$$

A proof can, for example, be found in [212, Section B.4], where this version is also taken from.

Finally, we recall the central limit theorem, in its multivariate formulation. We say that $(X_j)_{j\in\mathbb{N}}$ is an **i.i.d. sequence of random variables**, if the random variables are (pairwise) independent and identically distributed. For a proof see [117, Theorem 15.58].

Theorem A.25 (Multivariate central limit theorem)**.** *Let $(X_n)_{n\in\mathbb{N}}$ be an i.i.d. sequence of $\mathbb{R}^d$-valued random variables, such that $\mathbb{E}[X_n] = \mathbf{0} \in \mathbb{R}^d$ and $\mathbb{E}[X_{n,i}X_{n,j}] = C_{ij}$ for all i, $j = 1, \dots, d$. Let*

$$Y_n := \frac{X_1 + \cdots + X_n}{\sqrt{n}}.$$

Then Y_n converges weakly to $\mathrm{N}(\mathbf{0}, \boldsymbol{C})$ as $n \to \infty$.

Appendix B

Functional analysis

This appendix provides some basic notions and results in functional analysis required in the main text. It is intended as a revision for a reader already familiar with these concepts. For more details and proofs, we refer for example to the standard textbooks [195, 196, 41, 77].

B.1 Vector spaces

Definition B.1. Let $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. A **vector space (over $\mathbb{K}$)** is a set X such that the following holds:

(i) *Properties of addition:* For every x, $y \in X$ there exists $x + y \in X$ such that for all $z \in X$

$$x + y = y + x \quad \text{and} \quad x + (y + z) = (x + y) + z.$$

Moreover, there exists a unique element $0 \in X$ such that $x + 0 = x$ for all $x \in X$ and for each $x \in X$ there exists a unique $-x \in X$ such that $x + (-x) = 0$.

(ii) *Properties of scalar multiplication:* There exists a map $(\alpha, x) \mapsto \alpha x$ from $\mathbb{K} \times X$ to X called scalar multiplication. It satisfies $1x = x$ and $(\alpha\beta)x = \alpha(\beta x)$ for all $x \in X$.

We call the elements of a vector space **vectors**.

If the field is clear from context, we simply refer to X as a vector space. We will primarily consider the case $\mathbb{K} = \mathbb{R}$, and in this case we also say that X is a real vector space.

To introduce a notion of convergence on a vector space X, it needs to be equipped with a topology, see Definition A.2. A **topological vector space** is a vector space which is also a topological space, and in which addition and scalar multiplication are continuous maps. We next discuss the most important instances of topological vector spaces.

B.1.1 Metric spaces

An important class of topological vector spaces consists of vector spaces that are also metric spaces.

Definition B.2. For a set X, we call a map $d_X \colon X \times X \to \mathbb{R}_+$ a **metric**, if

(i) $0 \leq d_X(x, y) < \infty$ for all x, $y \in X$,

(ii) $d_X(x, y) = 0$ if and only if $x = y$,

(iii) $d_X(x, y) = d(y, x)$ for all x, $y \in X$,

(iv) $d_X(x, z) \leq d_X(x, y) + d_X(y, z)$ for all x, y, $z \in X$.

We call (X, d_X) a **metric space**.

In a metric space (X, d_X), we denote the **open ball with center** x **and radius** $r > 0$ by

$$B_r(x) := \{y \in X \,|\, d_X(x, y) < r\}. \tag{B.1.1}$$

Every metric space is naturally equipped with a topology: A set $A \subseteq X$ is open if and only if for every $x \in A$ exists $\varepsilon > 0$ such that $B_\varepsilon(x) \subseteq A$. Therefore every metric vector space is a topological vector space.

Definition B.3. A metric space (X, d_X) is called **complete**, if every Cauchy sequence with respect to d converges to an element in X.

For complete metric spaces, an immensely powerful tool is Baire's category theorem. To state it, we require the notion of density of sets. Let A, $B \subseteq X$ for a topological space X. Then A is **dense** in B if the closure of A, denoted by $\overline{A}$, satisfies $\overline{A} \supseteq B$.

Theorem B.4 (Baire's category theorem)**.** *Let X be a complete metric space. Then the intersection of every countable collection of dense open subsets of X dense in X.*

Theorem B.4 implies that if $X = \bigcup_{i=1}^{\infty} V_i$ for a sequence of sets V_i, then at least one of the V_i has to contain an open set. Indeed, assuming all V_i's have empty interior implies that $V_i^c = X \setminus V_i$ is dense for all $i \in \mathbb{N}$. By De Morgan's laws, it then holds that $\emptyset = \bigcap_{i=1}^{\infty} V_i^c$ which contradicts Theorem B.4.

B.1.2 Normed spaces

A norm is a way of assigning a length to a vector. A normed space is a vector space with a norm.

Definition B.5. Let X be a vector space over a field $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. A map $\|\cdot\|_X : X \to [0, \infty)$ is called a **norm** if the following hold for all x, $y \in X$ and all $\alpha \in \mathbb{K}$:

(i) **triangle inequality**: $\|x + y\|_X \leq \|x\|_X + \|y\|_X$,

(ii) **absolute homogeneity**: $\|\alpha x\|_X = |\alpha| \|x\|_X$,

(iii) **positive definiteness**: $\|x\|_X = 0$ if and only if $x = 0$.

We call $(X, \|\cdot\|_X)$ a **normed space** and omit $\|\cdot\|_X$ from the notation if it is clear from the context.

Every norm *induces a metric* d_X *and hence a topology* via $d_X(x, y) := \|x - y\|_X$. In particular, every normed vector space is a topological vector space with respect to this topology.

B.1.3 Banach spaces

Definition B.6. A normed vector space is called a **Banach space** if and only if it is complete.

Before presenting the main results on Banach spaces, we collect a couple of important examples.

- *Euclidean spaces:* Let $d \in \mathbb{N}$. Then $(\mathbb{R}^d, \|\cdot\|)$ is a Banach space.
- *Continuous functions:* Let $d \in \mathbb{N}$ and let $K \subseteq \mathbb{R}^d$ be compact. The set of continuous functions from K to $\mathbb{R}$ is denoted by $C(K)$. For α, $\beta \in \mathbb{R}$ and f, $g \in C(K)$, we define addition and scalar multiplication by $(\alpha f + \beta g)(\boldsymbol{x}) = \alpha f(\boldsymbol{x}) + \beta g(\boldsymbol{x})$ for all $\boldsymbol{x} \in K$. The vector space $C(K)$ equipped with the **supremum norm**

  $$\|f\|_\infty := \sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x})|,$$

  is a Banach space.
- *Lebesgue spaces:* Let $(\Omega, \mathfrak{A}, \mu)$ be a measure space and let $1 \leq p < \infty$. Then the **Lebesgue space** $L^p(\Omega, \mu)$ is defined as the vector space of all equivalence classes of measurable functions $f : \Omega \to \mathbb{R}$ that coincide μ-almost everywhere and satisfy

  $$\|f\|_{L^p(\Omega,\mu)} := \left(\int_\Omega |f(x)|^p d\mu(x) \right)^{1/p} < \infty. \tag{B.1.2}$$

  The integral is independent of the choice of representative of the equivalence class of f. Addition and scalar multiplication are defined pointwise as for $C(K)$. It then holds that $L^p(\Omega, \mu)$ is a Banach space. If Ω is a measurable subset of $\mathbb{R}^d$ for $d \in \mathbb{N}$, and μ is the Lebesgue measure, we typically omit μ from the notation and simply write $L^p(\Omega)$. If $\Omega = \mathbb{N}$ and the measure is the counting measure, we denote these spaces by $\ell^p(\mathbb{N})$ or simply ℓ^p.

The definition can be extended to complex or $\mathbb{R}^d$-valued functions. In the latter case the integrand in (B.1.2) is replaced by $\|f(x)\|^p$. We denote these spaces again by $L^p(\Omega, \mu)$ with the precise meaning being clear from context.

- *Essentially bounded functions:* Let $(\Omega, \mathfrak{A}, \mu)$ be a measure space. The L^p spaces can be extended to $p = \infty$ by defining the **L^∞-norm**

$$\|f\|_{L^\infty(\Omega,\mu)} := \inf\{C \geq 0 \,|\, \mu(\{|f| > C\}) = 0)\}.$$

This is indeed a norm on the space of equivalence classes of measurable functions from $\Omega \to \mathbb{R}$ that coincide μ-almost everywhere. Moreover, with this norm, $L^\infty(\Omega, \mu)$ is a Banach space. If $\Omega = \mathbb{N}$ and μ is the counting measure, we denote the resulting space by $\ell^\infty(\mathbb{N})$ or simply ℓ^∞. As in the case $p < \infty$, it is straightforward to extend the definition to complex or $\mathbb{R}^d$-valued functions, for which the same notation will be used.

We continue by introducing the concept of dual spaces.

Definition B.7. Let $(X, \|\cdot\|_X)$ be a normed vector space over $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$. Linear maps from $X \to \mathbb{K}$ are called **linear functionals**. The vector space of all continuous linear functionals on X is called the **(topological) dual space of** X and is denoted by X'.

Together with the natural addition and scalar multiplication (for all h, $g \in X'$, $\alpha \in \mathbb{K}$ and $x \in X$)

$$(h + g)(x) := h(x) + g(x) \quad \text{and} \quad (\alpha h)(x) := \alpha(h(x)),$$

X' is a vector space. We equip X' with the norm

$$\|f\|_{X'} := \sup_{\substack{x \in X \\ \|x\|_X = 1}} |f(x)|.$$

The space $(X', \|\cdot\|_{X'})$ is always a Banach space, even if $(X, \|\cdot\|_X)$ is not complete [196, Theorem 4.1].

The dual space can often be used to characterize the original Banach space. One way in which the dual space X' captures certain algebraic and geometric properties of the Banach space X is through the Hahn-Banach theorem. In this book, we use one specific variant of this theorem and its implication for the existence of dual bases, see for instance [196, Theorem 3.5].

Theorem B.8 (Geometric Hahn-Banach, subspace version)**.** *Let M be a subspace of a Banach space X and let $x_0 \in X$. If x_0 is not in the closure of M, then there exists $f \in X'$ such that $f(x_0) = 1$ and $f(x) = 0$ for every $x \in M$.*

An immediate consequence of Theorem B.8 that will be used throughout this book is the existence of a **dual basis**. Let X be a Banach space and let $(x_i)_{i \in \mathbb{N}} \subseteq X$ be such that for all $i \in \mathbb{N}$

$$x_i \notin \overline{\operatorname{span}\{x_j \,|\, j \in \mathbb{N},\ j \neq i\}}.$$

Then, for every $i \in \mathbb{N}$, there exists $f_i \in X'$ such that $f_i(x_j) = 0$ if $i \neq j$ and $f_i(x_i) = 1$.

B.1.4 Hilbert spaces

Often, we require more structure than that provided by normed spaces. An inner product offers additional tools to compare vectors by introducing notions of angle and orthogonality. For simplicity we restrict ourselves to real vector spaces in the following.

Definition B.9. Let X be a real vector space. A map $\langle \cdot, \cdot \rangle_X : X \times X \to \mathbb{R}$ is called an **inner product** on X if the following hold for all x, y, $z \in X$ and all α, $\beta \in \mathbb{R}$:

(i) **linearity**: $\langle \alpha x + \beta y, z \rangle_X = \alpha \langle x, z \rangle_X + \beta \langle y, z \rangle_X$,

(ii) **symmetry**: $\langle x, y \rangle_X = \langle y, x \rangle_X$,

(iii) **positive definiteness**: $\langle x, x \rangle_X > 0$ for all $x \neq 0$.

On inner product spaces the so-called Cauchy-Schwarz inequality holds.

Theorem B.10 (Cauchy-Schwarz inequality)**.** *Let X be a vector space with inner product $\langle \cdot, \cdot \rangle_X$. Then it holds for all x, $y \in X$*

$$|\langle x, y \rangle_X| \leq \sqrt{\langle x, x \rangle_X \langle y, y \rangle_X}.$$

Moreover, equality holds if and only if x and y are linearly dependent.

Proof. Let x, $y \in X$. If $y = 0$ then $\langle x, y \rangle_X = 0$ and thus the statement is trivial. Assume in the following $y \neq 0$, so that $\langle y, y \rangle_X > 0$. Using the linearity and symmetry properties it holds for all $\alpha \in \mathbb{R}$

$$0 \leq \langle x - \alpha y, x - \alpha y \rangle_X = \langle x, x \rangle_X - 2\alpha \langle x, y \rangle_X + \alpha^2 \langle y, y \rangle_X .$$

Letting $\alpha := \langle x, y \rangle_X / \langle y, y \rangle_X$ we get

$$0 \leq \langle x, x \rangle_X - 2\frac{\langle x, y \rangle_X^2}{\langle y, y \rangle_X} + \frac{\langle x, y \rangle_X^2}{\langle y, y \rangle_X} = \langle x, x \rangle_X - \frac{\langle x, y \rangle_X^2}{\langle y, y \rangle_X}.$$

Rearranging terms gives the claim. □

Every inner product $\langle \cdot, \cdot \rangle_X$ induces a norm via

$$\|x\|_X := \sqrt{\langle x, x \rangle} \qquad \text{for all } x \in X. \tag{B.1.3}$$

The properties of the inner product immediately yield the **polar identity**

$$\|x + y\|_X^2 = \|x\|_X^2 + 2\langle x, y \rangle_X + \|y\|_X^2. \tag{B.1.4}$$

The fact that (B.1.3) indeed defines a norm follows by an application of the Cauchy-Schwarz inequality to (B.1.4), which yields that $\| \cdot \|_X$ satisfies the triangle inequality. This gives rise to the definition of a Hilbert space.

Definition B.11. Let H be a real vector space with inner product $\langle \cdot, \cdot \rangle_H$. Then $(H, \langle \cdot, \cdot \rangle_H)$ is called a **Hilbert space** if and only if H is complete with respect to the norm $\| \cdot \|_H$ induced by the inner product.

A standard example of a Hilbert space is L^2: Let $(\Omega, \mathfrak{A}, \mu)$ be a measure space. Then

$$\langle f, g \rangle_{L^2(\Omega,\mu)} = \int_\Omega f(x) g(x) \, \mathrm{d}\mu(x) \qquad \text{for all } f, g \in L^2(\Omega, \mu),$$

defines an inner product on $L^2(\Omega, \mu)$ compatible with the $L^2(\Omega, \mu)$-norm.

In a Hilbert space, we can compare vectors not only via their distance, measured by the norm, but also by using the inner product, which corresponds to their relative orientation. This leads to the concept of orthogonality.

Definition B.12. Let $(H, \langle \cdot, \cdot \rangle_H)$ be a Hilbert space and let f, $g \in H$. We say that f and g are **orthogonal** if $\langle f, g \rangle_H = 0$, denoted by $f \perp g$. Moreover, for F, $G \subseteq H$ we write $F \perp G$ if $f \perp g$ for all $f \in F$, $g \in G$.

For orthogonal vectors, the polar identity immediately implies the Pythagorean theorem.

Theorem B.13 (Pythagorean theorem)**.** *Let $(H, \langle \cdot, \cdot \rangle_H)$ be a Hilbert space, $n \in \mathbb{N}$, and let $f_1, \ldots, f_n \in H$ be pairwise orthogonal vectors. Then,*

$$\left\| \sum_{i=1}^n f_i \right\|_H^2 = \sum_{i=1}^n \|f_i\|_H^2 .$$

A final property of Hilbert spaces that we encounter in this book is the existence of unique **projections** onto convex sets. For a proof, see for instance [195, Thm. 4.10].

Theorem B.14. *Let $(H, \langle \cdot, \cdot \rangle_H)$ be a Hilbert space and let $K \neq \emptyset$ be a closed convex subset of H. Then for all $h \in H$ exists a unique $k_0 \in K$ such that*

$$\|h - k_0\|_H = \inf\{\|h - k\|_H \,|\, k \in K\}.$$

B.2 Fourier transform

The Fourier transform is a powerful tool in analysis. It allows to represent functions as a superposition of frequencies.

Definition B.15. Let $d \in \mathbb{N}$. The **Fourier transform** of a function $f \in L^1(\mathbb{R}^d)$ is defined by

$$\mathcal{F}(f)(\boldsymbol{\omega}) := \hat{f}(\boldsymbol{\omega}) := \int_{\mathbb{R}^d} f(\boldsymbol{x}) e^{-2\pi i \boldsymbol{x}^\top \boldsymbol{\omega}} \, \mathrm{d}\boldsymbol{x} \qquad \text{for all } \boldsymbol{\omega} \in \mathbb{R}^d,$$

and the **inverse Fourier transform** by

$$\mathcal{F}^{-1}(f)(\boldsymbol{x}) := \check{f}(\boldsymbol{x}) := \hat{f}(-\boldsymbol{x}) = \int_{\mathbb{R}^d} f(\boldsymbol{\omega}) e^{2\pi i \boldsymbol{x}^\top \boldsymbol{\omega}} \, \mathrm{d}\boldsymbol{\omega} \qquad \text{for all } \boldsymbol{x} \in \mathbb{R}^d.$$

It is immediately clear from the definition, that $\|\hat{f}\|_{L^\infty(\mathbb{R}^d)} \leq \|f\|_{L^1(\mathbb{R}^d)}$. As a result, the operator $\mathcal{F}\colon f \mapsto \hat{f}$ is a bounded linear map from $L^1(\mathbb{R}^d)$ to $L^\infty(\mathbb{R}^d)$. We point out that $\hat{f}$ can take complex values and the definition is also meaningful for complex-valued functions f.

If $\hat{f} \in L^1(\mathbb{R}^d)$, then we can reverse the process of taking the Fourier transform by taking the inverse Fourier transform, see [195, Theorem 9.11].

Theorem B.16. *If f, $\hat{f} \in L^1(\mathbb{R}^d)$ then $\mathcal{F}^{-1}(\hat{f}) = f$ almost everywhere.*

Bibliography

[1] R. Alaifari, G. S. Alberti, and T. Gauksson. Adef: an iterative algorithm to construct adversarial deformations. *arXiv preprint arXiv:1804.07729*, 2018.

[2] Z. Allen-Zhu, Y. Li, and Y. Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[3] M. Anthony and P. L. Bartlett. *Neural network learning: theoretical foundations*. Cambridge University Press, Cambridge, 1999.

[4] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations*, 2018.

[5] S. Arora, S. S. Du, W. Hu, Z. Li, R. R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[6] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR, 2018.

[7] M. Baader, M. Mirman, and M. Vechev. Universal approximation with certified networks. *arXiv preprint arXiv:1909.13846*, 2019.

[8] S. Barocas, M. Hardt, and A. Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[9] A. R. Barron. Neural net approximation. In *Proc. 7th Yale workshop on adaptive and learning systems*, volume 1, pages 69–72, 1992.

[10] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inform. Theory*, 39(3):930–945, 1993.

[11] A. R. Barron and J. M. Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv preprint arXiv:1809.03090*, 2018.

[12] P. Bartlett. For valid generalization the size of the weights is more important than the size of the network. *Advances in neural information processing systems*, 9, 1996.

[13] G. Beliakov. Interpolation of lipschitz functions. *Journal of Computational and Applied Mathematics*, 196(1):20–44, 2006.

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

[15] M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In *International Conference on Machine Learning*, pages 541–549. PMLR, 2018.

[16] R. Bellman. On the theory of dynamic programming. *Proceedings of the national Academy of Sciences*, 38(8):716–719, 1952.

[17] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[18] J. Berner, P. Grohs, and A. Jentzen. Analysis of the generalization error: Empirical risk minimization over deep artificial neural networks overcomes the curse of dimensionality in the numerical approximation of black–scholes partial differential equations. *SIAM Journal on Mathematics of Data Science*, 2(3):631–657, 2020.

[19] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen. The modern mathematics of deep learning, 2021.

[20] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific Optimization and Computation Series. Athena Scientific, Belmont, MA, third edition, 2016.

[21] H. Bolcskei, P. Grohs, G. Kutyniok, and P. Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1):8–45, 2019.

[22] L. Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[23] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[24] O. Bousquet and A. Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

[25] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004.

[26] J. Braun and M. Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive Approximation*, 30(3):653–675, Dec 2009.

[27] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[28] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[29] O. Calin. *Deep learning architectures*. Springer, 2020.

[30] E. J. Candes. *Ridgelets: theory and applications*. Stanford University, 1998.

[31] C. Carathéodory. Über den variabilitätsbereich der fourier'schen konstanten von positiven harmonischen funktionen. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 32:193–217, 1911.

[32] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[33] S. M. Carroll and B. W. Dickinson. Construction of neural nets using the radon transform. *International 1989 Joint Conference on Neural Networks*, pages 607–611 vol.1, 1989.

[34] 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.

[35] M. Chen, H. Jiang, W. Liao, and T. Zhao. Efficient approximation of deep relu networks for functions on low dimensional manifolds. *Advances in neural information processing systems*, 32, 2019.

[36] L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[37] Y. Cho and L. Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[38] F. Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[39] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR, 2015.

[40] C. K. Chui and H. N. Mhaskar. Deep nets for local manifold learning. *Frontiers in Applied Mathematics and Statistics*, 4:12, 2018.

[41] J. B. Conway. *A course in functional analysis*, volume 96. Springer, 2019.

[42] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 1 edition, 2000.

[43] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

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

[45] Y. N. 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. *Advances in neural information processing systems*, 27, 2014.

[46] A. G. de G. Matthews. Sample-then-optimize posterior sampling for bayesian linear models. 2017.

[47] A. G. de G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[48] T. De Ryck, S. Lanthaler, and S. Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021.

[49] A. Défossez, L. Bottou, F. R. Bach, and N. Usunier. A simple convergence proof of adam and adagrad. *Trans. Mach. Learn. Res.*, 2022, 2022.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[51] H. R. M. C. DeVore, R. Optimal nonlinear approximation. *Manuscripta mathematica*, 63(4):469–478, 1989.

[52] R. A. DeVore. Nonlinear approximation. *Acta numerica*, 7:51–150, 1998.

[53] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[54] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In *International conference on machine learning*, pages 1309–1318. PMLR, 2018.

[55] M. Du, F. Yang, N. Zou, and X. Hu. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems*, 36(4):25–34, 2021.

[56] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 09–15 Jun 2019.

[57] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[58] W. E and Q. Wang. Exponential convergence of the deep neural network approximation for analytic functions. *Sci. China Math.*, 61(10):1733–1740, 2018.

[59] K. Eckle and J. Schmidt-Hieber. A comparison of deep networks with relu activation function and linear spline-type methods. *Neural Networks*, 110:232–242, 2019.

[60] R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In V. Feldman, A. Rakhlin, and O. Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[61] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, and M. Vechev. Dl2: training and querying neural networks with logic. In *International Conference on Machine Learning*, pages 1931–1941. PMLR, 2019.

[62] 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.

[63] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[64] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[65] G. Garrigos and R. M. Gower. Handbook of convergence theorems for (stochastic) gradient methods, 2023.

[66] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2018.

[67] A. Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Sebastopol, CA, 2017.

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

[69] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological cybernetics*, 63(3):169–176, 1990.

[70] G. Goh. Why momentum really works. *Distill*, 2017.

[71] L. Gonon and C. Schwab. Deep relu network expression rates for option prices in high-dimensional, exponential lévy models. *Finance and Stochastics*, 25(4):615–657, 2021.

[72] I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[73] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[74] I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

[75] L.-A. Gottlieb, A. Kontorovich, and R. Krauthgamer. Efficient regression in metric spaces via approximate lipschitz extension. *IEEE Transactions on Information Theory*, 63(8):4838–4849, 2017.

[76] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik. SGD: General analysis and improved rates. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5200–5209. PMLR, 09–15 Jun 2019.

[77] K. Gröchenig. *Foundations of time-frequency analysis*. Springer Science & Business Media, 2013.

[78] P. Grohs and L. Herrmann. Deep neural network approximation for high-dimensional elliptic pdes with boundary conditions. *IMA Journal of Numerical Analysis*, 42(3):2055–2082, 2022.

[79] P. Grohs, F. Hornung, A. Jentzen, and P. Von Wurstemberger. *A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black–Scholes partial differential equations*, volume 284. American Mathematical Society, 2023.

[80] P. Grohs, F. Hornung, A. Jentzen, and P. von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of Black-Scholes partial differential equations. *Mem. Amer. Math. Soc.*, 284(1410):v+93, 2023.

[81] I. Gühring and M. Raslan. Approximation rates for neural networks with encodable weights in smoothness spaces. *Neural Networks*, 134:107–130, 2021.

[82] B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.

[83] T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, 2022.

[84] S. S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[85] J. He, L. Li, J. Xu, and C. Zheng. Relu deep neural networks and linear finite elements. *J. Comput. Math.*, 38(3):502–527, 2020.

[86] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*, 2015.

[87] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[88] M. Hein and M. Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. *Advances in neural information processing systems*, 30, 2017.

[89] H. Heuser. *Lehrbuch der Analysis. Teil 1*. Vieweg + Teubner, Wiesbaden, revised edition, 2009.

[90] G. Hinton. Divide the gradient by a running average of its recent magnitude. https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6e.mp4, 2012. Lecture 6e.

[91] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

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

[93] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

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

[95] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[96] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1(2):3, 2017.

[97] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE transactions on neural networks*, 9(1):224–229, 1998.

[98] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011.

[99] T. Huster, C.-Y. J. Chiang, and R. Chadha. Limitations of the lipschitz constant as a defense against adversarial examples. In *ECML PKDD 2018 Workshops: Nemesis 2018, UrbReas 2018, SoGood 2018, IWAISe 2018, and Green Data Mining 2018, Dublin, Ireland, September 10-14, 2018, Proceedings 18*, pages 16–29. Springer, 2019.

[100] M. Hutzenthaler, A. Jentzen, T. Kruse, and T. A. Nguyen. A proof that rectified deep neural networks overcome the curse of dimensionality in the numerical approximation of semilinear heat equations. *SN partial differential equations and applications*, 1(2):10, 2020.

[101] J. Håstad. *Computational limitations of small depth circuits*. PhD thesis, Massachusetts Institute of Technology, 1987. Ph.D. Thesis, Department of Mathematics.

[102] D. J. Im, M. Tao, and K. Branson. An empirical analysis of deep network loss surfaces. 2016.

[103] V. E. Ismailov. *Ridge functions and applications in neural networks*, volume 263. American Mathematical Society, 2021.

[104] V. E. Ismailov. A three layer neural network can represent any multivariate function. *Journal of Mathematical Analysis and Applications*, 523(1):127096, 2023.

[105] Y. Ito and K. Saito. Superposition of linearly independent functions and finite mappings by neural networks. *The Mathematical Scientist*, 21(1):27, 1996.

[106] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[107] A. Jentzen, B. Kuckuck, and P. von Wurstemberger. Mathematical introduction to deep learning: methods, implementations, and theory. *arXiv preprint arXiv:2310.20360*, 2023.

[108] A. Jentzen and A. Riekert. On the existence of global minima and convergence analyses for gradient descent methods in the training of deep neural networks. *arXiv preprint arXiv:2112.09684*, 2021.

[109] A. Jentzen, D. Salimova, and T. Welti. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *Commun. Math. Sci.*, 19(5):1167–1205, 2021.

[110] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[111] P. C. Kainen, V. Kurkova, and A. Vogt. Approximation by neural networks is not continuous. *Neurocomputing*, 29(1-3):47–56, 1999.

[112] P. C. Kainen, V. Kurkova, and A. Vogt. Continuity of approximation by neural networks in l p spaces. *Annals of Operations Research*, 101:143–147, 2001.

[113] P. C. Kainen, V. Kurkova, and A. Vogt. Best approximation by linear combinations of characteristic functions of half-spaces. *Journal of Approximation Theory*, 122(2):151–159, 2003.

[114] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 795–811, Cham, 2016. Springer International Publishing.

[115] C. Karner, V. Kazeev, and P. C. Petersen. Limitations of gradient descent due to numerical instability of backpropagation. *arXiv preprint arXiv:2210.00805*, 2022.

[116] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 2015.

[117] A. Klenke. *Wahrscheinlichkeitstheorie*. Springer, 2006.

[118] M. Kohler, A. Krzyżak, and S. Langer. Estimation of a function of low local dimensionality by deep neural networks. *IEEE transactions on information theory*, 68(6):4032–4042, 2022.

[119] M. Kohler and S. Langer. On the rate of convergence of fully connected deep neural network regression estimates. *The Annals of Statistics*, 49(4):2231–2249, 2021.

[120] A. N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk SSSR*, 114:953–956, 1957.

[121] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[122] G. Kutyniok, P. Petersen, M. Raslan, and R. Schneider. A theoretical analysis of deep neural networks and parametric pdes. *Constructive Approximation*, 55(1):73–125, 2022.

[123] V. Kůrková. Kolmogorov's theorem is relevant. *Neural Computation*, 3(4):617–622, 1991.

[124] V. Kůrková. Kolmogorov's theorem and multilayer neural networks. *Neural Networks*, 5(3):501–506, 1992.

[125] F. Laakmann and P. Petersen. Efficient approximation of solutions of parametric linear transport equations by relu dnns. *Advances in Computational Mathematics*, 47(1):11, 2021.

[126] G. Lan. *First-order and Stochastic Optimization Methods for Machine Learning*. Springer Series in the Data Sciences. Springer International Publishing, Cham, 1st ed. 2020. edition, 2020.

[127] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[128] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[129] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[130] J. Lee, J. Sohl-dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[131] J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[132] 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.

[133] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM J. Optim.*, 26(1):57–95, 2016.

[134] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[135] W. Li. Generalization error of minimum weighted norm and kernel interpolation. *SIAM Journal on Mathematics of Data Science*, 3(1):414–438, 2021.

[136] M. Longo, J. A. Opschoor, N. Disch, C. Schwab, and J. Zech. De rham compatible deep neural network fem. *Neural Networks*, 165:721–739, 2023.

[137] C. Ma, S. Wojtowytsch, L. Wu, et al. Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. *arXiv preprint arXiv:2009.10713*, 2020.

[138] C. Ma, L. Wu, et al. A priori estimates of the population risk for two-layer neural networks. *arXiv preprint arXiv:1810.06397*, 2018.

[139] S. Mahan, E. J. King, and A. Cloninger. Nonclosedness of sets of neural networks in sobolev spaces. *Neural Networks*, 137:85–96, 2021.

[140] V. Maiorov and A. Pinkus. Lower bounds for approximation by mlp neural networks. *Neurocomputing*, 25(1):81–91, 1999.

[141] Y. Marzouk, Z. Ren, S. Wang, and J. Zech. Distribution learning via neural differential equations: a nonparametric statistical perspective. *Journal of Machine Learning Research (accepted)*, 2024.

[142] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.

[143] S. Mei and A. Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

[144] H. N. Mhaskar. Approximation properties of a multilayered feedforward artificial neural network. *Adv. Comput. Math.*, 1(1):61–80, 1993.

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

[146] H. N. Mhaskar and C. A. Micchelli. Approximation by superposition of sigmoidal and radial basis functions. *Adv. in Appl. Math.*, 13(3):350–373, 1992.

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

[148] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[149] C. Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[150] H. Montanelli and Q. Du. New error bounds for deep relu networks using sparse grids. *SIAM Journal on Mathematics of Data Science*, 1(1):78–92, 2019.

[151] H. Montanelli and H. Yang. Error bounds for deep relu networks using the kolmogorov–arnold superposition theorem. *Neural Networks*, 129:1–6, 2020.

[152] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[153] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[154] E. Moulines and F. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[155] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

[156] R. Nakada and M. Imaizumi. Adaptive approximation and generalization of deep neural network with intrinsic dimensionality. *Journal of Machine Learning Research*, 21(174):1–38, 2020.

[157] R. M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[158] Y. Nesterov. *Introductory lectures on convex optimization*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004. A basic course.

[159] Y. Nesterov. *Lectures on convex optimization*, volume 137 of *Springer Optimization and Its Applications*. Springer, Cham, second edition, 2018.

[160] Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269(3):543–547, 1983.

[161] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *Conference on learning theory*, pages 1376–1401. PMLR, 2015.

[162] R. H. Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA), volume III, pages 11–13. Piscataway, NJ: IEEE, 1987.

[163] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[164] E. Novak and H. Woźniakowski. Approximation of infinitely differentiable multivariate functions is intractable. *Journal of Complexity*, 25(4):398–404, 2009.

[165] B. O'Donoghue and E. Candès. Adaptive restart for accelerated gradient schemes. *Found. Comput. Math.*, 15(3):715–732, 2015.

[166] J. A. A. Opschoor, C. Schwab, and J. Zech. Exponential ReLU DNN expression of holomorphic maps in high dimension. *Constructive Approximation*, 2021.

[167] J. A. A. Opschoor, C. Schwab, and J. Zech. Deep learning in high dimension: ReLU neural network expression for Bayesian PDE inversion. In *Optimization and control for partial differential equations—uncertainty quantification, open and closed-loop control, and shape optimization*, volume 29 of *Radon Ser. Comput. Appl. Math.*, pages 419–462. De Gruyter, Berlin, 2022.

[168] P. Oswald. On the degree of nonlinear spline approximation in Besov-Sobolev spaces. *J. Approx. Theory*, 61(2):131–157, 1990.

[169] S. Ovchinnikov. Max-min representation of piecewise linear functions. *Beiträge Algebra Geom.*, 43(1):297–302, 2002.

[170] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[171] Y. C. Pati and P. S. Krishnaprasad. Analysis and synthesis of feedforward neural networks using discrete affine wavelet transformations. *IEEE Transactions on Neural Networks*, 4(1):73–85, 1993.

[172] J. Pennington and Y. Bahri. Geometry of neural network loss surfaces via random matrix theory. In *International Conference on Machine Learning*, pages 2798–2806. PMLR, 2017.

[173] P. Petersen, M. Raslan, and F. Voigtlaender. Topological properties of the set of functions generated by neural networks of fixed size. *Foundations of computational mathematics*, 21:375–444, 2021.

[174] P. Petersen and F. Voigtlaender. Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, 108:296–330, 2018.

[175] P. C. Petersen. *Neural Network Theory*. 2020. http://www.pc-petersen.eu/Neural_Network_Theory.pdf, Lecture notes.

[176] A. Pinkus. Approximation theory of the MLP model in neural networks. In *Acta numerica, 1999*, volume 8 of *Acta Numer.*, pages 143–195. Cambridge Univ. Press, Cambridge, 1999.

[177] G. Pisier. Remarques sur un résultat non publié de B. Maurey. *Séminaire Analyse fonctionnelle (dit "Maurey-Schwartz")*, 1980-1981.

[178] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *Int. J. Autom. Comput.*, 14(5):503–519, 2017.

[179] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428(6981):419–422, 2004.

[180] B. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[181] B. T. Polyak. *Introduction to optimization*. Translations Series in Mathematics and Engineering. Optimization Software, Inc., Publications Division, New York, 1987. Translated from the Russian, With a foreword by Dimitri P. Bertsekas.

[182] S. J. Prince. *Understanding Deep Learning*. MIT Press, 2023.

[183] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[184] M. H. Quynh Nguyen, Mahesh Chandra Mukkamala. On the loss landscape of a class of deep neural networks with no bad local valleys. In *International Conference on Learning Representations (ICLR)*, 2018.

[185] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, 06–11 Aug 2017.

[186] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[187] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[188] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006.

[189] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, Q. Le, and A. Kurakin. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4780–4789, 2019.

[190] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[191] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[192] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[193] W. Ruan, X. Yi, and X. Huang. Adversarial robustness of deep learning: Theory, algorithms, and applications. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 4866–4869, 2021.

[194] S. Ruder. An overview of gradient descent optimization algorithms, 2016.

[195] W. Rudin. *Real and complex analysis*. McGraw-Hill Book Co., New York, third edition, 1987.

[196] W. Rudin. *Functional analysis*. International Series in Pure and Applied Mathematics. McGraw-Hill, Inc., New York, second edition, 1991.

[197] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[198] T. D. Ryck and S. Mishra. Error analysis for deep neural network approximations of parametric hyperbolic conservation laws. *Mathematics of Computation*, 2023. Article electronically published on December 15, 2023.

[199] I. Safran and O. Shamir. Depth separation in relu networks for approximating smooth non-linear functions. *ArXiv*, abs/1610.09887, 2016.

[200] M. A. Sartori and P. J. Antsaklis. A simple method to derive bounds on the size and to train multilayer neural networks. *IEEE transactions on neural networks*, 2(4):467–471, 1991.

[201] R. Scheichl and J. Zech. Numerical methods for bayesian inverse problems, 2021. Lecture Notes.

[202] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[203] J. Schmidt-Hieber. Deep relu network approximation of functions on a manifold. *arXiv preprint arXiv:1908.00695*, 2019.

[204] J. Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. 2020.

[205] J. Schmidt-Hieber. The kolmogorov–arnold representation theorem revisited. *Neural Networks*, 137:119–126, 2021.

[206] B. Schölkopf and A. J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, 2002.

[207] L. Schumaker. *Spline Functions: Basic Theory*. Cambridge Mathematical Library. Cambridge University Press, 3 edition, 2007.

[208] C. Schwab and J. Zech. Deep learning in high dimension: neural network expression rates for generalized polynomial chaos expansions in UQ. *Anal. Appl. (Singap.)*, 17(1):19–55, 2019.

[209] C. Schwab and J. Zech. Deep learning in high dimension: neural network expression rates for analytic functions in $L^2(mathbbR^d, gamma_d)$. *SIAM/ASA J. Uncertain. Quantif.*, 11(1):199–234, 2023.

[210] T. Serra, C. Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks, 2018.

[211] U. Shaham, A. Cloninger, and R. R. Coifman. Provable approximation properties for deep neural networks. *Applied and Computational Harmonic Analysis*, 44(3):537–557, 2018.

[212] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press, 2014.

[213] J. W. Siegel and J. Xu. High-order approximation rates for shallow neural networks with cosine and reluk activation functions. *Applied and Computational Harmonic Analysis*, 58:1–26, 2022.

[214] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[215] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014.

[216] E. M. Stein. *Singular integrals and differentiability properties of functions*. Princeton Mathematical Series, No. 30. Princeton University Press, Princeton, N.J., 1970.

[217] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, New York, 2008.

[218] D. Stutz, M. Hein, and B. Schiele. Disentangling adversarial robustness and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6976–6987, 2019.

[219] A. Sukharev. Optimal method of constructing best uniform approximations for functions of a certain class. *USSR Computational Mathematics and Mathematical Physics*, 18(2):21–31, 1978.

[220] T. Sun, L. Qiao, and D. Li. Nonergodic complexity of proximal inertial gradient descents. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(10):4613–4626, 2021.

[221] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[222] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[223] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[224] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114, 2019.

[225] J. Tarela and M. Martínez. Region configurations for realizability of lattice piecewise-linear models. *Mathematical and Computer Modelling*, 30(11):17–27, 1999.

[226] J. M. Tarela, E. Alonso, and M. V. Martínez. A representation method for PWL functions oriented to parallel processing. *Math. Comput. Modelling*, 13(10):75–83, 1990.

[227] M. Telgarsky. Representation benefits of deep feedforward networks, 2015.

[228] M. Telgarsky. benefits of depth in neural networks. In V. Feldman, A. Rakhlin, and O. Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[229] M. Telgarsky. Deep learning theory lecture notes. https://mjt.cs.illinois.edu/dlt/, 2021. Version: 2021-10-27 v0.0-e7150f2d (alpha).

[230] V. M. Tikhomirov. ε-entropy and ε-capacity of sets in functional spaces. *Selected Works of AN Kolmogorov: Volume III: Information Theory and the Theory of Algorithms*, pages 86–170, 1993.

[231] S. Tu, S. Venkataraman, A. C. Wilson, A. Gittens, M. I. Jordan, and B. Recht. Breaking locality accelerates block Gauss-Seidel. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3482–3491. PMLR, 06–11 Aug 2017.

[232] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[233] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30. Springer, 2015.

[234] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[235] L. Venturi, A. S. Bandeira, and J. Bruna. Spurious valleys in one-hidden-layer neural network optimization landscapes. *Journal of Machine Learning Research*, 20:133, 2019.

[236] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.

[237] S. Wang and X. Sun. Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, 2005.

[238] Z. Wang, A. Albarghouthi, G. Prakriya, and S. Jha. Interval universal approximation for neural networks. *Proceedings of the ACM on Programming Languages*, 6(POPL):1–29, 2022.

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

[240] E. Weinan and S. Wojtowytsch. Representation formulas and pointwise properties for barron functions. *Calculus of Variations and Partial Differential Equations*, 61(2):46, 2022.

[241] A. C. Wilson, B. Recht, and M. I. Jordan. A lyapunov analysis of accelerated methods in optimization. *Journal of Machine Learning Research*, 22(113):1–34, 2021.

[242] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[243] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018.

[244] H. Xu and S. Mannor. Robustness and generalization. *Machine learning*, 86:391–423, 2012.

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

[246] D. Yarotsky and A. Zhevnerchuk. The phase diagram of approximation rates for deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13005–13015. Curran Associates, Inc., 2020.

[247] H. M. D. K. S. B. Yiding Jiang, Behnam Neyshabur. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations (ICLR)*, 2019.

[248] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.