

1

The Modern Mathematics of Deep Learning

Julius Berner, Philipp Grohs, Gitta Kutyniok and Philipp Petersen

Abstract: We describe the new field of the mathematical analysis of deep learning. This field emerged around a list of research questions that were not answered within the classical framework of learning theory. These questions concern: the outstanding generalization power of overparametrized neural networks, the role of depth in deep architectures, the apparent absence of the curse of dimensionality, a surprisingly successful optimization performance despite the non-convexity of the problem, understanding what features are learned, why deep architectures perform exceptionally well in physical problems, and which fine aspects of an architecture affect the behavior of a learning task in which way. We present an overview of modern approaches that yield partial answers to these questions. For selected approaches, we describe the main ideas in more detail.

1.1 Introduction

Deep learning has undoubtedly established itself as the outstanding machine learning technique of recent times. This dominant position has been claimed through a series of overwhelming successes in widely different application areas.

Perhaps the most famous application of deep learning, and certainly one of the first where these techniques became state-of-the-art, is image classification (LeCun et al., 1998; Krizhevsky et al., 2012; Szegedy et al., 2015; He et al., 2016). In this area, deep learning is nowadays the only method that is seriously considered. The prowess of deep learning classifiers goes so far that they often outperform humans in image-labelling tasks (He et al., 2015).

A second famous application area is the training of deep-learning-based agents to play board games or computer games, such as Atari games (Mnih et al., 2013). In this context, probably the most prominent achievement yet is the development of an algorithm that beat the best human player in the game of Go (Silver et al., 2016, 2017) – a feat that was previously unthinkable owing to the extreme complexity

2 *Berner et al. The Modern Mathematics of Deep Learning*

of this game. Moreover, even in multiplayer, team-based games with incomplete information, deep-learning-based agents nowadays outperform world-class human teams (Berner et al., 2019a; Vinyals et al., 2019).

In addition to playing games, deep learning has also led to impressive breakthroughs in the natural sciences. For example, it is used in the development of drugs (Ma et al., 2015), molecular dynamics (Faber et al., 2017), and in high-energy physics (Baldi et al., 2014). One of the most astounding recent breakthroughs in scientific applications is the development of a deep-learning-based predictor for the folding behavior of proteins (Senior et al., 2020). This predictor is the first method to match the accuracy of lab-based methods.

Finally, in the vast field of natural language processing, which includes the subtasks of understanding, summarizing, and generating text, impressive advances have been made based on deep learning. Here, we refer to Young et al. (2018) for an overview. One technique that has recently stood out is based on a so-called transformer neural network (Bahdanau et al., 2015; Vaswani et al., 2017). This network structure has given rise to the impressive GPT-3 model (Brown et al., 2020) which not only creates coherent and compelling texts but can also produce code, such as that for the layout of a webpage according to some instructions that a user inputs in plain English. Transformer neural networks have also been successfully employed in the field of symbolic mathematics (Saxton et al., 2018; Lample and Charton, 2019).

In this chapter, we present and discuss the mathematical foundations of the success story outlined above. More precisely, our goal is to outline the newly emerging field of *the mathematical analysis of deep learning*. To accurately describe this field, a necessary preparatory step is to sharpen our definition of the term deep learning. For the purposes of this chapter, we will use the term in the following narrow sense: *deep learning refers to techniques where deep neural networks¹ are trained with gradient-based methods*. This narrow definition helps to make this chapter more concise. We would like to stress, however, that we do not claim in any way that this is the *best* or the *right* definition of deep learning.

Having fixed a definition of deep learning, three questions arise concerning the aforementioned emerging field of mathematical analysis of deep learning. To what extent is a mathematical theory necessary? Is it truly a new field? What are the questions studied in this area?

Let us start by explaining the necessity of a theoretical analysis of the tools described above. From a scientific perspective, the primary reason why deep learning should be studied mathematically is simple curiosity. As we will see throughout this chapter, many practically observed phenomena in this context are not explained

¹ We will define the term *neural network* later but, for now, we can view it as a parametrized family of functions with a differentiable parametrization.

1.1 Introduction

3

theoretically. Moreover, theoretical insights and the development of a comprehensive theory often constitute the driving force underlying the development of new and improved methods. Prominent examples of mathematical theories with such an effect are the theory of fluid mechanics which is fundamental ingredient of the design of aircraft or cars, and the theory of information which affects and shapes all modern digital communication. In the words of Vladimir Vapnik²: “Nothing is more practical than a good theory,” (Vapnik, 2013, Preface). In addition to being interesting and practical, theoretical insight may also be necessary. Indeed, in many applications of machine learning, such as medical diagnosis, self-driving cars, and robotics, a significant level of control and predictability of deep learning methods is mandatory. Also, in services such as banking or insurance, the technology should be controllable in order to guarantee fair and explainable decisions.

Let us next address the claim that the field of mathematical analysis of deep learning is a newly emerging area. In fact, under the aforementioned definition of deep learning, there are two main ingredients of the technology: deep neural networks and gradient-based optimization. The first artificial neuron was already introduced in McCulloch and Pitts (1943). This neuron was not trained but instead used to explain a biological neuron. The first multi-layered network of such artificial neurons that was also trained can be found in Rosenblatt (1958). Since then, various neural network architectures have been developed. We will discuss these architectures in detail in the following sections. The second ingredient, gradient-based optimization, is made possible by the observation that, owing to the graph-based structure of neural networks, the gradient of an objective function with respect to the parameters of the neural network can be computed efficiently. This has been observed in various ways: see Kelley (1960); Dreyfus (1962); Linnainmaa (1970); Rumelhart et al. (1986). Again, these techniques will be discussed in the upcoming sections. Since then, techniques have been improved and extended. As the rest of the chapter is spent reviewing these methods, we will keep the discussion of literature brief at this point. Instead, we refer to some overviews of the history of deep learning from various perspectives: LeCun et al. (2015); Schmidhuber (2015); Goodfellow et al. (2016); Higham and Higham (2019).

Given the fact that the two main ingredients of deep neural networks have been around for a long time, one might expect that a comprehensive mathematical theory would have been developed that describes why and when deep-learning-based methods will perform well or when they will fail. Statistical learning theory (Anthony and Bartlett, 1999; Vapnik, 1999; Cucker and Smale, 2002; Bousquet et al., 2003; Vapnik, 2013) describes multiple aspects of the performance of general learning methods and in particular deep learning. We will review this theory in the

² This claim can be found earlier in a non-mathematical context in the works of Kurt Lewin (1943).

context of deep learning in §1.1.2 below. Here, we focus on the classical, deep-learning-related results that we consider to be well known in the machine learning community. Nonetheless, the choice of these results is guaranteed to be subjective. We will find that this classical theory is too general to explain the performance of deep learning adequately. In this context, we will identify the following questions that appear to be difficult to answer within the classical framework of learning theory: *Why do trained deep neural networks not overfit on the training data despite the enormous power of the architecture? What is the advantage of deep compared to shallow architectures? Why do these methods seemingly not suffer from the curse of dimensionality? Why does the optimization routine often succeed in finding good solutions despite the non-convexity, nonlinearity, and often non-smoothness of the problem? Which aspects of an architecture affect the performance of the associated models and how? Which features of data are learned by deep architectures? Why do these methods perform as well as or better than specialized numerical tools in the natural sciences?*

The new field of the mathematical analysis of deep learning has emerged around questions like those listed above. In the remainder of this chapter, we will collect some of the main recent advances towards answering these questions. Because this field of the mathematical analysis of deep learning is incredibly active and new material is added at breathtaking speed, a brief survey of recent advances in this area is guaranteed to miss not only a couple of references but also many of the most essential ones. Therefore we do not strive for a complete overview but, instead, showcase several fundamental ideas on a mostly intuitive level. In this way, we hope to allow readers to familiarize themselves with some exciting concepts and provide a convenient entry-point for further studies.

1.1.1 Notation

We denote by \mathbb{N} the set of natural numbers, by \mathbb{Z} the set of integers, and by \mathbb{R} the field of real numbers. For $N \in \mathbb{N}$, we denote by $[N]$ the set $\{1, \dots, N\}$. For two functions $f, g: X \rightarrow [0, \infty)$, we write $f \lesssim g$ if there exists a universal constant c such that $f(x) \leq cg(x)$ for all $x \in X$. In a pseudometric space (X, d_X) , we define the ball of radius $r \in (0, \infty)$ around a point $x \in X$ by $B_r^{d_X}(x)$, or $B_r(x)$ if the pseudometric d_X is clear from the context. By $\|\cdot\|_p$, $p \in [1, \infty]$, we denote the ℓ^p -norm, and by $\langle \cdot, \cdot \rangle$ the Euclidean inner product of given vectors. By $\|\cdot\|_{\text{op}}$ we denote the operator norm induced by the Euclidean norm and by $\|\cdot\|_F$ the Frobenius norm of given matrices. For $p \in [1, \infty]$, $s \in [0, \infty)$, $d \in \mathbb{N}$, and $X \subset \mathbb{R}^d$, we denote by $W^{s,p}(X)$ the Sobolev–Slobodeckij space, which for $s = 0$ is just a Lebesgue space, i.e., $W^{0,p}(X) = L^p(X)$. For measurable spaces X and Y , we define $\mathcal{M}(X, Y)$ to be the set of measurable functions from X to Y . We denote by \hat{g} the

1.1 Introduction

5

Fourier transform³ of a tempered distribution g . For probabilistic statements, we will assume a suitable underlying probability space with probability measure \mathcal{I} . For an \mathcal{X} -valued random variable X , we denote by $\mathbb{E}[X]$ and $\mathbb{V}[X]$ its expectation and variance and by \mathcal{I}_X the image measure of X on \mathcal{X} , i.e., $\mathcal{I}_X(A) = \mathcal{I}(X \in A)$ for every measurable set $A \subset \mathcal{X}$. If possible, we use the corresponding lowercase letter to denote the realization $x \in \mathcal{X}$ of the random variable X for a given outcome. We write \mathbf{I}_d for the d -dimensional identity matrix and, for a set A , we write 1_A for the indicator function of A , i.e., $1_A(x) = 1$ if $x \in A$ and $1_A(x) = 0$ otherwise.

1.1.2 Foundations of Learning Theory

Before we describe recent developments in the mathematical analysis of deep learning methods, we will start by providing a concise overview of the classical mathematical and statistical theory underlying machine learning tasks and algorithms that, in their most general form, can be formulated as follows.

Definition 1.1 (Learning – informal). Let \mathcal{X} , \mathcal{Y} , and \mathcal{Z} be measurable spaces. In a learning task, one is given data in \mathcal{Z} and a loss function $\mathcal{L}: \mathcal{M}(\mathcal{X}, \mathcal{Y}) \times \mathcal{Z} \rightarrow \mathbb{R}$. The goal is to choose a hypothesis set $\mathcal{F} \subset \mathcal{M}(\mathcal{X}, \mathcal{Y})$ and to construct a learning algorithm, i.e., a mapping

$$\mathcal{A}: \bigcup_{m \in \mathbb{N}} \mathcal{Z}^m \rightarrow \mathcal{F},$$

that uses training data $s = (z^{(i)})_{i=1}^m \in \mathcal{Z}^m$ to find a model $f_s = \mathcal{A}(s) \in \mathcal{F}$ that performs well on the training data s and also generalizes to unseen data $z \in \mathcal{Z}$. Here, performance is measured via the loss function \mathcal{L} and the corresponding loss $\mathcal{L}(f_s, z)$ and, informally speaking, generalization means that the out-of-sample performance of f_s at z behaves similarly to the in-sample performance on s .

Definition 1.1 is deliberately vague on how to measure generalization performance. Later, we will often study the *expected* out-of-sample performance. To talk about expected performance, a data distribution needs to be specified. We will revisit this point in Assumption 1.10 and Definition 1.11.

For simplicity, we focus on one-dimensional supervised prediction tasks with input features in Euclidean space, as defined in the following.

Definition 1.2 (Prediction task). In a prediction task, we have that $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$, i.e., we are given training data $s = ((x^{(i)}, y^{(i)}))_{i=1}^m$ that consist of input features $x^{(i)} \in \mathcal{X}$ and corresponding labels $y^{(i)} \in \mathcal{Y}$. For one-dimensional regression tasks with $\mathcal{Y} \subset \mathbb{R}$, we consider the quadratic loss $\mathcal{L}(f, (x, y)) = (f(x) - y)^2$ and, for binary

³ Respecting common notation, we will also use the hat symbol to denote the minimizer of the empirical risk \widehat{f}_s in Definition 1.8 but this clash of notation does not involve any ambiguity.

classification tasks with $\mathcal{Y} = \{-1, 1\}$, we consider the 0–1 loss $\mathcal{L}(f, (x, y)) = 1_{(-\infty, 0)}(yf(x))$. We assume that our input features are in Euclidean space, i.e., $\mathcal{X} \subset \mathbb{R}^d$ with input dimension $d \in \mathbb{N}$.

In a prediction task, we aim for a model $f_s: \mathcal{X} \rightarrow \mathcal{Y}$, such that, for unseen pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, $f_s(x)$ is a good prediction of the true label y . However, note that large parts of the presented theory can be applied to more general settings.

Remark 1.3 (Learning tasks). Apart from straightforward extensions to multi-dimensional prediction tasks and other loss functions, we want to mention that unsupervised and semi-supervised learning tasks are often treated as prediction tasks. More precisely, one transforms unlabeled training data $z^{(i)}$ into features $x^{(i)} = T_1(z^{(i)}) \in \mathcal{X}$ and labels $y^{(i)} = T_2(z^{(i)}) \in \mathcal{Y}$ using suitable transformations $T_1: \mathcal{Z} \rightarrow \mathcal{X}$, $T_2: \mathcal{Z} \rightarrow \mathcal{Y}$. In doing so, one asks for a model f_s approximating the transformation $T_2 \circ T_1^{-1}: \mathcal{X} \rightarrow \mathcal{Y}$ which is, for example, made in order to learn feature representations or invariances.

Furthermore, one can consider density estimation tasks, where $\mathcal{X} = \mathcal{Z}$, $\mathcal{Y} := [0, \infty]$, and \mathcal{F} consists of probability densities with respect to some σ -finite reference measure μ on \mathcal{Z} . One then aims for a probability density f_s that approximates the density of the unseen data z with respect to μ . One can perform $L^2(\mu)$ -approximation based on the discretization $\mathcal{L}(f, z) = -2f(z) + \|f\|_{L^2(\mu)}^2$ or maximum likelihood estimation based on the surprisal $\mathcal{L}(f, z) = -\log(f(z))$.

In deep learning the hypothesis set \mathcal{F} consists of *realizations of neural networks* $\Phi_a(\cdot, \theta)$, $\theta \in \mathcal{P}$, with a given *architecture* a and *parameter set* \mathcal{P} . In practice, one uses the term neural network for a range of functions that can be represented by directed acyclic graphs, where the vertices correspond to elementary almost everywhere differentiable functions parametrizable by $\theta \in \mathcal{P}$ and the edges symbolize compositions of these functions. In §1.6, we will review some frequently used architectures; in the other sections, however, we will mostly focus on *fully connected feed-forward* (FC) neural networks as defined below.

Definition 1.4 (FC neural network). A fully connected feed-forward neural network is given by its architecture $a = (N, \varrho)$, where $L \in \mathbb{N}$, $N \in \mathbb{N}^{L+1}$, and $\varrho: \mathbb{R} \rightarrow \mathbb{R}$. We refer to ϱ as the activation function, to L as the number of layers, and to N_0 , N_L , and N_ℓ , $\ell \in [L-1]$, as the number of neurons in the input, output, and ℓ th hidden layer, respectively. We denote the number of parameters by

$$P(N) := \sum_{\ell=1}^L N_\ell N_{\ell-1} + N_\ell$$

and define the corresponding realization function $\Phi_a: \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \rightarrow \mathbb{R}^{N_L}$, which

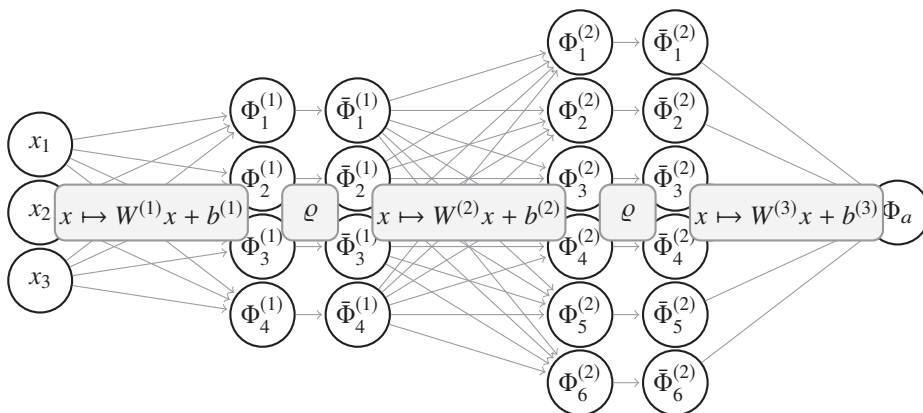


Figure 1.1 Graph (pale gray) and (pre-)activations of the neurons (white) of a deep fully connected feed-forward neural network $\Phi_a : \mathbb{R}^3 \times \mathbb{R}^{53} \mapsto \mathbb{R}$ with architecture $a = ((3, 4, 6, 1), \varrho)$ and parameters $\theta = ((W^{(\ell)}, b^{(\ell)})_{\ell=1}^3$.

satisfies, for every input $x \in \mathbb{R}^{N_0}$ and parameters

$$\theta = (\theta^{(\ell)})_{\ell=1}^L = ((W^{(\ell)}, b^{(\ell)}))_{\ell=1}^L \in \bigtimes_{\ell=1}^L (\mathbb{R}^{N_\ell \times N_{\ell-1}} \times \mathbb{R}^{N_\ell}) \cong \mathbb{R}^{P(N)},$$

that $\Phi_a(x, \theta) = \Phi^{(L)}(x, \theta)$, where

$$\begin{aligned} \Phi^{(1)}(x, \theta) &= W^{(1)}x + b^{(1)}, \\ \bar{\Phi}^{(\ell)}(x, \theta) &= \varrho(\Phi^{(\ell)}(x, \theta)), \quad \ell \in [L-1], \quad \text{and} \\ \Phi^{(\ell+1)}(x, \theta) &= W^{(\ell+1)}\bar{\Phi}^{(\ell)}(x, \theta) + b^{(\ell+1)}, \quad \ell \in [L-1], \end{aligned} \quad (1.1)$$

and ϱ is applied componentwise. We refer to $W^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and $b^{(\ell)} \in \mathbb{R}^{N_\ell}$ as the weight matrices and bias vectors, and to $\bar{\Phi}^{(\ell)}$ and $\Phi^{(\ell)}$ as the activations and pre-activations of the N_ℓ neurons in the ℓ th layer. The width and depth of the architecture are given by $\|N\|_\infty$ and L and we call the architecture deep if $L > 2$ and shallow if $L = 2$.

The underlying directed acyclic graph of FC networks is given by compositions of the affine linear maps $x \mapsto W^{(\ell)}x + b^{(\ell)}$, $\ell \in [L]$, with the activation function ϱ intertwined; see Figure 1.1. Typical activation functions used in practice are variants of the *rectified linear unit* (ReLU) given by $\varrho_R(x) := \max\{0, x\}$ and *sigmoidal functions* $\varrho \in C(\mathbb{R})$ satisfying $\varrho(x) \rightarrow 1$ for $x \rightarrow \infty$ and $\varrho(x) \rightarrow 0$ for $x \rightarrow -\infty$, such as the logistic function $\varrho_\sigma(x) := 1/(1 + e^{-x})$ (often referred to as *the sigmoid function*). See also Table 1.1 for a comprehensive list of widely used activation functions.

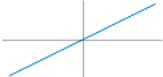

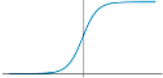

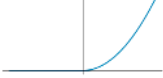
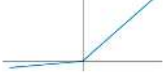
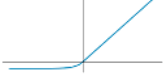



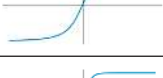
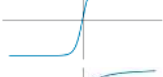

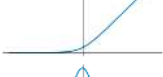
Name	Given as a function of $x \in \mathbb{R}$ by	Plot
linear	x	
Heaviside / step function	$1_{(0,\infty)}(x)$	
logistic / sigmoid	$\frac{1}{1+e^{-x}}$	
rectified linear unit (ReLU)	$\max\{0, x\}$	
power rectified linear unit	$\max\{0, x\}^k$ for $k \in \mathbb{N}$	
parametric ReLU (PReLU)	$\max\{ax, x\}$ for $a \geq 0, a \neq 1$	
exponential linear unit (ELU)	$x \cdot 1_{[0,\infty)}(x) + (e^x - 1) \cdot 1_{(-\infty,0)}(x)$	
softsign	$\frac{x}{1+ x }$	
inverse square root linear unit	$x \cdot 1_{[0,\infty)}(x) + \frac{x}{\sqrt{1+ax^2}} \cdot 1_{(-\infty,0)}(x)$ for $a > 0$	
inverse square root unit	$\frac{x}{\sqrt{1+ax^2}}$ for $a > 0$	
tanh	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	
arctan	$\arctan(x)$	
softplus	$\ln(1 + e^x)$	
Gaussian	$e^{-x^2/2}$	

Table 1.1 List of commonly used activation functions.

Remark 1.5 (Neural networks). If not further specified, we will use the term (neural) network, or the abbreviation NN, to refer to FC neural networks. Note that many of the architectures used in practice (see §1.6) can be written as special cases of Definition 1.4 where, for example, specific parameters are prescribed by constants or shared with other parameters. Furthermore, note that affine linear functions are NNs with depth $L = 1$. We will also consider biasless NNs given by linear mappings without bias vector, i.e., $b^{(\ell)} = 0$, $\ell \in [L]$. In particular, any NN can always be written without bias vectors by redefining

$$x \rightarrow \begin{bmatrix} x \\ 1 \end{bmatrix}; \quad (W^{(\ell)}, b^{(\ell)}) \rightarrow \begin{bmatrix} W^{(\ell)} & b^{(\ell)} \\ 0 & 1 \end{bmatrix}; \quad \ell \in [L-1]; \quad \text{and} \\ (W^{(L)}, b^{(L)}) \rightarrow \begin{bmatrix} W^{(L)} & b^{(L)} \end{bmatrix}.$$

To enhance readability we will often not specify the underlying architecture $a = (N, \varrho)$ or the parameters $\theta \in \mathbb{R}^{P(N)}$ but use the term NN to refer to the architecture as well as the realization functions $\Phi_a(\cdot, \theta): \mathbb{R}^{N_0} \rightarrow \mathbb{R}^{N_L}$ or $\Phi_a: \mathbb{R}^{N_0} \times \mathbb{R}^{P(N)} \rightarrow \mathbb{R}^{N_L}$. However, we want to emphasize that one cannot infer the underlying architecture or properties such as the magnitude of parameters solely from these functions, as the mapping $(a, \theta) \mapsto \Phi_a(\cdot, \theta)$ is highly non-injective. As an example, we can set $W^{(L)} = 0$, which implies $\Phi_a(\cdot, \theta) = b^{(L)}$ for all architectures $a = (N, \varrho)$ and all values of $(W^{(\ell)}, b^{(\ell)})_{\ell=1}^{L-1}$.

In view of our considered prediction tasks in Definition 1.2, this naturally leads to the following hypothesis sets of neural networks.

Definition 1.6 (Hypothesis sets of neural networks). Let $a = (N, \varrho)$ be a NN architecture with input dimension $N_0 = d$, output dimension $N_L = 1$, and measurable activation function ϱ . For regression tasks the corresponding hypothesis set is given by

$$\mathcal{F}_a = \{\Phi_a(\cdot, \theta): \theta \in \mathbb{R}^{P(N)}\}$$

and for classification tasks by

$$\mathcal{F}_{a, \text{sgn}} = \{\text{sgn}(\Phi_a(\cdot, \theta)): \theta \in \mathbb{R}^{P(N)}\}, \quad \text{where} \quad \text{sgn}(x) := \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0. \end{cases}$$

Note that we compose the output of the NN with the sign function in order to obtain functions mapping to $\mathcal{Y} = \{-1, 1\}$. This can be generalized to multi-dimensional classification tasks by replacing the sign by an argmax function. Given a hypothesis set, a popular learning algorithm is *empirical risk minimization* (ERM), which minimizes the average loss on the given training data, as described in the next two definitions.

10 *Berner et al. The Modern Mathematics of Deep Learning*

Definition 1.7 (Empirical risk). For training data $s = (z^{(i)})_{i=1}^m \in \mathcal{Z}^m$ and a function $f \in \mathcal{M}(\mathcal{X}, \mathcal{Y})$, we define the empirical risk by

$$\widehat{\mathcal{R}}_s(f) := \frac{1}{m} \sum_{i=1}^m \mathcal{L}(f, z^{(i)}). \quad (1.2)$$

Definition 1.8 (ERM learning algorithm). Given a hypothesis set \mathcal{F} , an empirical risk-minimization algorithm \mathcal{A}^{erm} chooses⁴ for training data $s \in \mathcal{Z}^m$ a minimizer $\widehat{f}_s \in \mathcal{F}$ of the empirical risk in \mathcal{F} , i.e.,

$$\mathcal{A}^{\text{erm}}(s) \in \operatorname{argmin}_{f \in \mathcal{F}} \widehat{\mathcal{R}}_s(f). \quad (1.3)$$

Remark 1.9 (Surrogate loss and regularization). Note that, for classification tasks, one needs to optimize over non-differentiable functions with discrete outputs in (1.3). For an NN hypothesis set $\mathcal{F}_{a, \text{sgn}}$ one typically uses the corresponding hypothesis set for regression tasks \mathcal{F}_a to find an approximate minimizer $\widehat{f}_s^{\text{surr}} \in \mathcal{F}_a$ of

$$\frac{1}{m} \sum_{i=1}^m \mathcal{L}^{\text{surr}}(f, z^{(i)}),$$

where $\mathcal{L}^{\text{surr}}: \mathcal{M}(\mathcal{X}, \mathbb{R}) \times \mathcal{Z} \rightarrow \mathbb{R}$ is a surrogate loss guaranteeing that $\operatorname{sgn}(\widehat{f}_s^{\text{surr}}) \in \operatorname{argmin}_{f \in \mathcal{F}_{a, \text{sgn}}} \widehat{\mathcal{R}}_s(f)$. A frequently used surrogate loss is the logistic loss,⁵ given by

$$\mathcal{L}^{\text{surr}}(f, z) = \log \left(1 + e^{-yf(x)} \right).$$

In various learning tasks one also adds regularization terms to the minimization problem in (1.3), such as penalties on the norm of the parameters of the NN, i.e.,

$$\min_{\theta \in \mathbb{R}^{P(N)}} \widehat{\mathcal{R}}_s(\Phi_a(\cdot, \theta)) + \alpha \|\theta\|_2^2,$$

where $\alpha \in (0, \infty)$ is a regularization parameter. Note that in this case the minimizer depends on the chosen parameters θ and not only on the realization function $\Phi_a(\cdot, \theta)$; see also Remark 1.5.

Coming back to our initial, informal description of learning in Definition 1.1, we have now outlined potential learning tasks in Definition 1.2, NN hypothesis sets in Definition 1.6, a metric for the in-sample performance in Definition 1.7, and a

⁴ For simplicity, we assume that the minimum is attained; this is the case, for instance, if \mathcal{F} is a compact topological space on which $\widehat{\mathcal{R}}_s$ is continuous. Hypothesis sets of NNs $\mathcal{F}_{(N, \varrho)}$ constitute a compact space if, for example, one chooses a compact parameter set $\mathcal{P} \subset \mathbb{R}^{P(N)}$ and a continuous activation function ϱ . One could also work with approximate minimizers: see Anthony and Bartlett (1999).

⁵ This can be viewed as cross-entropy between the label y and the output of f composed with a logistic function ϱ_σ . In a multi-dimensional setting one can replace the logistic function with a softmax function.