

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Numerically Solving Kolmogorov Partial Differential Equations with Variable Coefficient Parameters by Means of Deep Learning“

verfasst von / submitted by

Markus Ferdinand Dablander, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien, 2019 / Vienna 2019

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

A 066 821

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Master's programme in Mathematics

Betreut von / Supervisor:

Univ.-Prof. Dr. Philipp Grohs

:

Abstract (Deutsch)

Diese Arbeit startet mit einer rigorosen Einführung in die Theorie des überwachten statistischen Lernens auf Basis endlicher Datensätze bestehend aus Realisierungen unabhängiger, identisch verteilter Paare von Zufallsvariablen. Im Anschluss wird die Hypothesenklasse der künstlichen neuronalen Netze in Kombination mit gradientenbasierten Trainingsverfahren als Rahmenmodell zur Lösung statistischer Lernprobleme vorgestellt und detailliert beschrieben.

Darauf aufbauend wird die Theorie des statistischen Lernens mittels künstlicher neuronaler Netze benutzt, um einen maschinellen Lernalgorithmus zur numerischen Lösung der Klasse linearer parabolischer partieller Kolmogorov Differentialgleichungen mit variablen, affin-linearen Koeffizienten zu entwickeln. Um das korrekte Konvergenzverhalten des Algorithmus mathematisch zu verifizieren, wird eine formale Theorie aufgebaut und ein Theorem vorgeschlagen, das die Benutzung des Algorithmus theoretisch rechtfertigt. Eine technische Beweisidee des Theorems, basierend auf der Feynman-Kac Formel, wird entwickelt, die den Beweis auf zwei (zu diesem Zeitpunkt unbewiesene) Lemmata reduziert, die unabhängig untersucht werden können.

Erfolgreiche numerische Experimente werden präsentiert, die die Funktionsfähigkeit des vorgestellten Algorithmus im Falle der Wärmeleitungsgleichung und eines europäischen Put-Options-Preis-Modells empirisch belegen. Eine Implementierung des Algorithmus in `Python/Tensorflow` wird zur Verfügung gestellt.

Abstract (English)

This work starts with a rigorous introduction to the theory of supervised statistical learning on the basis of finite data sets consisting of realizations of independent, identically distributed pairs of random variables. Subsequently, the hypothesis class of artificial neural networks, along with gradient-based training methods, is presented and described in detail as a mathematical framework for the solution of statistical learning problems.

The theory of statistical learning via artificial neural networks is then used to develop a machine learning algorithm for the numerical solution of the class of linear parabolic Kolmogorov partial differential equations with variable, affine-linear coefficients. In order to mathematically verify the correct convergence behavior of the algorithm, a formal theory is developed and a theorem is proposed, which theoretically justifies the use of the algorithm. A technical proof idea of the theorem, based on the Feynman-Kac formula, is suggested, which reduces the proof to two (currently unproven) lemmata, which can be studied independently.

Successful numerical experiments are presented, which empirically confirm the functionality of the proposed algorithm in the case of the heat equation and a European put option pricing model. An implementation of the algorithm in `Python` / `Tensorflow` is provided.

For my grandmother.

Contents

1	Introduction	1
2	Mathematical Foundations of Supervised Statistical Learning	4
2.1	Elements of Learning	4
2.2	The Mathematical Learning Problem	6
2.3	The Statistical Learning Problem	9
2.3.1	Hypothesis Classes and Target Functions	11
2.3.2	Empirical Error and Empirical Target Functions	13
2.3.3	Asymptotic Behaviour of Empirical Estimates	16
3	Artificial Neural Networks	22
3.1	Definition and Properties of Classical Artificial Neural Networks	22
3.2	Parameter Tuning via Backpropagation and Gradient Descent . .	30
3.3	Refinements of Vanilla Training Methods.	36
4	Solving Kolmogorov Equations with Variable Coefficient Parameters via Statistical Learning	40
4.1	Goal and Setting	40
4.2	Kolmogorov Partial Differential Equations and Viscosity Solutions	42
4.3	Stochastic Differential Equations and their Connection to Kolmogorov PDEs via the Feynman-Kac Formula	50
4.4	Kolmogorov Equations with Variable Coefficient Parameters as a Statistical Learning Problem	58
5	Numerical Experiments	68
5.1	Algorithm	68
5.2	The 1-Dimensional Heat Equation	74
5.3	European Put Option Pricing	78
6	Conclusion	84
7	Appendix A	85
7.1	Source Codes	85

1 Introduction

Linear parabolic partial differential equations of the form

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0, \quad (1)$$

$$u(0, x) = \varphi(x),$$

are referred to as Kolmogorov PDEs. Throughout this project, the initial condition

$$\varphi : \mathbb{R}^d \rightarrow \mathbb{R} \quad (2)$$

is continuous and the coefficient maps

$$\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}, \quad (3)$$

$$\mu : \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad (4)$$

are affine-linear functions, which are implicitly determined by finite-dimensional, real parameters

$$\pi_\sigma \in D_{\Pi_\sigma} \subseteq (\mathbb{R}^{d \times d})^{d+1}, \quad \pi_\mu \in D_{\Pi_\mu} \subseteq \mathbb{R}^{d \times d} \times \mathbb{R}^d. \quad (5)$$

Equations of this type frequently appear in applications, such as physical modelling (heat flow) and financial engineering (option pricing); in almost all cases, they cannot be solved explicitly. The development, theoretical analysis and empirical investigation of new, computationally efficient methods for the numerical solution of Kolmogorov PDEs is therefore of natural interests for applied scientists.

Novel work by Berner, Grohs, Jentzen et. al. [9] [10] [11] has revealed, that it is possible to express the unique viscosity solution

$$[v, w]^d \ni x_0 \mapsto u^{(\pi_\sigma, \pi_\mu)}(T, x_0) \in \mathbb{R} \quad (6)$$

of Equation 1 on a compact hypercube $[v, w]^d$ at a fixed, predefined time $T \geq 0$ with fixed, predefined coefficient parameters $\pi_\sigma \in D_{\Pi_\sigma}$, and $\pi_\mu \in D_{\Pi_\mu}$ as the solution of an appropriately chosen mathematical learning problem, defined by a pair of random variables

$$(X_0, \varphi(S_T^{(X_0, \pi_\sigma, \pi_\mu)})) : \Omega \rightarrow [v, w]^d \times \mathbb{R}, \quad (7)$$

whereby

$$X_0 : \Omega \rightarrow [v, w]^d \quad (8)$$

is a uniformly distributed random variable and the random process

$$S^{(X_0, \pi_\sigma, \pi_\mu)} : \Omega \times [0, T] \rightarrow \mathbb{R}^d \quad (9)$$

is chosen in such a way, that, for each fixed $x_0 \in [v, w]^d$, the map

$$S^{(x_0, \pi_\sigma, \pi_\mu)} : \Omega \times [0, T] \rightarrow \mathbb{R}^d \quad (10)$$

is a solution of the associated stochastic differential equation

$$S_t = x_0 + \int_0^t \mu(S_s)ds + \int_0^t \sigma(S_s)dB_s, \quad t \in [0, T]. \quad (11)$$

This reformulation of Equation 1 as a learning problem relies on the Feynman-Kac formula,

$$u^{(\pi_\sigma, \pi_\mu)}(T, x_0) = \mathbb{E}[\varphi(S_T^{(x_0, \pi_\sigma, \pi_\mu)})], \quad (12)$$

which establishes a fundamental link between parabolic PDEs and stochastic differential equations. This opens up the door for the numerical approximation of global solutions $u^{(\pi_\sigma, \pi_\mu)}(T, \cdot)$ of Kolmogorov PDEs by means of statistical machine learning, using synthetic realisations of $(X_0, \varphi(S_T^{(X_0, \pi_\sigma, \pi_\mu)}))$ as learning data, which are simulated in silico on demand. In particular, theoretical results by Grohs, Hornung, Jentzen and Wurstemberger [10] have demonstrated the extraordinary ability of deep artificial neural networks to overcome the curse of dimensionality, when used in the above setting for the numerical solution of Kolmogorov PDEs on a global hypercube $[v, w]^d$ with a priori defined, fixed parameters π_σ, π_μ, T .

In this research project, we further explore the new paradigm of using deep neural networks for the numerical solution of Kolmogorov partial differential equations via statistical learning. More specifically, our research goal is to generalise the method of Berner, Grohs, Jentzen et. al. by interpreting all four quantities $x_0, \pi_\sigma, \pi_\mu, T$ as realisations of uniformly distributed random variables

$$X_0 : \Omega \rightarrow [v, w]^d, \quad (13)$$

$$\Pi_\sigma : \Omega \rightarrow D_{\Pi_\sigma}, \quad (14)$$

$$\Pi_\mu : \Omega \rightarrow D_{\Pi_\mu}, \quad (15)$$

$$\mathcal{T} : \Omega \rightarrow [0, T], \quad (16)$$

and subsequently use them as training data for a deep learning algorithm. This is in contrast to the above method, where only x_0 (interpreted as a realisation of X_0) is used as training data, while π_σ, π_μ and T remain fixed, predefined parameters. The aim of our proposed algorithm 5.2 is to approximate the solution of a whole class of Kolmogorov PDEs

$$[v, w]^d \times D_{\Pi_\sigma} \times D_{\Pi_\mu} \times [0, T] \ni (x_0, \pi_\sigma, \pi_\mu, t) \mapsto u^{(\pi_\sigma, \pi_\mu)}(t, x_0) \in \mathbb{R} \quad (17)$$

on a global space-time region $(t, x_0) \in [0, T] \times [v, w]^d$ with variable coefficient parameters $\pi_\sigma \in D_{\Pi_\sigma}$ and $\pi_\mu \in D_{\Pi_\mu}$ by training a single neural network model with simulated data. Again, our synthetic training data can be produced in silico and consists of realisations of a pair of random variables

$$\begin{aligned} & ((X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T}), \varphi(\bar{S}(\cdot, (X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T})))) : \\ & \Omega \rightarrow ([v, w]^d \times D_{\Pi_\sigma} \times D_{\Pi_\mu} \times [0, T]) \times \mathbb{R}, \end{aligned} \quad (18)$$

identified with an appropriately chosen mathematical learning problem, whereby we actively construct the function

$$\bar{S} : \Omega \times ([v, w]^d \times D_{\Pi_\sigma} \times D_{\Pi_\mu} \times [0, T]) \rightarrow \mathbb{R}^d \quad (19)$$

in such a way, that, for each fixed triple

$$(x_0, \pi_\sigma, \pi_\mu) \in [v, w]^d \times D_{\Pi_\sigma} \times D_{\Pi_\mu}, \quad (20)$$

the map

$$\Omega \times [0, T] \ni (\omega, t) \mapsto \bar{S}(\omega, (x_0, \pi_\sigma, \pi_\mu, t)) \in \mathbb{R}^d \quad (21)$$

is a solution process of the associated stochastic differential equation

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T]. \quad (22)$$

Chapter 2 and Chapter 3 provide essential preliminary knowledge for the understanding our research project. In Chapter 2, we give a rigorous introduction into the mathematical theory of supervised learning and the formal framework of empirical risk minimization, as discussed in the well-known work of Vapnik [4] and Cucker and Smale [5]. Chapter 3 gives an in-depth discussion of artificial neural networks as hypothesis classes, along with a description of classical neural network training methods, such as gradient descent via backpropagation, as well as more advanced training techniques.

In Chapter 4, we subsequently tackle our research goal from a theoretical angle. We formally construct the predictor variable $(X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T})$ and target variable $\varphi(\bar{S}(\cdot, (X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T})))$ of our learning problem 18 and develop a rigorous corresponding theory, culminating in the proposition of our main theorem 4.4, which links our learning problem 18 to the generalized Kolmogorov PDE solution map with variable coefficient parameters

$$[v, w]^d \times D_{\Pi_\sigma} \times D_{\Pi_\mu} \times [0, T] \ni (x_0, \pi_\sigma, \pi_\mu, t) \mapsto u^{(\pi_\sigma, \pi_\mu)}(t, x_0) \in \mathbb{R}. \quad (23)$$

We also suggest a mature (but as of yet incomplete) technical proof strategy for our main theorem 4.4, in which we too exploit the Feynman-Kac formula 12. In particular, we reduce the proof of Theorem 4.4 to a chain of arguments, which can be completed by independently proving two technical Lemmata, revolving around properties of conditional expected values of composite functions and special continuity requirements on $\bar{S}$ which remain to be shown.

Finally, in Chapter 5 we derive a practical deep learning algorithm 5.2 from our main theorem 4.4 for the numerical solution of Kolmogorov PDEs with variable coefficient parameters. We go on to present results of the successful application of our main algorithm 5.2 to the heat equation ($d = 1$) and a European put option pricing model ($d \in \{1, 10\}$), thus providing empirical evidence for the correctness of our method in several important cases of application. We include **Python**-code associated with our numerical experiments in Appendix 7.

Besides the usual canon of undergraduate mathematics knowledge, this thesis requires a good understanding of measure-theory as well as measure-theoretic probability theory. Strong familiarity with partial differential equations and stochastic differential equations is beneficial, but not strictly necessary.

2 Mathematical Foundations of Supervised Statistical Learning

2.1 Elements of Learning

Learning is one of the key processes associated with intelligent, adaptive behaviour and is studied in a variety of scientific disciplines, including psychology, neuroscience and certain subfields of mathematics and computer science, such as machine learning and artificial intelligence. While there is no precise, universally accepted definition of learning, it seems rather intuitive to suggest that learning has something to do with the acquisition of skills or knowledge via some kind of training process. In this training process, the learning system processes information in order to generate new insights or to improve its performance in certain tasks like prediction of future events or classification of distinct objects into different categories. The input information used in the training phase can either be taken from direct experience acquired through interactions with the environment (*reinforcement learning*), from previous unlabeled measurements, whose structure is to be investigated (*unsupervised learning*), or from associative knowledge provided by an external teacher (*supervised learning*). The areas of reinforcement learning, unsupervised learning, supervised learning cannot always be clearly distinguished from one another. Together they form the overarching technical discipline which is referred to as *machine learning*.

There is no one unified mathematical framework which can be used to formalize all aspects of learning. Reinforcement learning tasks are usually expressed as Markov decision processes [1] and unsupervised learning methods often take the form of clustering, dimensionality reduction and density estimation of clouds of data points in $\mathbb{R}^n$ [2]. In the case of supervised learning, the learner is formally confronted with a finite amount of i.i.d. realisations

$$(x_1, y_1), \dots, (x_m, y_m) \tag{24}$$

of a pair of random variables

$$(X, Y) \tag{25}$$

and is supposed to exploit this information to find an optimal mapping f from X to Y such that

$$f(X) \approx Y. \tag{26}$$

in some sense. This usually involves estimating certain quantities associated with the conditional expectation

$$\mathbb{E}[Y|X = x]. \tag{27}$$

Depending on whether Y is continuous or discrete, supervised learning problems naturally lead to either regression or classification tasks. A thorough treatment of a variety of supervised learning algorithms can be found in the well-known introductory book by C. M. Bishop [3].

In this thesis we experiment with the use of *supervised* learning methods for the numerical solution of certain classes of Kolmogorov equations and stochastic differential equations. As a technical framework for the machine learning aspect of this problem, we employ a formalism developed in statistical learning theory known as *empirical risk minimization* or, synonymously, *empirical error minimization*, which is discussed in depth in the following section. As our main sources, we choose the book of Vapnik [4] and the work presented by Cucker and Smale [5] as well as Smale and Poggio [6] in their papers about the mathematical foundations of supervised learning. The following pages deliver a concise, yet rigorous introduction into the key definitions and results of this branch of mathematical and statistical learning.

Before we start, let us introduce some notation as well as the basic formal setting in which we operate.

Setting 2.1. For the rest of Section 2,

- for any topological space $(U, \mathcal{O})$, let $\mathcal{B}(U)$ be the Borel σ -algebra on U ,
- let $(\Omega, \mathcal{G}, \mathbb{P})$ be a probability space,
- let $d, n, p, m \in \mathbb{N}_{\geq 1}$ be natural numbers,
- let $D \subset \mathbb{R}^d$ be a compact set, equipped with the Borel σ -algebra $\mathcal{B}(D)$,
- let $\mathbb{R}^n$ be equipped with the Borel σ -algebra $\mathcal{B}(\mathbb{R}^n)$, the standard scalar product

$$\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}, \quad \langle v, w \rangle = \sum_{i=1}^n v_i w_i, \quad (28)$$

and the associated Euclidean norm

$$\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}_{\geq 0}, \quad \|v\| = \langle v, v \rangle^{\frac{1}{2}}, \quad (29)$$

- let $X : \Omega \rightarrow D$ be a random vector,
- let $Y : \Omega \rightarrow \mathbb{R}^n$ be another random vector,
- let $Z := (X, Y) : \Omega \rightarrow D \times \mathbb{R}^n$ be the corresponding pair of random vectors,
- let $\mathbb{P}_X := \mathbb{P} \circ X^{-1}$, $\mathbb{P}_Y := \mathbb{P} \circ Y^{-1}$ and $\mathbb{P}_Z := \mathbb{P} \circ Z^{-1}$ be the pushforward measures of X , Y and Z induced on the measurable spaces $(D, \mathcal{B}(D))$, $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ and $(D \times \mathbb{R}^n, \mathcal{B}(D) \otimes \mathcal{B}(\mathbb{R}^n))$,
- let $\mathcal{L}^0(D, \mathbb{R}^n) := \{f : D \rightarrow \mathbb{R}^n \mid \forall A \in \mathcal{B}(\mathbb{R}^n) : f^{-1}(A) \in \mathcal{B}(D)\}$ be the space of $\mathcal{B}(D)/\mathcal{B}(\mathbb{R}^n)$ -measurable functions,
- let $\mathcal{L}^p(D, \mathbb{R}^n, \mathbb{P}_X) := \{f \in \mathcal{L}^0(D, \mathbb{R}^n) \mid \int_D \|f\|^p d\mathbb{P}_X < \infty\}$ be the space of $\mathcal{B}(D)/\mathcal{B}(\mathbb{R}^n)$ -measurable functions which are p -th power integrable with respect to the measure $\mathbb{P}_X$ on $(D, \mathcal{B}(D))$, and
- let $\mathcal{L}^0(\Omega, \mathbb{R}^n)$ as well as $\mathcal{L}^p(\Omega, \mathbb{R}^n, \mathbb{P})$, mutatis mutandis, be defined in analogy to $\mathcal{L}^0(D, \mathbb{R}^n)$ and $\mathcal{L}^p(D, \mathbb{R}^n, \mathbb{P}_X)$.

2.2 The Mathematical Learning Problem

We are interested in a functional relation $f : D \rightarrow \mathbb{R}^n$ such that $f(X)$ approximates Y . Given such a function, one can then estimate Y on the basis of X . In our search for a suitable f , we restrict ourselves to the space $\mathcal{L}^0(D, \mathbb{R}^n)$. The class of Borel-measurable functions is regular enough to allow useful definitions and theoretical investigations. On the other hand, $\mathcal{L}^0(D, \mathbb{R}^n)$ is large and sufficiently general for almost all practical purposes. Remember, for example, that all continuous functions from D to $\mathbb{R}^n$ are measurable.

Proposition 2.1. *All continuous functions $f : D \rightarrow \mathbb{R}^n$ are Borel-measurable.*

Proof. To see this, let $f : D \rightarrow \mathbb{R}^n$ be continuous and denote with $\mathcal{O}(V) := \{A \subseteq V \mid A \text{ is an open set}\}$ the collection of open sets of a topological space V . By definition, $\mathcal{B}(\mathbb{R}^n)$ is the σ -algebra generated by $\mathcal{O}(\mathbb{R}^n)$. A standard result from measure theory tells us, that it is sufficient to check measurability of functions only on generators, i.e. f is measurable if and only if $\forall U \in \mathcal{O}(\mathbb{R}^n) : f^{-1}(U) \in \mathcal{B}(D)$. Because f is continuous, we have (again by definition) $\forall U \in \mathcal{O}(\mathbb{R}^n) : f^{-1}(U) \in \mathcal{O}(D)$ and since $\mathcal{O}(D)$ is a generator of $\mathcal{B}(D)$, it follows that $\mathcal{O}(D) \subseteq \mathcal{B}(D)$. $\square$

In order to judge the quality of our approximation of Y via $f(X)$, we need to introduce some kind of error measure¹. A common choice is the expected least squares error.

Definition 2.1 (Error Function). *The map*

$$\mathcal{E}_Z : \mathcal{L}^0(D, \mathbb{R}^n) \rightarrow [0, \infty], \quad f \mapsto \mathbb{E}[\|f(X) - Y\|^2]. \quad (30)$$

is called "error function", "least squares error", "expected error" or simply "error".

Due to its simplicity and regularity, the least squares error serves as a reasonable building block in our theory of learning and we will stick with it. In theory, however, one could explore a variety of other error measures.

We now have introduced the concepts we need for a rigorous definition of our approximation problem.

Definition 2.2 (Mathematical Learning Problem). *Let $Z = (X, Y)$ be as defined in 2.1. The mathematical learning problem defined by Z consists of finding all measurable functions $f \in \mathcal{L}^0(D, \mathbb{R}^n)$ which minimize the least squares error $\mathcal{E}_Z$. More precisely, the goal of the mathematical learning problem is to fully characterize the set*

$$\begin{aligned} & \arg \min_{f \in \mathcal{L}^0(D, \mathbb{R}^n)} \mathcal{E}_Z(f) \\ & := \{f \in \mathcal{L}^0(D, \mathbb{R}^n) \mid \forall \tilde{f} \in \mathcal{L}^0(D, \mathbb{R}^n) : \mathcal{E}_Z(f) \leq \mathcal{E}_Z(\tilde{f})\}. \end{aligned} \quad (31)$$

¹In Vapnik's work [4] and some other theories of statistical learning, the error measure is referred to as the (expected) *risk*.

A natural starting point for our search for a solution to the learning problem is the conditional expected value of Y given X . If $Y \in \mathcal{L}^1(\Omega, \mathbb{R}^n, \mathbb{P})$, then the Radon-Nikodym theorem implies the existence ² of a $\mathbb{P}$ -unique (componentwise) conditional expectation

$$(\mathbb{E}[Y_j|X])_{j=1}^n := \mathbb{E}[Y|X] : \Omega \rightarrow \mathbb{R}^n. \quad (32)$$

The next proposition shows, that this conditional expectation can be expressed as a function of X .

Proposition 2.2. *Let $Y \in \mathcal{L}^1(\Omega, \mathbb{R}^n, \mathbb{P})$ (such that $\mathbb{E}[Y|X]$ exists). Then there also exists a measurable function*

$$\hat{f}_Z \in \mathcal{L}^0(D, \mathbb{R}^n) \quad (33)$$

such that

$$\hat{f}_Z(X) = \mathbb{E}[Y|X]. \quad (34)$$

Proof. To prove this, assume w.l.o.g. that $n = 1$. Furthermore, note that since $\mathbb{E}[Y|X] : \Omega \rightarrow \mathbb{R}$ is a $\sigma(X)/\mathcal{B}(\mathbb{R})$ -measurable function, there exists a sequence of $\sigma(X)/\mathcal{B}(\mathbb{R})$ -measurable simple functions $E_k : \Omega \rightarrow \mathbb{R}$ such that $E_k \rightarrow \mathbb{E}[Y|X]$ pointwise. It is possible, to write this sequence in the form $E_k = \sum_{i=1}^{m_k} a_{i,k} \mathbb{1}_{X^{-1}(B_{i,k})}$ for $a_{i,k} \in \mathbb{R}$ and $B_{i,k} \in \mathcal{B}(D)$. If we define $\hat{f}_k := \sum_{i=1}^{m_k} a_{i,k} \mathbb{1}_{B_{i,k}}$, we get $E_k = \hat{f}_k(X)$. Setting $\hat{f}_Z := \limsup f_k$ yields the result. $\square$

Definition 2.3 (Regression Function). *$\hat{f}_Z$ is called the regression function and is often denoted as*

$$\hat{f}_Z(x) := \mathbb{E}[Y|X = x]. \quad (35)$$

Intuitively, we can already suspect that $\hat{f}_Z$ might be a promising candidate in our attempts to minimize $\mathcal{E}_Z$ and solve the mathematical learning problem. But before we explore the relationship between $\hat{f}_Z$ and $\mathcal{E}_Z$, we show, that there is essentially only one regression function.

Proposition 2.3. *Let $Y \in \mathcal{L}^1(\Omega, \mathbb{R}^n, \mathbb{P})$, such that $\mathbb{E}[Y|X]$ and the regression function $\hat{f}_Z$ exist. Then $\hat{f}_Z$ is $\mathbb{P}_X$ -unique.*

Proof. Let $\mathbb{E}_1$ and $\mathbb{E}_2$ be two versions of $\mathbb{E}[Y|X]$ and let $\hat{f}_1$ and $\hat{f}_2$ be regression functions such that $\mathbb{E}_1 = \hat{f}_1(X)$ and $\mathbb{E}_2 = \hat{f}_2(X)$. Since $\hat{f}_1, \hat{f}_2 \in \mathcal{L}^0(D, \mathbb{R}^n)$, the set $\{x \in D \mid \hat{f}_1(x) \neq \hat{f}_2(x)\}$ belongs to $\mathcal{B}(D)$ and we have:

$$\begin{aligned} \mathbb{P}_X(\{x \in D \mid \hat{f}_1(x) \neq \hat{f}_2(x)\}) &= \\ \mathbb{P}(X^{-1}(\{x \in D \mid \hat{f}_1(x) \neq \hat{f}_2(x)\})) &= \\ \mathbb{P}(\{\omega \in \Omega \mid \hat{f}_1(X(\omega)) \neq \hat{f}_2(X(\omega))\}) &= \\ \mathbb{P}(\{\omega \in \Omega \mid \mathbb{E}_1(\omega) \neq \mathbb{E}_2(\omega)\}) &= 0. \end{aligned} \quad (36)$$

The last equality follows from the fact that $\mathbb{E}_1$ and $\mathbb{E}_2$ must be $\mathbb{P}$ -a.s. equal. $\square$

² For a detailed proof of existence, take a look at the book of Knill [7].

We can now conclude this subsection with an important theorem, which shows that, under mild assumptions on Y , the regression function indeed solves the learning problem.

Theorem 2.1 (The Regression Function Minimizes the Error). *Assume that Y has finite expectation and finite variance, i.e. $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$. Moreover, let $\hat{f}_Z$ be a version of the regression function associated with X and Y and let $f \in \mathcal{L}^0(D, \mathbb{R}^n)$ be an arbitrary measurable function.*

If $f \notin \mathcal{L}^2(D, \mathbb{R}^n, \mathbb{P}_X)$, we have

$$\mathcal{E}_Z(f) = \infty. \quad (37)$$

If $f \in \mathcal{L}^2(D, \mathbb{R}^n, \mathbb{P}_X)$, then

$$\mathcal{E}_Z(f) = \mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2] + \mathcal{E}_Z(\hat{f}_Z) = \int_D \|f - \hat{f}_Z\|^2 d\mathbb{P}_X + \mathcal{E}_Z(\hat{f}_Z) < \infty. \quad (38)$$

Proof. First, let $f \in \mathcal{L}^0(D, \mathbb{R}^n) \setminus \mathcal{L}^2(D, \mathbb{R}^n, \mathbb{P}_X)$. By using a change of variables³, we see that $f(X) \in \mathcal{L}^0(\Omega, \mathbb{R}^n) \setminus \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$:

$$\mathbb{E}[\|f(X)\|^2] = \int_{\Omega} \|f(X)\|^2 d\mathbb{P} = \int_D \|f\|^2 d\mathbb{P}_X = \infty. \quad (39)$$

Now, using the fact that $\forall a, b \in \mathbb{R} : a^2 + b^2 \geq 2ab$, we obtain

$$\begin{aligned} \infty &= \mathbb{E}[\|f(X)\|^2] \leq \mathbb{E}[(\|f(X) - Y\| + \|Y\|)^2] \\ &\leq \mathbb{E}[2(\|f(X) - Y\|^2 + \|Y\|^2)] \\ &= 2(\mathcal{E}_Z(f) + \mathbb{E}[\|Y\|^2]). \end{aligned} \quad (40)$$

Since $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$, the term $\mathbb{E}[\|Y\|^2]$ must be finite. This shows that $\mathcal{E}_Z(f) = \infty$, which proves the first part of the theorem.

We continue by assuming that $f \in \mathcal{L}^2(D, \mathbb{R}^n, \mathbb{P}_X)$. This implies that $f(X) \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$. Moreover, note that the fact that $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$ implies that $\hat{f}_Z(X) \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$ as well⁴. We can proceed by investigating the (now finite) error of f :

$$\begin{aligned} \mathcal{E}_Z(f) &= \mathbb{E}[\|f(X) - Y\|^2] = \mathbb{E}[\|f(X) - \hat{f}_Z(X) + \hat{f}_Z(X) - Y\|^2] \\ &= \mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2 + \|\hat{f}_Z(X) - Y\|^2 + 2\langle f(X) - \hat{f}_Z(X), \hat{f}_Z(X) - Y \rangle] \\ &= \mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2] + \mathcal{E}_Z(\hat{f}_Z) + 2\mathbb{E}[\langle f(X) - \hat{f}_Z(X), \hat{f}_Z(X) - Y \rangle] \\ &= \int_D \|f - \hat{f}_Z\|^2 d\mathbb{P}_X + \mathcal{E}_Z(\hat{f}_Z) + 2\mathbb{E}[\langle f(X) - \hat{f}_Z(X), \hat{f}_Z(X) - Y \rangle]. \end{aligned} \quad (41)$$

³For the exact change of variables formula and its proof, see, for example, Theorem 1.6.9 in the book of Durrett [8]

⁴To show this, remember that $\hat{f}_Z(X) = \mathbb{E}[Y|X]$ and use conditional Jensen's inequality on $\mathbb{E}[\|\mathbb{E}[Y|X]\|^2]$.

It remains to show, that the term $\mathbb{E}[\langle f(X) - \hat{f}_Z(X), \hat{f}_Z(X) - Y \rangle]$ vanishes. To see this, remember two properties of the conditional expectation. First, if $Z : \Omega \rightarrow \mathbb{R}^n$ is $\sigma(X)/\mathcal{B}(\mathbb{R}^n)$ -measurable, then $\langle Z, \mathbb{E}[Y|X] \rangle = \mathbb{E}[\langle Z, Y \rangle | X]$. Second, it holds that $\mathbb{E}[\mathbb{E}[Y|X]] = \mathbb{E}[Y]$. Finally, recall that $\hat{f}_Z(X) = \mathbb{E}[Y|X]$ and observe that $f(X) - \hat{f}_Z(X)$ is $\sigma(X)/\mathcal{B}(\mathbb{R}^n)$ -measurable. With this in mind, we conclude that

$$\begin{aligned} & \mathbb{E}[\langle f(X) - \hat{f}_Z(X), \hat{f}_Z(X) - Y \rangle] \\ &= \mathbb{E}[\langle f(X) - \hat{f}_Z(X), \mathbb{E}[Y|X] \rangle] - \mathbb{E}[\langle f(X) - \hat{f}_Z(X), Y \rangle] \\ &= \mathbb{E}[\mathbb{E}[\langle f(X) - \hat{f}_Z(X), Y \rangle | X]] - \mathbb{E}[\langle f(X) - \hat{f}_Z(X), Y \rangle] \\ &= \mathbb{E}[\langle f(X) - \hat{f}_Z(X), Y \rangle] - \mathbb{E}[\langle f(X) - \hat{f}_Z(X), Y \rangle] = 0. \end{aligned} \tag{42}$$

□

Corollary 2.1 (Solution of the Mathematical Learning Problem). *Let $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$. Then it holds that*

$$\arg \min_{f \in \mathcal{L}^0(D, \mathbb{R}^n)} \mathcal{E}_Z(f) = \{f \in \mathcal{L}^0(D, \mathbb{R}^n) \mid f = \hat{f}_Z \text{ } \mathbb{P}_X\text{-a.s.}\}. \tag{43}$$

Proof. Immediate consequence of Theorem 2.1. □

2.3 The Statistical Learning Problem

In the last section, we solved the mathematical learning problem under the weak assumption that $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$. In theory this is great, since we now know what functions we are looking for. As a result, we can try to determine a $\mathbb{P}_X$ -unique version of the regression function $\hat{f}_Z$ in one way or another if we explicitly know the joint distribution $\mathbb{P}_Z$ of $Z = (X, Y)$ on the product space $(D \times \mathbb{R}^n, \mathcal{B}(D) \otimes \mathcal{B}(\mathbb{R}^n))$.

In reality, however, one is almost never equipped with such a luxurious amount of knowledge. Rather than being given the exact distribution $\mathbb{P}_Z$, in practical research settings we can usually only hope to observe a finite amount of empirical realisations of $Z = (X, Y)$. These observations are our only source of information about the relationship between X and Y , and on the basis of these we must try to estimate the regression function. This leads us to a new type of problem.

Definition 2.4 (Statistical Learning Problem). *Let $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$ and let*

$$(z_i)_{i=1}^m = (z_1, \dots, z_m) := z^{(m)} \in (D \times \mathbb{R}^n)^m, \tag{44}$$

be the empirical realisations of m independent and identically distributed (i.i.d) copies $Z_i := (X_i, Y_i)$ of $Z = (X, Y)$, defined on Ω :

$$(Z_i)_{i=1}^m = (Z_1, \dots, Z_m) := Z^{(m)} : \Omega \rightarrow (D \times \mathbb{R}^n)^m. \tag{45}$$

This means, that there exists an ω , drawn from the probability space $(\Omega, \mathcal{G}, \mathbb{P})$, such that $Z^{(m)}(\omega) = z^{(m)}$.

The statistical learning problem asks for a way to use the information encapsulated in the realisation $z^{(m)}$ of $Z^{(m)}$ to find functions $f \in \mathcal{L}^0(D, \mathbb{R}^n)$ which approximate the regression function $\hat{f}_Z$ with high probability, i.e. for a method to identify functions $f \in \mathcal{L}^0(D, \mathbb{R}^n)$ on the basis of $z^{(m)}$ such that

$$\mathbb{E}[\|f(X) - \hat{f}_Z(X)\|^2] \quad (46)$$

is small with high probability. Due to Theorem 2.1, this is equivalent to finding functions $f \in \mathcal{L}^0(D, \mathbb{R}^n)$ such that the error

$$\mathcal{E}_Z(f) = \mathbb{E}[\|f(X) - Y\|^2] \quad (47)$$

is small with high probability.

The statistical learning problem can be seen as a probabilistic version of the mathematical learning problem, where uncertainty is created by our limited knowledge about the relationship between X and Y . The rest of this chapter is dedicated towards the description of a conceptual solution strategy.

Setting 2.2. In addition to Basic Setting 2.1, we now assume the requirements described in Definition 2.4 for the rest of Section 2, i.e. we assume that

- $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$,
- $Z^{(m)} = (Z_1, \dots, Z_m)$ consists of m i.i.d. copies of Z and
- $z^{(m)} = (z_1, \dots, z_m) := (Z_1(\omega), \dots, Z_m(\omega)) = Z^{(m)}(\omega)$ denotes a realisation of $Z^{(m)}$, where ω was drawn from $(\Omega, \mathcal{G}, \mathbb{P})$.

Remark 2.1. Note, that it is always possible to assume w.l.o.g. that Ω admits the construction of $Z_1, \dots, Z_n$. One can always take $(\Omega, \mathcal{G}, \mathbb{P})$, as defined in the beginning of section 2.2, and use it to construct the corresponding canonical m -fold product measure space

$$(\Omega^m, \otimes_{i=1}^m \mathcal{G}, \otimes_{i=1}^m \mathbb{P}). \quad (48)$$

One can then set

$$Z_i := Z(\pi_i) : \Omega^m \rightarrow D \times \mathbb{R}^n, \quad (49)$$

where $\pi_i : \Omega^m \rightarrow \Omega$ denote the canonical projections. Redefining

$$(\Omega, \mathcal{G}, \mathbb{P}) := (\Omega^m, \otimes_{i=1}^m \mathcal{G}, \otimes_{i=1}^m \mathbb{P}) \quad (50)$$

and identifying Z with Z_1 completes the procedure.

Our conceptual plan to tackle the statistical learning problem goes as follows: We start by defining a sufficiently well-behaved *hypothesis class*

$$\mathcal{H} \subseteq \mathcal{L}^0(D, \mathbb{R}^n), \quad (51)$$

of functions. We would then ideally like to find a *target function*

$$\hat{f}_{Z,\mathcal{H}} \in \arg \min_{f \in \mathcal{H}} \mathcal{E}_Z(f) \quad (52)$$

which minimizes $\mathcal{E}_Z$ over $\mathcal{H}$ and, in this sense, represents the "best possible hypothesis", which we want to identify. But here we are running into a problem. Since we do not know the exact distribution of Z , we cannot compute any explicit value for $\mathcal{E}_Z$. To work our way around this, we use our data $Z^{(m)}$ to replace the true error $\mathcal{E}_Z$ by a computable estimate, which we call the *empirical error*:

$$\tilde{\mathcal{E}}_{Z^{(m)}} : \Omega \times \mathcal{H} \rightarrow [0, \infty). \quad (53)$$

The empirical error is a random variable, which depends on our data $Z^{(m)}$ and is constructed as to approximate the real error $\mathcal{E}_Z$ reasonably well with high probability if the amount of data m is big. As a last step, we can minimize our given realisation

$$\tilde{\mathcal{E}}_{z^{(m)}} : \mathcal{H} \rightarrow [0, \infty) \quad (54)$$

of $\tilde{\mathcal{E}}_{Z^{(m)}}$ over $\mathcal{H}$ in order to "learn" an *empirical target function*

$$\tilde{f}_{z^{(m)},\mathcal{H}} \in \arg \min_{f \in \mathcal{H}} \mathcal{E}_{z^{(m)}}(f). \quad (55)$$

The empirical target function $\tilde{f}_{z^{(m)},\mathcal{H}}$ can be seen as the realisation of a random variable $\tilde{f}_{Z^{(m)},\mathcal{H}}$ which makes the (non-negative) difference between the errors

$$\mathcal{E}_Z(\tilde{f}_{Z^{(m)},\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_{Z,\mathcal{H}}) \quad (56)$$

small with high probability if m is big.

2.3.1 Hypothesis Classes and Target Functions

To turn the above thoughts into a rigorous theory, we start by discussing the concept of hypothesis classes. Many times, $\mathcal{H}$ is chosen to be a small subset of $\mathcal{L}^0(D, \mathbb{R}^n)$ which can be parametrized by a finite amount of variables. This restriction to a set of functions much narrower than $\mathcal{L}^0(D, \mathbb{R}^n)$ is done for at least two reasons. First of all, the mere size of $\mathcal{L}^0(D, \mathbb{R}^n)$ makes the design of optimization algorithms over this space practically impossible. In order to achieve computational feasibility, one has to severely constrain the search space to a sufficiently regular class of functions (such as the class of artificial neural networks, which we discuss in detail in the next section). The second reason to choose a restricted hypothesis class consists in the fact that it offers the opportunity to impose an advantageous a priori structure on the learning process using intuition or previously obtained knowledge about the nature of the problem at hand.

In this thesis, we require the hypothesis class to be (at least) a compact subset of the vector space of continuous functions. This is in line with the work of Cucker and Smale [5]. It is also in alignment with the work of Berner, Grohs, Jentzen et. al. [9] [10] [11], on whose work we refer to in Chapter 4.

Definition 2.5 (Hypothesis Class). *Let*

$$\mathcal{C}(D, \mathbb{R}^n) := \{f : D \rightarrow \mathbb{R}^n \mid \forall U \in \mathcal{O}(\mathbb{R}^n) : f^{-1}(U) \in \mathcal{O}(D)\} \quad (57)$$

be the Banach space of continuous functions from D to $\mathbb{R}^n$, equipped with the maximum norm

$$\|f\|_\infty := \max_{x \in D} \|f(x)\|. \quad (58)$$

If $\mathcal{H} \subseteq \mathcal{C}(D, \mathbb{R}^n)$ and $\mathcal{H}$ is non-empty and compact, then $\mathcal{H}$ is called a hypothesis class.

Remark 2.2. Since we assume that D is compact, the maximum norm is well-defined and finite on the space of continuous functions.

We remember, that since $\mathcal{H}$ is compact, it must also be closed and bounded. Also, compactness of our hypothesis class ensures that $\mathcal{E}_Z$ attains its minimum over $\mathcal{H}$. To see this, consider the following proposition.

Proposition 2.4 (Continuity of the Error). *The error map*

$$\mathcal{E}_Z : \mathcal{C}(D, \mathbb{R}^n) \rightarrow [0, \infty), \quad \mathcal{E}_Z(f) = \mathbb{E}[\|f(X) - Y\|^2], \quad (59)$$

is continuous (if restricted to the Banach space of continuous functions).

Proof. First note, that every continuous map f on compact D is necessarily bounded. Therefore, and because of our requirement that $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$, $\mathcal{E}_Z$ is always finite.

Now let f be a function and $(f_n)_{n \geq 1}$ be a sequence of functions, both in $\mathcal{C}(D, \mathbb{R}^n)$, such that

$$\|f_n - f\|_\infty \rightarrow 0. \quad (60)$$

To prove continuity of $\mathcal{E}_Z$, we show, that this implies that

$$|\mathcal{E}_Z(f_n) - \mathcal{E}_Z(f)| \rightarrow 0, \quad (61)$$

To see this, we use Cauchy-Schwarz and the inverse triangle inequality to calculate:

$$\begin{aligned} & |\mathcal{E}_Z(f_n) - \mathcal{E}_Z(f)| \\ &= |\mathbb{E}[\|f_n(X) - Y\|^2 - \|f(X) - Y\|^2]| \\ &= |\mathbb{E}[\|f_n(X)\|^2 - \|f(X)\|^2 + 2\langle f(X) - f_n(X), Y \rangle]| \\ &\leq \mathbb{E}[|\|f_n(X)\|^2 - \|f(X)\|^2| + 2|\langle f(X) - f_n(X), Y \rangle|] \\ &\leq \mathbb{E}[(\|f_n(X)\| - \|f(X)\|)(\|f_n(X)\| + \|f(X)\|) + 2\|f(X) - f_n(X)\|\|Y\|] \\ &\leq \mathbb{E}[\|f_n(X) - f(X)\|(\|f_n(X)\| + \|f(X)\|) + 2\|f(X) - f_n(X)\|\|Y\|] \\ &\leq \|f_n - f\|_\infty \mathbb{E}[\|f_n(X)\| + \|f(X)\| + 2\|Y\|] \\ &\leq \|f_n - f\|_\infty \mathbb{E}[\|f_n(X) - f(X)\| + 2\|f(X)\| + 2\|Y\|] \\ &\leq \|f_n - f\|_\infty \mathbb{E}[\|f_n - f\|_\infty + 2\|f(X)\| + 2\|Y\|] \\ &= \|f_n - f\|_\infty^2 + \|f_n - f\|_\infty \mathbb{E}[2\|f(X)\| + 2\|Y\|]. \end{aligned} \quad (62)$$

Observing that $\mathbb{E}[2\|f(X)\| + 2\|Y\|] < \infty$ completes the proof. $\square$

Definition 2.6 (Target Functions). *Let $\mathcal{H}$ be a hypothesis class. All functions*

$$\hat{f}_{Z,\mathcal{H}} \in \arg \min_{f \in \mathcal{H}} \mathcal{E}_Z(f) \quad (63)$$

are called target functions.

Corollary 2.2 (Existence of Target Functions). *Let $\mathcal{H}$ be a hypothesis class. Then the set*

$$\arg \min_{f \in \mathcal{H}} \mathcal{E}_Z(f) \quad (64)$$

is non-empty.

Proof. By Proposition 2.4, $\mathcal{E}_Z : \mathcal{C}(D, \mathbb{R}^n) \rightarrow [0, \infty)$ is continuous whenever $Y \in \mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$, and $\mathcal{H} \subseteq \mathcal{C}(D, \mathbb{R}^n)$ is per definition compact. It is a well-known fact, that continuous functions between metric spaces always attain a minimum (as well as a maximum) on compact sets. $\square$

Remark 2.3. Note that since the regression function $\hat{f}_Z$ does not have to be contained in a given hypothesis class $\mathcal{H}$, our uniqueness-result from Corollary 2.1 does not apply in general; the set $\arg \min_{f \in \mathcal{H}} \mathcal{E}_Z(f)$ may well contain several functions which are not $\mathbb{P}_X$ -equal.

2.3.2 Empirical Error and Empirical Target Functions

The target functions $\hat{f}_{Z,\mathcal{H}}$ are the "closest" we can get to the true regression function within our hypothesis class; ideally, we would like to minimize $\mathcal{E}_Z$ over $\mathcal{H}$ in order to identify candidate functions. But since we do not know the exact distribution of Z , we cannot determine the value of $\mathcal{E}_Z$ in practise. We can, however, use our observed data set to replace the true error function by a computable empirical estimate.

Definition 2.7 (Empirical Error Function ⁵). *The map*

$$\tilde{\mathcal{E}}_{Z^{(m)}} : \Omega \times \mathcal{L}^0(D, \mathbb{R}^n) \rightarrow [0, \infty), \quad \tilde{\mathcal{E}}_{Z^{(m)}(\omega)}(f) = \frac{1}{m} \sum_{i=1}^m \|f(X_i(\omega)) - Y_i(\omega)\|^2, \quad (65)$$

is called "empirical error function", "empirical least squares error" or simply "empirical error". For fixed $\omega \in \Omega$, we denote our realisation of the empirical error as

$$\tilde{\mathcal{E}}_{z^{(m)}} := \tilde{\mathcal{E}}_{Z^{(m)}(\omega)} : \mathcal{L}^0(D, \mathbb{R}^n) \rightarrow [0, \infty). \quad (66)$$

For fixed $f \in \mathcal{L}^0(D, \mathbb{R}^n)$, we write

$$\tilde{\mathcal{E}}_{Z^{(m)},f} := \tilde{\mathcal{E}}_{Z^{(m)}}(f) : \Omega \rightarrow [0, \infty). \quad (67)$$

⁵In the book of Vapnik [4], this quantity is called the empirical *risk* rather than the empirical error

The next proposition shows, that the empirical error is a random variable in its first argument and a continuous function in its second argument.

Proposition 2.5 (Measurability and Continuity of the Empirical Error). *Let $\omega \in \Omega$ and $f \in \mathcal{L}^0(D, \mathbb{R}^n)$. Then the function*

$$\tilde{\mathcal{E}}_{Z^{(m)}, f} : \Omega \rightarrow [0, \infty) \quad (68)$$

is a $\mathcal{G}/\mathcal{B}([0, \infty))$ -measurable random variable. Moreover, the realisation of the empirical error

$$\tilde{\mathcal{E}}_{z^{(m)}} : \mathcal{C}(D, \mathbb{R}^n) \rightarrow [0, \infty) \quad (69)$$

is a continuous map (if restricted to the Banach space of continuous functions).

Proof. Measurability of $\tilde{\mathcal{E}}_{Z^{(m)}, f}$ is trivial if one remembers that the composition of measurable functions is again measurable and that continuous functions between Borel spaces are automatically measurable.

To prove that $\tilde{\mathcal{E}}_{z^{(m)}}$ is continuous, define

$$\delta_x : \mathcal{C}(D, \mathbb{R}^n) \rightarrow \mathbb{R}^n, \quad \delta_x(f) = f(x), \quad (70)$$

as the pointwise evaluation map for some arbitrary $x \in D$. Then we can write

$$\tilde{\mathcal{E}}_{z^{(m)}} = \frac{1}{m} \sum_{i=1}^m \|\delta_{x_i}(f) - y_i\|^2. \quad (71)$$

From this, one can see, that, in order to show continuity of $\tilde{\mathcal{E}}_{z^{(m)}}$, it is sufficient to show continuity of δ_x . The rest follows from the fact that the composition of continuous functions remains continuous. To check, that δ_x is continuous, let f and $(f_n)_{n \geq 1}$ be functions in $\mathcal{C}(D, \mathbb{R}^n)$ with

$$\|f_n - f\|_\infty \rightarrow 0. \quad (72)$$

Then

$$\|\delta_x(f_n) - \delta_x(f)\| = \|f_n(x) - f(x)\| \leq \|f_n - f\|_\infty \rightarrow 0. \quad (73)$$

□

The next results show the existence of a minimizer for $\tilde{\mathcal{E}}_{z^{(m)}}(f)$ as an easy consequence of the previous proposition.

Definition 2.8 (Empirical Target Functions). *Let $\mathcal{H}$ be a hypothesis class. Then every function*

$$\tilde{f}_{z^{(m)}, \mathcal{H}} \in \arg \min_{f \in \mathcal{H}} \tilde{\mathcal{E}}_{z^{(m)}}(f) \quad (74)$$

is called an empirical target function.

Corollary 2.3 (Existence of Empirical Target Functions). *Let $\mathcal{H}$ be a hypothesis class. Then the set*

$$\arg \min_{f \in \mathcal{H}} \tilde{\mathcal{E}}_{z^{(m)}}(f) \quad (75)$$

is non-empty.

Proof. We argue in the same ways as in Corollary 2.2. Under our assumptions, $\mathcal{H} \subseteq \mathcal{C}(D, \mathbb{R}^n)$ is compact and Proposition 2.5 tells us, that $\tilde{\mathcal{E}}_{z^{(m)}} : \mathcal{C}(D, \mathbb{R}^n) \rightarrow [0, \infty)$ is continuous. Therefore, $\tilde{\mathcal{E}}_{z^{(m)}}$ attains a minimum and a maximum on $\mathcal{H}$. $\square$

Remark 2.4 (Computability of Empirical Target Function). Note that, for given data $z^{(m)}$ and a function f from a hypothesis class $\mathcal{H}$, it is possible to compute the exact value of $\tilde{\mathcal{E}}_{z^{(m)}}$ in a finite amount of steps. This opens up the practical possibility to use optimization algorithms to minimize $\tilde{\mathcal{E}}_{z^{(m)}}$ over $\mathcal{H}$ as to identify a concrete empirical target function $\tilde{f}_{z^{(m)}, \mathcal{H}}$. The structure of $\mathcal{H}$ informs the choice of the optimization method.

Since $z^{(m)} = Z^{(m)}(\omega)$ depends on $\omega \in \Omega$, one can see from Definition 2.8, that the content of the set of empirical target functions is influenced by the outcome of a random variable. The next proposition shows that this allows us to construct a stochastic analogon of an empirical target function.

Proposition 2.6 (Random Empirical Target Function). *Let $\mathcal{H}$ be a hypothesis class. Then there is a $\mathcal{G}/\mathcal{B}(\mathcal{H})$ -measurable function (i.e. a random variable)*

$$T_{\mathcal{H}} : \Omega \rightarrow \mathcal{H} \quad (76)$$

such that for all $\omega \in \Omega$, $T_{\mathcal{H}}(\omega)$ is an empirical target function:

$$\forall \omega \in \Omega : T_{\mathcal{H}}(\omega) \in \arg \min_{f \in \mathcal{H}} \tilde{\mathcal{E}}_{Z^{(m)}(\omega)}(f). \quad (77)$$

We therefore denote $T_{\mathcal{H}} := \tilde{f}_{Z^{(m)}, \mathcal{H}}$ and $T_{\mathcal{H}}(\omega) := \tilde{f}_{Z^{(m)}(\omega), \mathcal{H}} = \tilde{f}_{z^{(m)}, \mathcal{H}}$.

To show this technicality, we use two lemmas. The first one is essentially an adapted version of the Measurable Maximum Theorem in Aliprantis and Borders work on infinite dimensional analysis [12]. The second one is a well-known result from topology.

Lemma 2.1. *Let $(\bar{\Omega}, \bar{\mathcal{G}})$ be a measurable space, $\bar{\mathcal{H}}$ be a separable metric space and denote with $2^{\bar{\mathcal{H}}}$ the power set of $\bar{\mathcal{H}}$. Moreover, let the map*

$$\bar{\mathcal{E}} : \bar{\Omega} \times \bar{\mathcal{H}} \rightarrow \mathbb{R} \quad (78)$$

be $\bar{\mathcal{G}}/\mathcal{B}(\mathbb{R})$ -measurable in its first argument and continuous in its second argument⁶ Then the set-valued function

$$\bar{\Omega} \ni \bar{\omega} \mapsto \arg \min_{\bar{f} \in \bar{\mathcal{H}}} \bar{\mathcal{E}}(\bar{\omega}, \bar{f}) \in 2^{\bar{\mathcal{H}}} \quad (79)$$

admits a measurable selector, i.e. there exists a $\bar{\mathcal{G}}/\mathcal{B}(\bar{\mathcal{H}})$ -measurable function

$$\bar{T} : \bar{\Omega} \rightarrow \bar{\mathcal{H}} \quad (80)$$

with the property that

$$\forall \bar{\omega} \in \bar{\Omega} : \bar{T}(\bar{\omega}) \in \arg \min_{\bar{f} \in \bar{\mathcal{H}}} \bar{\mathcal{E}}(\bar{\omega}, \bar{f}). \quad (81)$$

⁶Maps of this type are also called *Carathodory functions*.

Proof. See Theorem 18.19 in [12] for an (even more general) version and proof of this fact. $\square$

Lemma 2.2. *Let (C, s) be a compact metric space. Then (C, s) is separable.*

Proof. Choose $k \in \mathbb{N}_{\geq 1}$. Due to compactness, every open cover of (C, s) has a finite subcover. This implies, that there are finitely many open balls

$$B_{1/k}(c_{k,1}), \dots, B_{1/k}(c_{k,l_k}) \quad (82)$$

with radius $1/k$ and centers

$$c_{k,1}, \dots, c_{k,l_k} \in C \quad (83)$$

in the open cover

$$\{B_{1/k}(c) \mid c \in C\} \supseteq C, \quad (84)$$

such that

$$\bigcup_{i=1}^{l_k} B_{1/k}(c_{k,i}) \supseteq C. \quad (85)$$

Per construction, all elements in C are closer than $1/k$ to at least one ball in the set

$$\{B_{1/k}(c_{k,1}), \dots, B_{1/k}(c_{k,l_k})\}. \quad (86)$$

Iterating this procedure for all $k \in \mathbb{N}_{\geq 1}$ and collecting all the centers $c_{k,j}$ delivers the set

$$\bigcup_{k=1}^{\infty} \bigcup_{i=1}^{l_k} \{c_{k,i}\}, \quad (87)$$

which is countable and dense in (C, d) . $\square$

Now we can proceed with our proof of Proposition 2.6.

Proof (Random Empirical Target Function). We first note, that $(\mathcal{H}, \|\cdot\|_{\infty})$ is a compact metric space and therefore Lemma 2.2 shows that it is automatically separable. Furthermore, Proposition 2.5 assures that the empirical error $\tilde{\mathcal{E}}_{Z^{(m)}(\omega)}(f)$ is $\mathcal{G}/\mathcal{B}(\mathbb{R})$ -measurable in $\omega \in \Omega$ and continuous in $f \in \mathcal{H}$. The result now follows as a direct consequence of Lemma 2.1. $\square$

2.3.3 Asymptotic Behaviour of Empirical Estimates

We have now managed to replace both the real error $\mathcal{E}_Z$ and the real target function $\hat{f}_{Z,\mathcal{H}}$ by random variables $\tilde{\mathcal{E}}_{Z^{(m)}}$ and $\tilde{f}_{Z^{(m)},\mathcal{H}}$, whose realised values can actually be computed on the basis of input data $z^{(m)}$. But of course, the quantities $\tilde{\mathcal{E}}_{Z^{(m)}}$ and $\tilde{f}_{Z^{(m)},\mathcal{H}}$ are only valuable to us, if they asymptotically behave in the way we expect them to and actually approximate $\mathcal{E}_Z$ and $\hat{f}_{Z,\mathcal{H}}$ in some useful way. More specifically, if our number of data points m increases, we would like that, with high probability,

- the empirical error function $\tilde{\mathcal{E}}_{Z^{(m)}}(f)$ is close to the real error function $\mathcal{E}_Z(f)$ for all f in our hypothesis class $\mathcal{H}$ and
- the random empirical target function $\tilde{f}_{Z^{(m)}, \mathcal{H}}$ is such that its random error $\mathcal{E}_Z(\tilde{f}_{Z^{(m)}, \mathcal{H}})$ is close to the optimal error $\mathcal{E}_Z(\hat{f}_{Z, \mathcal{H}})$.

The following statements show that this type of consistency can indeed be achieved under some further restrictions on our hypothesis class $\mathcal{H}$. We start with a proposition describing the limit behaviour of the empirical error of a fixed functions f , of which we require nothing more but square-integrability with respect to $\mathbb{P}_X$.

Proposition 2.7 (Convergence of Empirical Error). *Let $f \in \mathcal{L}^2(D, \mathbb{R}^n, \mathbb{P}_X)$ be fixed. Then it holds, that, for $m \rightarrow \infty$, the random variable $\tilde{\mathcal{E}}_{Z^{(m)}, f}$ converges $\mathbb{P}$ -a.s. and in probability to $\mathcal{E}_Z(f)$. More specifically,*

$$\mathbb{P}(\{\lim_{m \rightarrow \infty} \tilde{\mathcal{E}}_{Z^{(m)}, f} = \mathcal{E}_Z(f)\}) = 1 \quad (88)$$

and, for all $\varepsilon > 0$,

$$\lim_{m \rightarrow \infty} \mathbb{P}(\{|\tilde{\mathcal{E}}_{Z^{(m)}, f} - \mathcal{E}_Z(f)| \geq \varepsilon\}) = 0. \quad (89)$$

Proof. It is sufficient to show $\mathbb{P}$ -almost sure convergence, since this automatically implies convergence in probability. For $i \in \{1, \dots, m\}$, define

$$V_{i, f} : \Omega \rightarrow [0, \infty), \quad V_{i, f}(\omega) = \|f(X_i(\omega)) - Y_i(\omega)\|^2. \quad (90)$$

Observe, that all $V_{i, f}$ as well as

$$\tilde{\mathcal{E}}_{Z^{(m)}, f} = \frac{1}{m} \sum_{i=1}^m V_{i, f} \quad (91)$$

are $\mathcal{G}/\mathcal{B}([0, \infty))$ -measurable. By construction, $V_{1, f}, \dots, V_{m, f}$ are independent and identically distributed. Moreover, $f \in \mathcal{L}^2(D, \mathbb{R}^n, \mathbb{P}_X)$ implies, that the expectation of $V_{1, f}$ is finite and coincides with the error of f :

$$\mathbb{E}[V_{1, f}] = \mathbb{E}[\|f(X_1(\omega)) - Y_1(\omega)\|^2] = \mathcal{E}_Z(f) \in [0, \infty). \quad (92)$$

$\mathbb{P}$ -almost sure convergence of $\tilde{\mathcal{E}}_{Z^{(m)}, f}$ against $\mathcal{E}_Z(f)$ for $m \rightarrow \infty$ now follows as a direct consequence of the strong law of large numbers ⁷. $\square$

Remark 2.5. Note that since D is compact, all continuous functions from D to $\mathbb{R}^n$ are bounded and therefore $\mathcal{C}(D, \mathbb{R}^n) \subseteq \mathcal{L}^2(D, \mathbb{R}^n, \mathbb{P}_X)$. This implies, that the empirical error converges $\mathbb{P}$ -a.s. and in probability to the real error if f is continuous, which is always the case if f is an element of a hypothesis class.

⁷For a precise formulation of the well-known strong law of large numbers, take a look at Theorem 2.4.1 in the book of Durrett [8]

Proposition 2.7 shows convergence of the empirical error to the true error in the case where f is fixed. But we can do even better than that: we now cite two theorems, which summarise some of the main results of the paper of Cucker and Smale [5]. The first one provides a quantitative bound for the speed of probabilistic convergence of $\tilde{\mathcal{E}}_{Z^{(m)},f}$ for a fixed function f . The second one gives an assurance, that $\tilde{\mathcal{E}}_{Z^{(m)}}$ converges uniformly in probability to $\mathcal{E}_Z$ over the whole hypothesis class $\mathcal{H}$ and that the error of the random empirical target function $\mathcal{E}_Z(\hat{f}_{Z^{(m)},\mathcal{H}})$ indeed approximates the error of an actual target function $\mathcal{E}_Z(\hat{f}_{Z,\mathcal{H}})$. For these results, we require a boundedness-assumption on the functions in $\mathcal{H}$ as well as the concept of *covering numbers*, which serve as a measure of complexity for hypothesis classes and affect the speed of convergence of our empirical quantities.

Theorem 2.2 (Convergence Rate of Empirical Error). *Let $\mathcal{H}$ be a hypothesis class and $f \in \mathcal{H}$ be fixed. Let $M > 0$ and suppose that*

$$\|f(X) - Y\| \leq M \quad \mathbb{P}\text{-a.s.} \quad (93)$$

In addition, define

$$\sigma_f^2 := \mathbb{V}[\|f(X) - Y\|^2]. \quad (94)$$

Then,

$$0 \leq \sigma_f^2 \leq M^4 < \infty \quad (95)$$

and for all $\varepsilon > 0$,

$$\mathbb{P}(\{|\tilde{\mathcal{E}}_{Z^{(m)},f} - \mathcal{E}_Z(f)| \leq \varepsilon\}) \geq 1 - 2e^{-\frac{m\varepsilon^2}{2(\sigma_f^2 + M^2\varepsilon)}} \quad (96)$$

Proof. The result follows from Bernstein's inequality, which can in turn be interpreted as a quantitative version of the weak law of large numbers. Details can be found in [5] (Theorem A). $\square$

Definition 2.9 (Covering Numbers of Hypothesis Classes). *Let $\mathcal{H}$ be a hypothesis class and $\varepsilon > 0$. For any $f \in \mathcal{H}$, let*

$$B_\varepsilon(f) := \{g \in \mathcal{H} \mid \|f - g\|_\infty < \varepsilon\} \quad (97)$$

be the ε -ball around f in $\mathcal{H}$. Then we define the covering number $\text{Cov}(\mathcal{H}, \varepsilon)$ of $\mathcal{H}$ with respect to ε as the minimum number of ε -balls required to cover $\mathcal{H}$:

$$\text{Cov}(\mathcal{H}, \varepsilon) := \min\{k \in \mathbb{N}_{\geq 1} \mid \exists f_1, \dots, f_k \in \mathcal{H} : \bigcup_{i=1}^k B_\varepsilon(f_i) \supseteq \mathcal{H}\}. \quad (98)$$

Remark 2.6. Note that since we assume that $\mathcal{H}$ is compact, covering numbers are always well-defined and finite. The scaling behaviour of $\text{Cov}(\mathcal{H}, \varepsilon)$ with ε can be interpreted as a measure of complexity for $\mathcal{H}$.

Theorem 2.3 (Consistency of the Principle of Empirical Risk Minimization). *Let $\mathcal{H}$ be a hypothesis class and $M > 0$. Suppose that, for all $f \in \mathcal{H}$,*

$$\|f(X) - Y\| \leq M \quad \mathbb{P}\text{-a.s.} \quad (99)$$

Moreover, define

$$\sigma_{\mathcal{H}}^2 := \sup_{f \in \mathcal{H}} \mathbb{V}[\|f(X) - Y\|^2]. \quad (100)$$

Then

$$0 \leq \sigma_{\mathcal{H}}^2 \leq M^4 < \infty. \quad (101)$$

Furthermore, for all $\varepsilon > 0$,

$$\mathbb{P}(\{\sup_{f \in \mathcal{H}} |\tilde{\mathcal{E}}_{Z^{(m)},f} - \mathcal{E}_Z(f)| \leq \varepsilon\}) \geq 1 - \text{Cov}(\mathcal{H}, \frac{\varepsilon}{8M}) 2e^{-\frac{m\varepsilon^2}{4(2\sigma_{\mathcal{H}}^2 + \frac{M^2\varepsilon}{3})}} \quad (102)$$

and

$$\mathbb{P}(\{\mathcal{E}_Z(\tilde{f}_{Z^{(m)},\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_{Z,\mathcal{H}}) \leq \varepsilon\}) \geq 1 - \text{Cov}(\mathcal{H}, \frac{\varepsilon}{16M}) 2e^{-\frac{m\varepsilon^2}{8(4\sigma_{\mathcal{H}}^2 + \frac{M^2\varepsilon}{3})}}. \quad (103)$$

Proof. See Theorems B and C in [5]. $\square$

Theorem 2.3 assures us, that the empirical error $\tilde{\mathcal{E}}_{Z^{(m)}}$ indeed approximates the real error $\mathcal{E}_Z$ in a strong, uniform sense over the whole hypothesis class $\mathcal{H}$ if the associated functions fulfill a boundedness-requirement. But even more importantly, under the same assumptions, the error of the random empirical target function $\mathcal{E}_Z(\tilde{f}_{Z^{(m)},\mathcal{H}})$ converges in probability to the error of an actual target function $\mathcal{E}_Z(\hat{f}_{Z,\mathcal{H}})$. These results imply the *consistency* of our learning strategy of empirical risk minimization as originally defined by Vapnik [4]. We can now be sure, that, given a predefined hypothesis class $\mathcal{H}$, it is in fact possible to "learn" a close-to-optimal functional relation between X and Y purely on the basis of a sufficient amount of empirical data $z^{(m)}$. In this sense, theorems 2.5 and 2.3 serve as a theoretical justification for our solution strategy for the statistical learning problem 2.4.

But before, we briefly put our attention to what happens if we keep the amount of data m fixed but increase the size of $\mathcal{H}$.

Definition 2.10 (Bias-Variance Decomposition of the Error). *Let $\mathcal{H}$ be a hypothesis class. Define*

$$\mathcal{S}_{Z^{(m)},\mathcal{H}} := \mathcal{E}_Z(\tilde{f}_{Z^{(m)},\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_{Z,\mathcal{H}}) \quad (104)$$

and

$$\mathcal{A}_{\mathcal{H}} := \mathcal{E}_Z(\hat{f}_{Z,\mathcal{H}}) - \mathcal{E}_Z(\hat{f}_Z) \quad (105)$$

Then $\mathcal{S}_{Z^{(m)},\mathcal{H}}$ is called *sample error (or variance)* and $\mathcal{A}_{\mathcal{H}}$ is referred to as *approximation error (or bias)*.

Remark 2.7 (Bias-Variance Tradeoff). From Definition 2.10 it is immediately clear, that

$$\mathcal{E}_Z(\tilde{f}_{Z^{(m)}, \mathcal{H}}) = \mathcal{S}_{Z^{(m)}, \mathcal{H}} + \mathcal{A}_{\mathcal{H}} + \mathcal{E}_Z(\hat{f}_Z). \quad (106)$$

Note, that the variance $\mathcal{S}_{Z^{(m)}, \mathcal{H}}$ is the only random variable in this decomposition; it depends on the realisation of $Z^{(m)}$ and on our choice of $\mathcal{H}$. The bias $\mathcal{A}_{\mathcal{H}}$, on the other hand, depends only on $\mathcal{H}$. Last, but not least, $\mathcal{E}_Z(\hat{f}_Z)$ neither depends on $Z^{(m)}$ nor $\mathcal{H}$. In addition, all three quantities are non-negative:

$$\mathcal{S}_{Z^{(m)}, \mathcal{H}}, \mathcal{A}_{\mathcal{H}}, \mathcal{E}_Z(\hat{f}_Z) \geq 0. \quad (107)$$

If, for fixed sample size m , we enlarge $\mathcal{H}$, then $\mathcal{S}_{Z^{(m)}}$ tends to grow, $\mathcal{A}_{\mathcal{H}}$ cannot grow and tends to shrink, and $\mathcal{E}_Z(\hat{f}_Z)$ forms an irreducible error which stays unaffected. This phenomenon, known as bias-variance tradeoff, is an unavoidable challenges in the design of machine learning algorithms. Choosing $\mathcal{H}$ too large/small, results in an empirical target function $\tilde{f}_{z^{(m)}, \mathcal{H}}$ which overfits/underfits to the available data $z^{(m)}$, resulting in a suboptimal value of $\mathcal{E}_Z(\tilde{f}_{z^{(m)}, \mathcal{H}})$. In practise, often numerical simulations are used in the search for the size of $\mathcal{H}$ which optimally balances bias and variance. Technical modifications, whose aim it is to decrease the expected amount of variance $\mathcal{S}_{Z^{(m)}}$, are widely used in practise and are referred to as *regularization methods*.

We close this chapter with a concise summary of how to tackle the statistical learning problem.

Remark 2.8 (Solution Strategy for the Statistical Learning Problem). The results discussed in this chapter lead to an applied strategy to approach the statistical learning problem described in Definition 2.4. This strategy can be seen as a general guide to solve supervised machine learning tasks in practise. A summary can be found below:

1. **Regularity Requirements.** Make sure, that the requirements in our basic setting described in 2.1 and 2.2 represent an (at least more or less) correct model of reality: $D \subseteq \mathbb{R}^d$ should be compact, Y should be in $\mathcal{L}^2(\Omega, \mathbb{R}^n, \mathbb{P})$, and $z^{(m)} = (z_1, \dots, z_m) = ((x_1, y_1), \dots, (x_m, y_m))$ should consists of m independent, identically distributed realisations of $Z = (X, Y)$.
2. **Hypothesis Space.** Choose a compact hypothesis class of continuous functions $\mathcal{H} \subseteq \mathcal{C}(D, \mathbb{R}^n)$. For our theory to be applicable, $\mathcal{H}$ must be compatible with the learning problem in the sense that there must be an $M > 0$, such that, for all $f \in \mathcal{H}$, the inequality

$$\|f(X) - Y\| \leq M \quad (108)$$

holds $\mathbb{P}$ -a.s.

3. **Empirical Error.** Use the data set $z^{(m)}$ to construct a realisation of the empirical error function $\tilde{\mathcal{E}}_{z^{(m)}} : \mathcal{H} \rightarrow [0, \infty)$.

4. **Training.** Find a way to minimize $\tilde{\mathcal{E}}_{z^{(m)}}$ over $\mathcal{H}$ in order to identify an empirical target function $\tilde{f}_{Z^{(m)}, \mathcal{H}}$, which can then be used to predict Y on the basis of X . This step is referred to as *training* of the model using the *training set* $z^{(m)}$.
5. **Meta-Optimization.** Methods to further optimize this process include (but are not limited to): gathering more data (increasing m), trying several hypothesis classes to find a better fit for the problem and a better balance between bias and variance, improving the minimization algorithm used to identify $\tilde{f}_{Z^{(m)}, \mathcal{H}}$, and reducing the dimensionality of $x_1, \dots, x_m$ to obtain low-dimensional information-preserving *features* of $x_1, \dots, x_m$ instead of the original data.

Remark 2.9 (Generalizing the Predictor Variable). Note, that one can replace $D \subseteq \mathbb{R}^d$ with an arbitrary (abstract) compact topological space $\tilde{D}$ in the complete theory of Section 2.2 and still arrive at the same results.

3 Artificial Neural Networks

This chapter is devoted to the description of a subtype of a powerful and flexible family of hypothesis classes known as *artificial neural networks* (ANNs), first introduced in 1943 by McCulloch and Pitts [13]. In recent years, ANNs in its different forms (classical feedforward neural networks, convolutional neural networks, recurrent neural networks,...) have achieved dramatic success in a wide variety of application areas, such as speech perception [14] [15] [16], image classification [17] [18] [19] and the synthetic generation of music and images [20] [21] [22]. Today, ANNs have become the dominant machine learning paradigm and are considered state-of-the-art technology for a broad class of tasks.

Here, we focus on hypothesis classes consisting of *classical fully-connected feedforward* ANNs, also known as *multilayer perceptrons*, which merely represent the oldest and most-established subspecies of a plethora of possible neural network architectures⁸. Below, we give a mathematical description of such ANNs, along with some important properties and theorems, and describe the standard method of minimizing the empirical error over the associated hypothesis classes.

3.1 Definition and Properties of Classical Artificial Neural Networks

We start with a rigorous mathematical definition of neural networks in terms of compositions of well-known types of functions, and subsequently describe an intuitive way to visualize these compositions as actual networks.

Definition 3.1 (Concatenation of Vectors). *Let $k_1, \dots, k_j \in \mathbb{N}_{\geq 1}$ natural numbers, $v_1 \in \mathbb{R}^{k_1}, \dots, v_j \in \mathbb{R}^{k_j}$ be vectors and $k := k_1 + \dots + k_j$. Then*

$$\begin{aligned} & v_1 \frown v_2 \frown \dots \frown v_{j-1} \frown v_j \\ & := (v_{1,1}, \dots, v_{1,k_1}, v_{2,1}, \dots, v_{2,k_2}, \dots, v_{j-1,1}, \dots, v_{j-1,k_{j-1}}, v_{j,1}, \dots, v_{j,k_j}) \in \mathbb{R}^k \end{aligned} \quad (109)$$

denotes the concatenation of $v_1, \dots, v_j$ to a single vector.

Definition 3.2 (Vectorization of Matrices). *Let $k_1, k_2 \in \mathbb{N}$ be natural numbers and $W = (W_{i,j}) \in \mathbb{R}^{k_1 \times k_2}$ be a matrix. Then*

$$\text{vec}(W) := (W_{1,1}, \dots, W_{k_1,1}, \dots, W_{1,k_2}, \dots, W_{k_1,k_2}) \in \mathbb{R}^{k_1 k_2} \quad (110)$$

denotes the vectorization of W , constructed by stacking the columns of W on top of one another.

Definition 3.3 (Fully-Connected Feedforward Artificial Neural Networks). *Let*

- $\rho \in \mathcal{C}(\mathbb{R}, \mathbb{R})$ be a continuous function

⁸For a thorough treatment of various types of artificial neural networks in the broader context of machine learning, consider, for example, the book of Goodfellow, Bengio and Courville [23] or the work of Bishop [24]

- $l \in \mathbb{N}_{\geq 1}$ be a natural number,
- $a := (a_0, \dots, a_{l+1}) \in \mathbb{N}_{\geq 1}^{l+2}$ an $l+2$ -tuple of natural numbers,
- $s : \mathbb{N}_{\geq 1}^{l+2} \rightarrow \mathbb{N}_{\geq 1}$ be a function given by $s(\tilde{a}) = \sum_{i=0}^l \tilde{a}_{i+1} \tilde{a}_i + \tilde{a}_{l+1}$, and
- $\mathbb{R}^{s(a)} \ni \theta \mapsto ((W^{(i)}(\theta), b^{(i)}(\theta)))_{i=0}^l \in \times_{i=0}^l (\mathbb{R}^{a_{i+1} \times a_i} \times \mathbb{R}^{a_{i+1}})$ be a bijective mapping which assigns an $s(a)$ -dimensional vector θ to an $(l+1)$ -tuple of matrix-vector pairs such that the entries of $W^{(0)}, \dots, W^{(l)}$ and $b^{(0)}, \dots, b^{(l)}$ are identical to the components of the vector θ :

$$\text{vec}(W^{(0)}(\theta)) \frown b^{(0)}(\theta) \frown \dots \frown \text{vec}(W^{(l)}(\theta)) \frown b^{(l)}(\theta) := \theta. \quad (111)$$

For arbitrary $k_1, k_2 \in \mathbb{N}_{\geq 1}$, $W \in \mathbb{R}^{k_2 \times k_1}$, and $b \in \mathbb{R}^{k_2}$, denote with

$$\varphi_{W,b} : \mathbb{R}^{k_1} \rightarrow \mathbb{R}^{k_2}, \quad \varphi_{W,b}(x) = Wx + b \quad (112)$$

the associated affine linear map and with

$$\rho_{k_2} : \mathbb{R}^{k_2} \rightarrow \mathbb{R}^{k_2}, \quad \rho_{k_2}((x_1, \dots, x_{k_2})) = (\rho(x_1), \dots, \rho(x_{k_2})), \quad (113)$$

the componentwise application of ρ to a k_2 -dimensional vector.

Then ρ, l, a uniquely define a function

$$\begin{aligned} \Phi_{\rho,l,a} : \mathbb{R}^{s(a)} \times \mathbb{R}^{a_0} &\rightarrow \mathbb{R}^{a_{l+1}}, \\ \Phi_{\rho,l,a}(\theta, x) &= \varphi_{W^{(l)}(\theta), b^{(l)}(\theta)} \circ \rho_{a_l} \circ \varphi_{W^{(l-1)}(\theta), b^{(l-1)}(\theta)} \dots \circ \rho_{a_1} \circ \varphi_{W^{(0)}(\theta), b^{(0)}(\theta)}(x). \end{aligned} \quad (114)$$

Similarly, ρ, l, a and $\theta \in \mathbb{R}^{s(a)}$ uniquely define a continuous function

$$\begin{aligned} \Phi_{\rho,l,a}(\theta, \cdot) : \mathbb{R}^{a_0} &\rightarrow \mathbb{R}^{a_{l+1}}, \\ x &\mapsto \Phi_{\rho,l,a}(\theta, x). \end{aligned} \quad (115)$$

$\Phi_{\rho,l,a}(\theta, \cdot)$ is called a fully connected feedforward artificial neural network with activation function ρ , a number of hidden layers l , an associated architecture a , parameters θ , a size (or number of parameters) $s(a)$, (parameter) weights $W^{(0)}(\theta), \dots, W^{(l)}(\theta)$ and (parameter) biases $b^{(0)}(\theta), \dots, b^{(l)}(\theta)$. The quantities ρ, l, a are called hyperparameters. If $l > 1$, then $\Phi_{\rho,l,a}(\theta, \cdot)$ is often referred to as deep network.

Setting 3.1. For the rest of this thesis,

- the term *neural network* always refers to the type of feedforward neural networks specified in Definition 3.3,
- the symbol $\Phi_{\rho,l,a}(\theta, \cdot)$ always refers to a neural network with hyperparameters ρ, l, a , parameters θ , and a size (or number of parameters) $s(a) = \sum_{i=0}^l a_{i+1}a_i + a_{l+1}$,

- every pair (l, a) consisting of a number of hidden layers l and architecture a implicitly comes with associated parameter maps $W^{(k)} : \mathbb{R}^{s(a)} \rightarrow \mathbb{R}^{a_{k+1} \times a_k}$ and $b^{(k)} : \mathbb{R}^{s(a)} \rightarrow \mathbb{R}^{a_{k+1}}$ for $k \in \{1, \dots, s(a)\}$, as described in Definition 3.3, and
- the domain of $\Phi_{\rho, l, a}(\theta, \cdot)$ can either be $\mathbb{R}^{a_0}$ or, if specified differently, any other subset $D \subseteq \mathbb{R}^{a_0}$, which is made clear from the context without the use of function restriction symbols.

Remark 3.1 (Activation Functions). Frequently used neural network activation functions ρ include the rectified linear unit function $\text{relu}(x) = \max(\{0, x\})$ and the sigmoid activation function $\text{sigm}(x) = \frac{1}{1+e^{-x}}$.

Remark 3.2 (Neural Networks as Directed Acyclic Graphs). Every neural network $\Phi_{\rho, l, a}(\theta, \cdot)$ is associated with a directed acyclic graph $(N_{l, a}, C_{l, a})$, whose structure is determined by the number of hidden layers l and the architecture $a = (a_0, \dots, a_{l+1})$ ⁹. The set of vertices of this graph is defined as

$$N_{l, a} := \{(q, j) \mid j \in \{0, \dots, l+1\}, q \in \{1, \dots, a_j\}\} \quad (116)$$

and can be interpreted as a set of artificial neurons forming a layered structure, where (q, j) refers to the q -th neuron in the j -th layer. We speak of the input layer ($j = 0$), the hidden layers ($j \in \{1, \dots, l\}$), and the output layer ($j = l+1$). A set of directed edges between our vertices (i.e. a set of connections between our neurons) is given by

$$C_{l, a} := \{((q_1, j), (q_2, j+1)) \mid (q_1, j), (q_2, j+1) \in N_{l, a}\} \subset N_{l, a} \times N_{l, a}. \quad (117)$$

In words, for $j \in \{0, \dots, l\}$, there are directed edges going from all neurons of the j -th layer to all neurons of the $j+1$ -th layer. It is not hard to illustrate the network $(N_{l, a}, C_{l, a})$ (see Figure 1).

⁹Obviously, l can be derived from the number of entries in the tuple a . However, for the sake of completeness and to remind the reader that the number of entries in a varies, we mention both l and a .

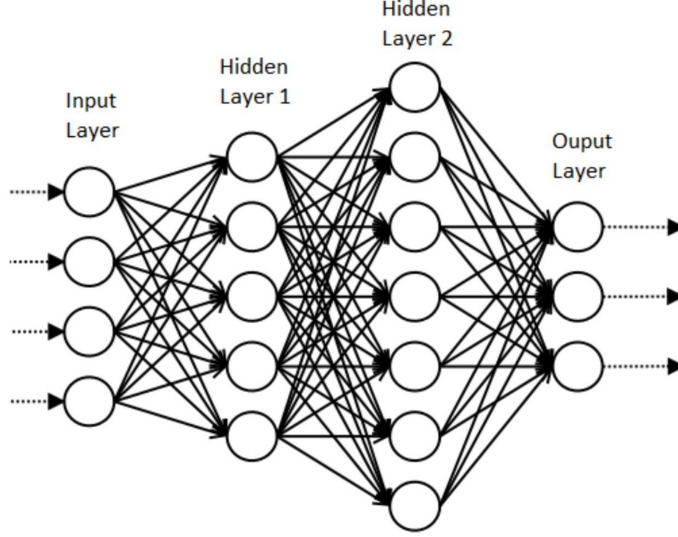


Figure 1: Artificial neural network with $l = 2$ hidden layers and architecture $a = (4, 5, 7, 3)$ depicted as directed acyclic graph. Source: <https://goo.gl/images/qQ1B41> (20.02.2019).

To see, how $(N_{l,a}, C_{l,a})$ can be used to visualize the computation of the function value $\Phi_{\rho,l,a}(\theta, x) \in \mathbb{R}^{a_l+1}$ for some $x = (x_1, \dots, x_{a_0}) \in \mathbb{R}^{a_0}$, we first imagine the vector x entering the input layer. More specifically, we imagine each value x_i entering the input neuron $(i, 0)$. Then, along each edge $((i, 0), (k, 1))$, x_i is transferred to the neuron $(k, 1)$ in the next layer while being multiplied with an edgeweight $w_{k,i}^{(0)}$; this globally results in the vector $W^{(0)}x$ entering the first hidden layer. At this hidden layer, a bias $b_k^{(0)}$ is added to the input of each neuron $(k, 1)$ and then the activation function ρ is applied, leading to the state $\rho_{a_1}(W^{(0)}x + b^{(0)})$ at the first hidden layer. The computation then starts again with new weights $W^{(1)}$ and biases $b^{(1)}$ whereby the values are transferred from the first to the second hidden layer. This cycle repeats until the flow of values reaches the output layer. The only layers at which no activation function ρ is applied are the input and the output layer.

We now focus our attention on an approximation result, which is of central importance in the theory of ANNs, because it assures that they are, at least in theory, flexible enough to model all continuous functions with arbitrary precision. We then proceed by introducing the compact hypothesis class of neural networks with bounded parameters and fixed hyperparameters.

Definition 3.4 (Neural Networks with Fixed Input and Output Dimensions). *Let $d, n \in \mathbb{N}_{\geq 1}$, $D \subseteq \mathbb{R}^d$, and $\rho \in \mathcal{C}(\mathbb{R}, \mathbb{R})$. Then we define the family of all neural networks with activation function ρ which map from the domain D to $\mathbb{R}^n$*

as the set of functions

$$\begin{aligned} & \mathcal{N}_\rho(D, \mathbb{R}^n) \\ & := \{\Phi_{\rho,l,a}(\theta, \cdot) : D \rightarrow \mathbb{R}^n \mid l \in \mathbb{N}_{\geq 1}, a \in \mathbb{N}_{\geq 1}^l, (a_0, a_{l+1}) = (d, n), \theta \in \mathbb{R}^{s(a)}\}. \end{aligned} \quad (118)$$

Remark 3.3. Note, that for $D_1, D_2 \subseteq \mathbb{R}^d$, the sets $\mathcal{N}_\rho(D_1, \mathbb{R}^n)$ and $\mathcal{N}_\rho(D_2, \mathbb{R}^n)$ contain the exact same neural networks, restricted to different domains. Moreover, for any compact $D \subseteq \mathbb{R}^d$, the family $\mathcal{N}_\rho(D, \mathbb{R}^n)$ is a (non-compact) subset of the Banach space $(\mathcal{C}(D, \mathbb{R}^n), \|\cdot\|_\infty)$, since all neural networks are automatically continuous.

Definition 3.5 (Set of Polynomials). *Let $d, n \in \mathbb{N}_{\geq 1}$ and $D \subseteq \mathbb{R}^d$. Then we denote with $\mathcal{P}(D, \mathbb{R}^n)$ the set of all (possibly multi-variable) polynomial functions from D to $\mathbb{R}^n$.*

Theorem 3.1 (Universal Approximation Theorem). *Let $d, n \in \mathbb{N}_{\geq 1}$ and let $\rho \in \mathcal{C}(\mathbb{R}, \mathbb{R}) \setminus \mathcal{P}(\mathbb{R}, \mathbb{R})$ be a continuous function which is not a polynomial. Furthermore, let $D \subseteq \mathbb{R}^d$ be a compact set. Now choose a continuous function $f \in \mathcal{C}(D, \mathbb{R}^n)$. Then, for all $\varepsilon > 0$, there exists a neural network $\Phi_{\rho,l,a}(\theta, \cdot) \in \mathcal{N}_\rho(D, \mathbb{R}^n) \subseteq \mathcal{C}(D, \mathbb{R}^n)$ such that*

$$\|f - \Phi_{\rho,l,a}(\theta, \cdot)\|_\infty < \varepsilon. \quad (119)$$

In other words, $\mathcal{N}_\rho(D, \mathbb{R}^n)$ is a dense subset of the Banach space $(\mathcal{C}(D, \mathbb{R}^n), \|\cdot\|_\infty)$. Moreover, it is always possible to choose $\Phi_{\rho,l,a}(\theta, \cdot)$ in such a way as that $l = 1$, i.e. one hidden layer is enough to approximate all continuous functions with arbitrary precision by a neural network.

Proof. For the sake of brevity, we only prove the special case where

$$\rho \in \mathcal{C}^\infty(\mathbb{R}, \mathbb{R}) \subseteq \mathcal{C}(\mathbb{R}, \mathbb{R}) \quad (120)$$

is smooth and $d = n = 1$. To see treatments of the proof in full generality, consider the works of Pinkus [25], Leshno and Schocken [26], Hornik et al. [27] or Cybenko [28].

We start by recalling, that the well known Stone-Weierstrass Theorem¹⁰ assures us, that, since D is compact, the set of polynomials $\mathcal{P}(D, \mathbb{R})$ is dense in $(\mathcal{C}(D, \mathbb{R}), \|\cdot\|_\infty)$. Therefore, it is sufficient to prove, that all polynomials $p \in \mathcal{P}(D, \mathbb{R})$ can be approximated arbitrarily well (with respect to the $\|\cdot\|_\infty$ -norm) by elements of the set

$$\mathcal{N}_{\rho,l=1}(D, \mathbb{R}) := \{\Phi_{\rho,l,a}(\theta, \cdot) \in \mathcal{N}_\rho(D, \mathbb{R}) \mid l = 1\} \quad (121)$$

of all neural networks with $l = 1$ hidden layer. A different, yet equivalent way to characterize $\mathcal{N}_{\rho,l=1}(D, \mathbb{R})$ is as the linear span of a set of functions in the

¹⁰For a formulation and proof of the Stone-Weierstrass Theorem under general assumptions, go to Theorem 7.32 in Rubin's book on mathematical analysis [29]

vector space $\mathcal{C}(D, \mathbb{R})$:

$$\begin{aligned}\mathcal{R}_\rho(D, \mathbb{R}) &:= \{r_{w,d} : D \rightarrow \mathbb{R} \mid \exists w, d \in \mathbb{R} : r(x) = \rho(wx + d)\} \\ \mathcal{R}_\rho(D, \mathbb{R}) &\subseteq \text{span}(\mathcal{R}_\rho(D, \mathbb{R})) = \mathcal{N}_{\rho,l=1}(D, \mathbb{R})\end{aligned}\quad (122)$$

It is not hard to show, that it is possible to approximate all polynomials $p \in \mathcal{P}(D, \mathbb{R})$ by neural networks in $\mathcal{N}_{\rho,l=1}(D, \mathbb{R})$ with arbitrary precision if and only if it is possible to approximate all monomials

$$p_{\lambda,k} \in \{p_{\lambda,k} : D \rightarrow \mathbb{R} \mid \exists \lambda \in \mathbb{R}, k \in \mathbb{N}_{\geq 0} : p_{\lambda,k} = \lambda x^k\} \subset \mathcal{P}(D, \mathbb{R}) \quad (123)$$

by neural networks in $\mathcal{N}_{\rho,l=1}(D, \mathbb{R})$ with arbitrary precision. This justifies that, w.l.o.g., we can only focus on the approximation of monomials $p_{\lambda,k}$.

A monomial $p_{\lambda,k}$ can be approximated by elements in $\mathcal{N}_{\rho,l=1}(D, \mathbb{R})$ with respect to the $\|\cdot\|_\infty$ -norm if and only if $p_{\lambda,k}$ is contained in the topological closure of $\mathcal{N}_{\rho,l=1}(D, \mathbb{R})$; so what is left to prove is that, for all $k \in \mathbb{N}_{\geq 0}$ and all $\lambda \in \mathbb{R}$,

$$p_{\lambda,k} \in \overline{\mathcal{N}_{\rho,l=1}(D, \mathbb{R})}. \quad (124)$$

We proceed by noting, that smoothness and non-polynomiality of ρ imply¹¹, that there exists an $x_0 \in \mathbb{R}$ such that for all $k \in \mathbb{N}_{\geq 0}$ it holds that

$$\rho^{(k)}(x_0) \neq 0. \quad (125)$$

Using x_0 , we observe, that the neural network $r_{0,x_0} \in \mathcal{R}_\rho(D, \mathbb{R}) \subseteq \mathcal{N}_{\rho,l=1}(D, \mathbb{R})$ is a constant function:

$$r_{0,x_0} = \rho(0 * x + x_0) = \rho(x_0) \neq 0. \quad (126)$$

Therefore, all monomials $p_{\lambda,0} \in \mathcal{P}(D, \mathbb{R})$ of degree $k = 0$ (i.e. all constant functions) must be contained in $\text{span}(\mathcal{R}_\rho(D, \mathbb{R})) = \mathcal{N}_{\rho,l=1}(D, \mathbb{R})$. To see, that this inclusion also holds for all monomials of degree $k = 1$, choose $h \neq 0$, $\tilde{\lambda} \in \mathbb{R}$ and consider the neural network

$$\frac{r_{h+\tilde{\lambda},x_0} - r_{\tilde{\lambda},x_0}}{h} \in \mathcal{N}_{\rho,l=1}(D, \mathbb{R}) \quad (127)$$

with $r_{h+\tilde{\lambda},x_0}, r_{\tilde{\lambda},x_0} \in \mathcal{R}_\rho(D, \mathbb{R})$. Taking the limit $h \rightarrow 0$ in the Banach space $(\mathcal{C}(D, \mathbb{R}^n), \|\cdot\|_\infty)$, we obtain

$$\lim_{h \rightarrow 0} \frac{r_{h+\tilde{\lambda},x_0} - r_{\tilde{\lambda},x_0}}{h} = \frac{d}{d\lambda} r_{\lambda,x_0} \big|_{\lambda=\tilde{\lambda}} \in \overline{\mathcal{N}_{\rho,l=1}(D, \mathbb{R})}. \quad (128)$$

This convergence can be shown rigorously using the mean value theorem. We now compute for $\tilde{\lambda} = 0$ and $x \in D$:

$$\frac{d}{d\lambda} r_{\lambda,x_0} \big|_{\lambda=0}(x) = x\rho'(x_0) = p_{\rho'(x_0),1}(x) \neq 0. \quad (129)$$

¹¹This implication was first shown by Corominas and Balaguer [30] in 1954.

This implies, that the monomial $p_{\rho'(x_0),1}$ is an element of $\overline{\mathcal{N}_{\rho,l=1}(D, \mathbb{R})}$ and it follows that all monomials of degree $k = 1$ must be elements of $\overline{\mathcal{N}_{\rho,l=1}(D, \mathbb{R})}$. To show the same inclusion for monomials $p_{\lambda,k}$ with degree $k = 2$, one can replace r_{λ,x_0} by $\frac{d}{d\lambda} r_{\lambda,x_0}$ before constructing the difference quotient in Expression 127, repeat the steps above and observe that

$$\frac{d^2}{d\lambda^2} r_{\lambda,x_0}|_{\lambda=0}(x) = x^2 \rho^{(2)}(x_0) = p_{\rho^{(2)}(x_0),2}(x) \neq 0. \quad (130)$$

The only subtle difference to the case $k = 1$ is, that now the difference quotient in 127 is in $\overline{\mathcal{N}_{\rho,l=1}(D, \mathbb{R})}$ instead of $\mathcal{N}_{\rho,l=1}(D, \mathbb{R})$, but this does not impact the validity of the argument. Continuing to iterate the argument like this over all remaining monomial degrees $k \in \mathbb{N}_{\geq 2}$ while noting that

$$\frac{d^k}{d\lambda^k} r_{\lambda,x_0}|_{\lambda=0}(x) = x^k \rho^{(k)}(x_0) = p_{\rho^{(k)}(x_0),k}(x) \neq 0 \quad (131)$$

completes the proof. $\square$

Remark 3.4. Theorem 3.1 shows that *depth-bounded* neural networks (i.e. neurons with only one hidden layer but an arbitrary number of hidden neurons) are universal approximators. In 2017, Lu et al. [31] showed an analogous and equally strong approximation theorem for *width-bounded* neural networks. Such networks are allowed to have an arbitrary number of hidden layers but no more than $a_{max}(a_0) \in \mathbb{N}_{\geq 1}$ hidden neurons in each layer, where $a_{max}(a_0)$ is some global upper bound on the width of the network which depends only on the input dimension a_0 .

Remark 3.5 (Practical Limitations of Neural Network Approximation). Theorem 3.1 assures us that, in theory, neural networks can approximate continuous functions on compact sets with arbitrary degrees of accuracy; but it does not supply us with an efficient, universally applicable algorithm to find the right number of hidden neurons a_1 appropriate parameters θ of for a neural network to approximate a given continuous function f . In fact, a priori finding a well-suited neural network architecture a for a given task is a non-trivial problem which is currently poorly understood from a theoretical point of view. In addition, tuning the parameters θ of the network typically leads to a high-dimensional non-convex optimization problem. Later in this thesis, we present the most prominent parameter optimization algorithm associated with neural networks: gradient descent via backpropagation. However, new algorithms to adress this challenge are constantly being developed.

Now that we have familiarized ourselves with the definition of neural network functions and explored some of their most important properties, we want to define a hypothesis class of neural networks which is in accordance with Definition 2.5 from the previous chapter. To achieve compactness, we fix the architecture a of the network and restrict the parameter space to a compact set.

Definition 3.6 (Hypothesis Class of Neural Networks). *Let*

- $d \in \mathbb{N}_{\geq 1}$ be a natural number (input dimension),
- $n \in \mathbb{N}_{\geq 1}$ be a natural number (output dimension),
- $D \subseteq \mathbb{R}^d$ be a compact set (neural network domain),
- $\rho \in \mathcal{C}(\mathbb{R}, \mathbb{R})$ be a continuous nonpolynomial function (activation function),
- $l \in \mathbb{N}_{\geq 1}$ be a natural number (number of hidden layers),
- $a := (a_0, \dots, a_{l+1}) \in \mathbb{N}_{\geq 1}^{l+2}$ be an $l+2$ -tuple of natural numbers fulfilling $(a_0, a_{l+1}) = (d, n)$ (neural network architecture), and
- $R > 0$ be a positive constant (bound on absolute parameter size).

Then the set

$$\begin{aligned} & \mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n) \\ & := \{\Phi_{\rho, l, a}(\theta_R, \cdot) : D \rightarrow \mathbb{R}^n \mid \theta_R \in [-R, R]^{s(a)}\}. \end{aligned} \quad (132)$$

is called hypothesis class of (classical fully-connected feedforward artificial) neural networks with hyperparameters ρ, l, a, R .

Proposition 3.1 (Compactness of Parameter-Bounded Neural Networks). *Assume the setting of Definition 3.6. Then $\mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n)$ is indeed a hypothesis class in the sense of Definition 2.5, i.e. $\mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n)$ is a compact subset of the Banach space $(\mathcal{C}(D, \mathbb{R}^n), \|\cdot\|_{\mathcal{C}})$.*

Proof. Clearly $\mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n)$ is non-empty. Moreover, every neural network $\Phi_{\rho, l, a}(\theta_R, \cdot) \in \mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n)$ is, by definition, a composition of continuous functions:

$$\begin{aligned} & \Phi_{\rho, l, a}(\theta_R, \cdot) \\ & = \varphi_{W^{(l)}(\theta_R), b^{(l)}(\theta_R)} \circ \rho_{a_l} \circ \varphi_{W^{(l-1)}(\theta_R), b^{(l-1)}(\theta_R)} \cdots \circ \rho_{a_1} \circ \varphi_{W^{(0)}(\theta_R), b^{(0)}(\theta_R)}. \end{aligned} \quad (133)$$

Therefore, $\Phi_{\rho, l, a}(\theta_R, \cdot)$ must itself be continuous. Here, $W^{(k)}$ and $b^{(k)}$ are the parameter maps implicitly associated with l and a as described in Basic Setting 3.1. In order to prove compactness of $\mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n) \subseteq \mathcal{C}(D, \mathbb{R}^n)$, we consider the map

$$j : [-R, R]^{s(a)} \rightarrow \mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n), \quad j(\theta_R)(\cdot) = \Phi_{\rho, l, a}(\theta_R, \cdot). \quad (134)$$

One can show, that j is continuous with respect to the topologies induced by the norm

$$\begin{aligned} & \|\cdot\|_{l, a} : \mathbb{R}^{s(a)} \rightarrow \mathbb{R}_{\geq 0}, \\ & \|\theta\|_{l, a} := \max_{k \in \{0, \dots, l\}} \max_{i \in \{1, \dots, a_{k+1}\}, j \in \{1, \dots, a_k\}} |W_{i, j}^{(k)}(\theta)| + \max_{k \in \{0, \dots, l\}} \max_{i \in \{1, \dots, a_{k+1}\}} |b_i^{(k)}(\theta)|, \end{aligned} \quad (135)$$

on the hypercube $[-R, R]^{s(a)}$ and by the maximum norm $\|\cdot\|_\infty$ on the vector space $\mathcal{C}(D, \mathbb{R}^n)$. For a detailed proof of the continuity of j , see Proposition 5.1 in the 2018 paper from Peterson et. al. about the topological properties of neural network spaces [32]. Now note, that all norms on finite dimensional vector spaces induce the same topology, which implies that $[-R, R]^{s(a)}$ must be compact in the topology induced by $\|\cdot\|_{l,a}$, since it is compact in the topology induced by the Euclidean norm. Because continuous functions map compact sets to compact sets and j is obviously surjective, the set

$$\mathcal{N}_{\rho,l,a,R}(D, \mathbb{R}^n) = j([-R, R]^{s(a)}) \quad (136)$$

must be compact. $\square$

3.2 Parameter Tuning via Backpropagation and Gradient Descent

We now turn our attention to the problem of minimizing the empirical error function over a hypothesis class of neural network functions. This step is known as *training* of the neural network and sophisticated training algorithms are key for the success of any machine learning endeavour. We start by returning to the classical setting of the statistical learning problem.

Setting 3.2. For the rest of Section 3, we again assume Setting 2.1 and Setting 2.2 from Section 2. Moreover, we demand $\mathcal{H} := \mathcal{N}_{\rho,l,a,R}(D, \mathbb{R}^n)$ to be a hypothesis class of neural network functions. To stay consistent with our solution plan for the statistical learning problem 2.8 and implicitly guarantee uniform convergence (and therefore usefulness) of the empirical error, we assume in addition, that there exists an $M > 0$ such that, for all $\Phi_{\rho,l,a}(\theta_R, \cdot) \in \mathcal{N}_{\rho,l,a,R}(D, \mathbb{R}^n)$, the inequality

$$\|\Phi_{\rho,l,a}(\theta_R, X) - Y\| \leq M \quad (137)$$

holds $\mathbb{P}$ -a.s.; this requirement, however, is not strictly necessary to give meaning to the statements in the rest of this section.

Following the solution strategy for the statistical learning problem summarised in Remark 2.8, the next (and essentially final) step is to find an algorithm for the minimization of the empirical error function over a fixed hypothesis class of neural networks. We first simplify the notation for the empirical error and emphasize its dependence on the parameter value θ_R by introducing a *loss function*. For the next definition, recall, that

$$z_1 = (x_1, y_1), \dots, z_m = (x_m, y_m) \in D \times \mathbb{R}^n \quad (138)$$

refer to m fixed i.i.d. observations of (X, Y) .

Definition 3.7 (Loss Function). *From now on, we denote the empirical error as*

$$\tilde{\mathcal{E}}_{z(m)}(\Phi_{\rho,l,a}(\theta_R, \cdot)) = \frac{1}{m} \sum_{i=1}^m \|\Phi_{\rho,l,a}(\theta_R, x_i) - y_i\|^2 := \frac{1}{m} \sum_{i=1}^m \mathcal{V}_{\rho,l,a,z_i}(\theta_R) \quad (139)$$

with

$$\mathcal{V}_{\rho,l,a,z_i} : \mathbb{R}^{s(a)} \rightarrow [0, \infty), \quad \mathcal{V}_{\rho,l,a,z_i}(\theta) = \|\Phi_{\rho,l,a}(\theta, x_i) - y_i\|^2. \quad (140)$$

The map $\mathcal{V}_{\rho,l,a,z_i}$ is called (quadratic) loss function.

We are now interested in finding elements of the (non-empty) set

$$\arg \min_{\theta_R \in [-R, R]^{s(a)}} \tilde{\mathcal{E}}_{z^{(m)}}(\Phi_{\rho,l,a}(\theta_R, \cdot)) = \arg \min_{\theta_R \in [-R, R]^{s(a)}} \frac{1}{m} \sum_{i=1}^m \mathcal{V}_{\rho,l,a,z_i}(\theta_R). \quad (141)$$

The classical approach to solve this problem is via *stochastic gradient descent* minimization. We first introduce regular (non-stochastic) gradient descent.

Definition 3.8 (Gradient Descent Minimization). *Let $s \in \mathbb{N}_{\geq 1}$ be a natural number and $E \in \mathcal{C}^1(\mathbb{R}^s, \mathbb{R})$ be a continuously differentiable function. Furthermore, choose $x_0 \in \mathbb{R}^s$ and let $\Gamma := (\gamma_k)_{k \in \mathbb{N}_{\geq 0}}$ be a sequence of strictly positive numbers in $\mathbb{R}_{>0}$. Then the sequence $(x_k^{(E, x_0, \Gamma)})_{k \in \mathbb{N}_{\geq 0}}$ in $\mathbb{R}^s$, defined by*

$$x_0^{(E, x_0, \Gamma)} = x_0, \quad x_{k+1}^{(E, x_0, \Gamma)} = x_k^{(E, x_0, \Gamma)} - \gamma_k(\nabla E)(x_k^{(E, x_0, \Gamma)}), \quad (142)$$

is called *gradient descent sequence of E with starting point x_0 , stepsizes $\Gamma = (\gamma_k)_{k \in \mathbb{N}_{\geq 0}}$ and gradient descent function values*

$$(E_k^{x_0, \Gamma})_{k \in \mathbb{N}_{\geq 0}} := (E(x_k^{(E, x_0, \Gamma)}))_{k \in \mathbb{N}_{\geq 0}}. \quad (143)$$

Note, that for $s \in \mathbb{N}_{\geq 0}$, $E \in \mathcal{C}^1(\mathbb{R}^s, \mathbb{R})$ and $x_0 \in \mathbb{R}^s$, the vector $(\nabla E)(x_0) \in \mathbb{R}^s$ either vanishes or points into the direction of steepest decent of E when moving away from x_0 ; phrased differently, E decreases fastest, if one goes from x_0 in direction of $(\nabla E)(x_0)$. In particular, for small enough $\gamma > 0$, it always holds that

$$E(x_0) \geq E(x_0 - \gamma(\nabla E)(x_0)). \quad (144)$$

This implies, that, for a choice of small-enough stepsizes $\Gamma = (\gamma_k)_{k \in \mathbb{N}_{\geq 0}}$, the sequence of gradient descent function values $(E_k^{x_0, \Gamma})_{k \in \mathbb{N}_{\geq 0}}$ is monotonically decreasing; in this sense, Equation 145 defines a recursive minimization algorithm for E . Under some additional regularity assumptions on E , such as boundedness of E from below and Lipschitz continuity of ∇E , as well as sufficiently small stepsizes Γ , convergence of $(x_k^{(E, x_0, \Gamma)})_{k \in \mathbb{N}_{\geq 0}}$ to a critical point $x_c \in \mathbb{R}^s$ with $(\nabla E)(x_c) = 0$ can be guaranteed [33] [34].

Definition 3.9 (Projected Gradient Descent). *Let $R > 0$ and assume the setting of the definition of gradient descent optimization 3.8. If one wants to constrain the gradient descent sequence to a hypercube of the form $C := [-R, R]^s$, then one can choose $x_0 \in C$ and modify Equation 145 in the following way:*

$$x_0^{(E, x_0, \Gamma)} = x_0, \quad x_{k+1}^{(E, x_0, \Gamma)} = P_C[x_k^{(E, x_0, \Gamma)} - \gamma_k(\nabla E)(x_k^{(E, x_0, \Gamma)})]. \quad (145)$$

Here $P_C : \mathbb{R}^s \rightarrow C$ is the projection in $\mathbb{R}^s$ onto C such that, for all $x \in \mathbb{R}^s$, the point $P(x) \in C$ is the element in C which is closest to x ¹². This minimization method is known as *projected gradient descent* [35].

To apply the gradient descent algorithm for neural network parameter tuning, we first observe that

$$\nabla_{\theta_R} \tilde{\mathcal{E}}_{z^{(m)}}(\Phi_{\rho,l,a}(\theta_R, \cdot)) = \frac{1}{m} \sum_{i=1}^m \nabla_{\theta_R} (\mathcal{V}_{\rho,l,a,z_i})(\theta_R). \quad (146)$$

It is therefore sufficient to compute $(\nabla_{\theta_R} \mathcal{V}_{\rho,l,a,z_i})(\theta_R)$ for all $i \in \{1, \dots, m\}$ and then take the average.

The standard algorithm for the computation of the gradient of the loss function $(\nabla_{\theta_R} \mathcal{V}_{\rho,l,a,z_i})(\theta_R)$ is known as *backpropagation*. The backpropagation algorithm was developed by multiple independent researchers in the 1960's [36] and is essentially based on the recursive application of the multivariate chain rule of classical calculus. The easiest and most straightforward way to understand the working mechanisms of backpropagation is by studying a simple, yet non-trivial example in detail.

Example 3.1 (Backpropagation Algorithm). Let $\rho \in \mathcal{C}^1(\mathbb{R}, \mathbb{R})$, $l = 1$, $a = (2, 2, 2)$, and $z = (x, y) \in D \times \mathbb{R}^2$ with compact $D \subset \mathbb{R}^2$. We want to compute the gradient of the continuously differentiable function

$$\mathcal{V}_{\rho,l,a,z} : \mathbb{R}^{s(a)} \rightarrow [0, \infty), \quad \mathcal{V}_{\rho,l,a,z}(\theta) = Q_y(\Phi_{\rho,l,a}(\theta, x)) \quad (147)$$

with

$$Q_y : \mathbb{R}^2 \rightarrow [0, \infty), \quad Q_y(\Phi) = \|\Phi - y\|^2. \quad (148)$$

We can see, that $\mathcal{V}_{\rho,l,a,z}$ is associated with a neural network $\Phi_{\rho,l,a}(\theta, \cdot)$, whose architecture is depicted in Figure 2.

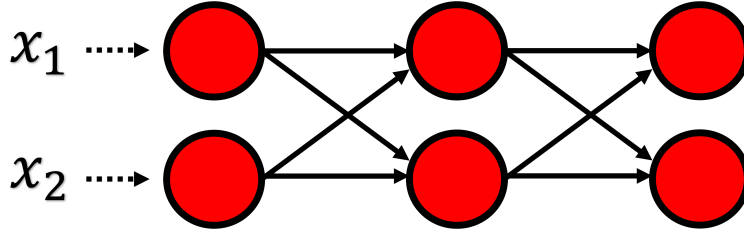


Figure 2: Neural network with $l = 1$ hidden layer and an architecture of $a = (2, 2, 2)$.

Note, that, under our assumptions, the size of $\Phi_{\rho,l,a}(\theta, x)$ is

$$s(a) = 2 * 2 + 2 + 2 * 2 + 2 = 12, \quad (149)$$

¹²Since C is convex, it is not hard to show that, for all $x \in \mathbb{R}^s$, the point $P(x) \in \mathbb{R}^s$ exists and is unique.

i.e. the parameter vector $\theta = (\theta_1, \dots, \theta_{12})$ is of dimension 12. Using the parameter maps

$$W^{(0)}(\theta) = \begin{pmatrix} \theta_1 & \theta_3 \\ \theta_2 & \theta_4 \end{pmatrix}, \quad b^{(0)}(\theta) = \begin{pmatrix} \theta_5 \\ \theta_6 \end{pmatrix}, \quad (150)$$

$$W^{(1)}(\theta) = \begin{pmatrix} \theta_7 & \theta_9 \\ \theta_8 & \theta_{10} \end{pmatrix}, \quad b^{(1)}(\theta) = \begin{pmatrix} \theta_{11} \\ \theta_{12} \end{pmatrix}, \quad (151)$$

we can write

$$\Phi_{\rho,l,a}(\theta, x) = W^{(1)}(\theta)\rho_2(W^{(0)}(\theta)x + b^{(0)}(\theta)) + b^{(1)}(\theta). \quad (152)$$

To determine $\nabla_{\theta} \mathcal{V}_{\rho,l,a,z}$, our plan is to compute all partial derivatives $\frac{\partial \mathcal{V}_{\rho,l,a,z}}{\partial \theta_i}$ in backwards order: we start with the parameters $\theta_{b^{(1)}} := b^{(1)}(\theta)$, then $\theta_{W^{(1)}} := \text{vec}(W^{(1)}(\theta))$, then $\theta_{b^{(0)}} := b^{(0)}(\theta)$, and finally $\theta_{W^{(0)}} := \text{vec}(W^{(0)}(\theta))$.

Denote with O_0, I_1, O_1, I_2 the outputs and inputs of layers 0, 1 and 2. In particular, $O_0 = x$ and $I_2 = \Phi_{\rho,l,a}(\theta, x)$. Depending on which components of θ are seen as constant, one can interpret I_i and O_i as functions of the vectors $\theta_{W^{(0)}}, \theta_{b^{(0)}}, \theta_{W^{(1)}}$ and $\theta_{b^{(1)}}$. Similarly, one can interpret I_i, O_i as functions of each other, e.g. $O_1(I_1) = \sigma_2(I_1)$. By abuse of notation, we use the same symbols I_i and O_i for these functions in both situations and their identity is made clear by the notation of their argument, e.g. $I_2(\theta_{b^{(1)}})$ is a function describing the change of I_2 with $\theta_{b^{(1)}}$ (while all other parameters are held constant) and, analogously, $I_2(\theta_{W^{(1)}})$ is a (technically different) function describing the change of I_2 with $\theta_{W^{(1)}}$. Lastly, $I_2(O_1)$ denotes a third function describing the change of I_2 with O_1 if all parameters θ are held constant.

Note, that we can split up the gradient of $\mathcal{V}_{\rho,l,a,z}$ into a concatenation of smaller gradient vectors associated with the weights and biases of each layer of our neural network:

$$\nabla_{\theta} \mathcal{V}_{\rho,l,a,z} = \nabla_{\theta_{W^{(0)}}} \mathcal{V}_{\rho,l,a,z}^{W^{(0)}} \frown \nabla_{\theta_{b^{(0)}}} \mathcal{V}_{\rho,l,a,z}^{b^{(0)}} \frown \nabla_{\theta_{W^{(1)}}} \mathcal{V}_{\rho,l,a,z}^{W^{(1)}} \frown \nabla_{\theta_{b^{(1)}}} \mathcal{V}_{\rho,l,a,z}^{b^{(1)}}. \quad (153)$$

with

$$\begin{aligned} \mathcal{V}_{\rho,l,a,z}^{b^{(1)}} : \mathbb{R}^2 &\rightarrow [0, \infty), & \mathcal{V}_{\rho,l,a,z}^{b^{(1)}}(\theta_{b^{(1)}}) &= Q_y(I_2(\theta_{b^{(1)}})), \\ \mathcal{V}_{\rho,l,a,z}^{W^{(1)}} : \mathbb{R}^4 &\rightarrow [0, \infty), & \mathcal{V}_{\rho,l,a,z}^{W^{(1)}}(\theta_{W^{(1)}}) &= Q_y(I_2(\theta_{W^{(1)}})), \\ \mathcal{V}_{\rho,l,a,z}^{b^{(0)}} : \mathbb{R}^2 &\rightarrow [0, \infty), & \mathcal{V}_{\rho,l,a,z}^{b^{(0)}}(\theta_{b^{(0)}}) &= Q_y(I_2(O_1(I_1(\theta_{b^{(0)}})))), \\ \mathcal{V}_{\rho,l,a,z}^{W^{(0)}} : \mathbb{R}^4 &\rightarrow [0, \infty), & \mathcal{V}_{\rho,l,a,z}^{W^{(0)}}(\theta_{W^{(0)}}) &= Q_y(I_2(O_1(I_1(\theta_{W^{(0)}})))). \end{aligned} \quad (154)$$

Now denote with $J_f(x_0)$ the Jacobian matrix of a continuously differentiable function f at a point x_0 . The form of the functions defined in 154 along with Equation 153 allows us to calculate $\nabla_{\theta} \mathcal{V}_{\rho,l,a,z}$ by making use of the multivariate

chain rule. We start with the gradients associated with $b^{(1)}$ and $W^{(1)}$:

$$\begin{aligned}
[(\nabla_{\theta_{b^{(1)}}} \mathcal{V}_{\rho,l,a,z}^{b^{(1)}})(\theta_{b^{(1)}})]^\top &= J_{\mathcal{V}_{\rho,l,a,z}^{b^{(1)}}}(\theta_{b^{(1)}}) = J_{Q_y \circ I_2}(\theta_{b^{(1)}}) \\
&= J_{Q_y}(I_2(\theta_{b^{(1)}}))J_{I_2}(\theta_{b^{(1)}}) \\
&= \begin{pmatrix} 2((I_2(\theta_{b^{(1)}}))_1 - y_1) & 2((I_2(\theta_{b^{(1)}}))_2 - y_2) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \\
[(\nabla_{\theta_{W^{(1)}}} \mathcal{V}_{\rho,l,a,z}^{W^{(1)}})(\theta_{W^{(1)}})]^\top &= J_{\mathcal{V}_{\rho,l,a,z}^{W^{(1)}}}(\theta_{W^{(1)}}) = J_{Q_y \circ I_2}(\theta_{W^{(1)}}) \\
&= J_{Q_y}(I_2(\theta_{W^{(1)}}))J_{I_2}(\theta_{W^{(1)}}) \\
&= \begin{pmatrix} 2((I_2(\theta_{W^{(1)}}))_1 - y_1) & 2((I_2(\theta_{W^{(1)}}))_2 - y_2) \end{pmatrix} \begin{pmatrix} O_{1,1} & 0 & O_{1,2} & 0 \\ 0 & O_{1,1} & 0 & O_{1,2} \end{pmatrix}
\end{aligned} \tag{155}$$

Note that the 2×1 -matrices

$$J_{Q_y}(I_2(\theta_{b^{(1)}})) = \begin{pmatrix} 2((I_2(\theta_{b^{(1)}}))_1 - y_1) & 2((I_2(\theta_{b^{(1)}}))_2 - y_2) \end{pmatrix} \tag{156}$$

and

$$J_{Q_y}(I_2(\theta_{W^{(1)}})) = \begin{pmatrix} 2((I_2(\theta_{W^{(1)}}))_1 - y_1) & 2((I_2(\theta_{W^{(1)}}))_2 - y_2) \end{pmatrix} \tag{157}$$

actually have the exact same entries, and therefore need to be computed only once. To simplify our notation further, we write

$$J_{Q_y}(I_2(\theta_{b^{(1)}})) = J_{Q_y}(I_2(\theta_{W^{(1)}})) := J_{Q_y}(I_2) = \begin{pmatrix} 2((I_2)_1 - y_1) & 2((I_2)_2 - y_2) \end{pmatrix} \tag{158}$$

From now on, we use an analogous notation for identical Jacobi matrices in equivalent situations. With this in mind, we can continue to use the multivariate chain rule to recursively compute all four components of our gradient vector:

$$\begin{aligned}
[(\nabla_{\theta_{b^{(1)}}} \mathcal{V}_{\rho,l,a,z}^{b^{(1)}})(\theta_{b^{(1)}})]^\top &= J_{Q_y}(I_2)J_{I_2}(\theta_{b^{(1)}}) \\
[(\nabla_{\theta_{W^{(1)}}} \mathcal{V}_{\rho,l,a,z}^{W^{(1)}})(\theta_{W^{(1)}})]^\top &= J_{Q_y}(I_2)J_{I_2}(\theta_{W^{(1)}}) \\
[(\nabla_{\theta_{b^{(0)}}} \mathcal{V}_{\rho,l,a,z}^{b^{(0)}})(\theta_{b^{(0)}})]^\top &= J_{Q_y}(I_2)J_{I_2}(O_1)J_{O_1}(I_1)J_{I_1}(\theta_{b^{(0)}}) \\
[(\nabla_{\theta_{W^{(0)}}} \mathcal{V}_{\rho,l,a,z}^{W^{(0)}})(\theta_{W^{(0)}})]^\top &= J_{Q_y}(I_2)J_{I_2}(O_1)J_{O_1}(I_1)J_{I_1}(\theta_{W^{(0)}})
\end{aligned} \tag{159}$$

In this sense, we can "backpropagate" the derivative of the error through the neural network by prolonging a chain of Jacobi matrices with each "jump" to a previous layer, and eventually compute $\nabla_{\theta} \mathcal{V}_{\rho,l,a,z}$.

It is clear, that the backpropagation method presented in Example 3.1 readily generalizes to arbitrary hyperparameters l and a . For a more detailed technical description of backpropagation for arbitrary network architectures, consider the book of Bishop [24] or the paper by Rumelhart et. al. [37].

Remark 3.6 (Non-differentiable Activation Function). If ρ is continuous and differentiable everywhere except for a finite number of points $p_1, \dots, p_k$ (as for

example is the case if $\rho(x) = \text{relu}(x)$) one can formally replace the gradient in our optimization method with a subgradient [38] and use *subgradient descent* as an alternative algorithm. The subgradient is a natural generalization of the gradient for non-differentiable functions. Each gradient of a function at a point (if it exists) is automatically a subgradient, but the converse does not hold. Since the input of a neuron rarely coincides exactly with a non-differentiable point p_i , the subgradient method is basically identical to gradient descent in most applications from a practical point of view.

Unfortunately, gradient descent optimization of neural networks, can come with a relatively heavy computational cost since it scales with the sample size m ; every gradient update requires the computation of m loss-function-gradients $\nabla_{\theta} \mathcal{V}_{\rho,l,a,z_i}$. Each backpropagation step associated with one gradient $\nabla_{\theta} \mathcal{V}_{\rho,l,a,z_i}$, on the other hand, requires an asymptotic computational effort of $\mathcal{O}(s(a))$ [24], meaning, that the cost for one backpropagation step scales linearly with the number of network parameters $s(a)$. This implies a total asymptotic complexity of neural network gradient descent via backpropagation of $\mathcal{O}(ms(a))$. A common way to reduce this degree of complexity is by only computing the error-gradient of a small *batch* of size $\beta < m$ drawn uniformly from the training set $z^{(m)}$ instead of using the full training set in each iteration. This leads us to the formal definition of the (mini-batch) stochastic gradient descent algorithm, which is seen as the prototypical method for neural network parameter tuning.

Definition 3.10 (Stochastic Gradient Descent). *For a neural network $\Phi_{\rho,l,a}(\theta, \cdot)$ and a training set $z^{(m)}$, define a sequence of parameter updates*

$$\theta_0, \theta_1, \theta_2, \dots \in \mathbb{R}^{s(a)} \quad (160)$$

via the following procedure:

- Choose a natural number $\beta \in \{1, \dots, m\}$ (referred to as *batch size*).
- Choose a sequence of strictly positive numbers $\Gamma = (\gamma_k)_{k \in \mathbb{N}_{\geq 0}}$ (referred to as *step sizes*).
- Set $\theta_0 := \theta$.
- For $k = 0, 1, 2, \dots$, choose some index set of size β

$$I_k^{(\beta)} := \{I_{k,1}^{(\beta)}, \dots, I_{k,\beta}^{(\beta)}\} \subseteq \{1, \dots, m\} \quad (161)$$

uniformly at random from the set $\{I \subseteq \{1, \dots, m\} \mid |I| = \beta\}$ and set

$$\theta_{k+1} := \theta_k - \frac{\gamma_k}{\beta} \sum_{i=1}^{\beta} (\nabla_{\theta} \mathcal{V}_{\rho,l,a,z_{I_{k,i}^{(\beta)}}})(\theta_k). \quad (162)$$

The sequence $(\theta_k)_{k \in \mathbb{N}_{\geq 0}}$ is called stochastic gradient descent sequence of $\Phi_{\rho,l,a}(\theta, \cdot)$ with batch size β and step sizes Γ .

Remark 3.7 (Properties of Stochastic Gradient Descent). For a batch size of $\beta = m$, the sequence $(\theta_k)_{k \in \mathbb{N}_{\geq 0}}$ from Definition 3.10 reduces to a regular gradient descent sequence. Furthermore, it is easy to show, that for $\beta < m$, the sum expression from Equation 162 provides an unbiased estimator of the actual gradient of the empirical error at θ_k :

$$\mathbb{E} \left[\frac{1}{\beta} \sum_{i=1}^{\beta} (\nabla_{\theta} \mathcal{V}_{\rho, l, a, z_{I_{k,i}^{(\beta)}}})(\theta_k) \right] = \frac{1}{m} \sum_{i=1}^m (\nabla_{\theta} \mathcal{V}_{\rho, l, a, z_i})(\theta_k) = (\nabla_{\theta} \tilde{\mathcal{E}}_{z^{(m)}}(\Phi_{\rho, l, a}(\theta, \cdot)))(\theta_k) \quad (163)$$

In this sense, stochastic gradient descent for neural networks is both a generalization and computationally efficient approximation of regular gradient descent. Note, that the execution of the update rule in Equation 162 requires only β backpropagation computations. This implies, that, unlike in regular gradient descent, the computational complexity of the algorithm does not change with increasing sample size m . In practise, often many training points in $z^{(m)}$ are somewhat similar, which means that a stochastic mini-batch of sufficient size is usually capable of catching the basic characteristics of the whole training set.

Remark 3.8 (Staying within the hypothesis class). If one wants to remain in a fixed predefined hypothesis class of neural networks $\mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n)$ during training, one can modify the stochastic gradient descent algorithm given in Definition 3.10 and use *projected* stochastic gradient descent instead 3.9 in order to guarantee, that the sequence of parameter updates $(\theta_k)_{k \in \mathbb{N}_{\geq 0}}$ stays within the hypercube $[R, R]^{s(a)}$.

This, however, is almost never done in practise, since an a priori constraint on the parameter space is rarely useful and, more often than not, detrimental. Usually, one does not define $R > 0$ in advance, but rather starts by initializing some reasonably small value $\theta_0 \in \mathbb{R}^{s(a)}$ and then minimizing $\tilde{\mathcal{E}}_{z^{(m)}}(\Phi_{\rho, l, a}(\theta_0, \cdot))$ with unconstrained $\theta_k \in \mathbb{R}^{s(a)}$. This is done, until $(\theta_k)_{k \in \mathbb{N}_{\geq 0}}$ has converged to some $\hat{\theta} \in \mathbb{R}^{s(a)}$, or shows some other type of bounded (possibly oscillating) behaviour. If it is necessary for formal or theoretical reasons, one can then afterwards tacitly choose $R > 0$ in a such a way that all accumulation points of $(\theta_k)_{k \in \mathbb{N}_{\geq 0}}$ are contained in $[-R, R]^{s(a)}$.

3.3 Refinements of Vanilla Training Methods.

There are countless methods to further improve the standard training method for neural networks based on stochastic gradient descent via backpropagation. Even an exhaustive description of only the most famous modifications of the training algorithm would far exceed the scope of this thesis. However, we still mention a selected number of important training techniques which are commonly used in state-of-the-art deep learning and point towards relevant literature for the interested reader:

- **Adam optimization.** *Adam optimization* [39] is another effective first-order gradient-based optimization method for objective functions and is

often used in combination with neural networks to replace stochastic gradient descent. In Adam optimization, at each step a mini-batch from the training set is selected uniformly at random; then the parameter vector θ_k is updated by subtracting an estimator for the first moment of the (stochastic) mini-batch gradient multiplied with a step-size depending on an estimator for the second moment of the mini-batch gradient.

- **Efficient parameter initialization.** Clever parameter initialization schemes, such as Xavier initialization [40], have been shown to speed up training and reduce the chance of getting stuck early in suboptimal minima. The idea of many parameter initialization schemes is to guarantee, that the size of the input of each neuron starts off within reasonable upper and lower bounds in order to avoid "saturated nonlinearities" in the activation functions, which tend to promote vanishing gradients.
- **Batch normalization.** *Batch Normalization* [41] modifies the neural network architecture in such a way that the input vector of each layer (including the hidden layers) is dynamically (z-score) normalized componentwise during computation. Empirical results show, that batch normalization can lead to better learning results, significantly speed up training by up to 1 order of magnitude and can make the network more resilient against suboptimal hyperparameters, learning rates and parameter initialization schemes. Possible reasons for these improvements include the control of the layer's input distributions during training (reduction of the *internal covariate shift*) and a significant increase in the smoothness of the parameter optimization landscape [42].
- **Mitigation of the vanishing gradient problem.** Assume, that we have an activation functions ρ with $\rho'(x) \in (0, 1)$, such as the sigmoid function $\text{sigm}(x) = \frac{1}{1+e^{-x}}$. In this case, sufficiently deep neural networks can exhibit *vanishing gradients* during training. The partial derivatives of $\tilde{\mathcal{E}}_{z^{(m)}}(\Phi_{\rho,l,a}(\theta, \cdot))$ with respect to the parameters associated with the early layers of the network tend to become extremely small, which prevents all but the very last layers to be effectively updated via gradient descent. The algebraic reason for this is, that the backpropagation algorithm requires consecutive multiplication with factors of the form $\rho'(x) \in (0, 1)$, causing the partial derivatives to shrink for parameters "far away" from the output layer. The most common way to adress the vanishing gradient problem is by using $\text{relu}(x) = \max(\{0, x\})$ as an activation function [43]. The fact, that the derivative $\text{relu}'(x) \in \{0, 1\}$ for all $x \in \mathbb{R} \setminus \{0\}$, makes it possible to train considerably deeper networks.
- **Gradient clipping.** The error surface in the parameter space of deep neural networks tends to have very steep cliffs, often causing the nearby gradients to explode and overshoot the cliff in a gradient descent setting. A straightforward solution to avoid this problem is *gradient clipping* (see [23], Section 10.11.1). In the simplest case, gradient clipping just means

to choose a maximum gradient length $g > 0$ and then, right before the gradient descent update, "clip" the mini-batch gradient

$$\frac{1}{\beta} \sum_{i=1}^{\beta} (\nabla_{\theta} \mathcal{V}_{\rho, l, a, z_{I_k^{(\beta)}, i}})(\theta_k) := (\nabla \tilde{\mathcal{E}}_{I_k^{(\beta)}})(\theta_k) \quad (164)$$

via

$$(\nabla \tilde{\mathcal{E}}_{I_k^{(\beta)}})(\theta_k) \rightarrow \frac{(\nabla \tilde{\mathcal{E}}_{I_k^{(\beta)}})(\theta_k)}{\|(\nabla \tilde{\mathcal{E}}_{I_k^{(\beta)}})(\theta_k)\|} g \quad (165)$$

if $\|(\nabla \tilde{\mathcal{E}}_{I_k^{(\beta)}})(\theta_k)\| > g$.

- **Regularization via Dropout.** Recall our discussion of the bias-variance tradeoff in Remark 2.7. Machine learning models with a large number of parameters tend to *overfit* during training due to increased variance, effectively learning the noise in the data along with the actual pattern of interest. Merely reducing the number of parameters, however, does not always solve the problem, since the smaller network can get too inflexible and *underfit* the available data due to increased bias. Technical modifications at varying stages of a machine learning algorithm with a fixed hypothesis class $\mathcal{H}$ in order to decrease the variance *without* substantially increasing the bias are called regularization methods.

A popular and effective regularization methods for neural networks is called *dropout* [44]. At each gradient descent iteration step, the dropout algorithm randomly drops neurons (along with their connections) with probability $p \in (0, 1)$ and only updates the parameters of the resulting "thinned" network. After training, the weights of the full network (without dropout of neurons) are multiplied by p to approximate averaging over all pre-trained thinned networks.

- **Hyperparameter-optimization.** Various methods exist to support the search for a suitable hypothesis space $\mathcal{N}_{\rho, l, a, R}(D, \mathbb{R}^n)$ over which to minimize the empirical error, usually by optimizing the discrete hyperparameters l and a , which determine the architecture of the neural network. Often, this is simply done by performing an exhaustive grid search over a preselected set of pairs of the form (l, a) . More sophisticated methods for hyperparameter tuning include Bayesian optimization [45] and genetic algorithms [46].

Remark 3.9 (Derivative-free Optimization Methods for Neural Networks). Besides derivative-based optimization methods, other noteworthy approaches for the parameter tuning of neural networks have been investigated in the literature. While most of these methods have so far fallen short compared to gradient techniques in terms of performance, others have been shown to be superior in certain settings.

Alternative (derivative-free) optimization approaches for neural networks include *particle swarm optimization* [47] [48] and *genetic algorithms* [49] [50] [51]. In particle swarm optimization, a colony of individual points move through the search space according to their own prior experience and the current experience of their neighbours. Genetic algorithms, on the other hand, try to model the optimization process of natural selection, by producing a family of parameter candidates, assigning a fitness to those candidates, killing of all but the fittest "individuals", letting the survivors breed to mix their properties, mutate the resulting offspring to introduce some stochastic flexibility, and then iterate this cycle.

4 Solving Kolmogorov Equations with Variable Coefficient Parameters via Statistical Learning

4.1 Goal and Setting

In this chapter, we apply the theoretical framework from Chapter 2 to suggest (and partially prove) a formal theorem, which describes, how the solution functions of an important family of partial differential equations, *Kolmogorov equations*, can be expressed as regression functions of appropriately chosen mathematical learning problems¹³. We do this by exploiting the *Feynman-Kac formula*, which establishes a link between (deterministic) Kolmogorov equations and certain types of stochastic differential equations. This potentially opens the door for the numerical solution of parabolic partial differential equation by means of statistical machine learning.

The results in this section are a direct generalization attempt of the work of Berner, Grohs, Jentzen et. al. [10] [11] [9]. They showed, how neural networks along with synthetic training data can be used to numerically solve a given Kolmogorov equation with predefined coefficient functions σ, μ on a predefined hypercube $[v, w]^d$ and at a predefined time $T > 0$. Furthermore, they delivered a theoretical proof, that this solution method indeed overcomes the *curse of dimensionality*¹⁴. This represents a crucial and rare advantage over classical finite difference and finite element methods, all of which typically cannot be used in high dimensions due to the resulting exponential explosion of required computational resources. The classical way to circumvent the curse of dimensionality has been the employment of stochastic Monte-Carlo based methods; these techniques, however, are only suitable to compute solution points of the form $u(T, x)$ at a fixed predefined time-space point (T, x) , limiting their usefulness in practise. The method introduced by Berner, Grohs, Jentzen et. al., however, can compute the global solution $u(T, x)$ of a Kolmogorov PDE at time $T > 0$ on an entire region of the form $[v, w]^d$ without incurring the curse of dimensionality.

We now aim to take the results of Berner, Grohs and Jentzen one step further. Our proposed learning method is based on the simulation of training data, which not only contains information about a particular solution function u of a particular Kolmogorov PDE on a compact hypercube $[v, w]^d$, but also varying information about the time of solution $t \in [0, T]$ and the initial coefficients σ, μ of the Kolmogorov PDE associated with u . In other words: we modify the method of Berner, Grohs and Jentzen in a way which allows us to treat the coefficients σ, μ of our Kolmogorovs equation and the time of solution $T > 0$

¹³In the next section, our aim will be to empirically test the correctness of our main theorem via numerical simulations.

¹⁴A numerical approximation algorithm for a Kolmogorov PDE is said to suffer the curse of dimensionality if its required computational effort grows exponentially in the dimension of the PDE.

not as fixed, but as variable training data, which can be fed into a new learning algorithm. If successful, our approach allows us to teach a single model the numerical solution functions u of a whole family of Kolmogorov equations with varying coefficient functions $\sigma(x), \mu(x)$ at varying times $t \in [0, T]$ on a predefined, fixed hypercube $[v, w]^d$. In this section of the thesis, we try to give a rigorous formalization and partial mathematical justification of the theoretical ideas underlying our empirical approach.

We start by introducing our basic setting for this task and then discuss the family of equations which we are interested in.

Setting 4.1. For the rest of Section 4,

- let $T \in (0, \infty)$ be a positive constant,
- let $d \in \mathbb{N}_{\geq 1}$ be a natural number,
- let $\mathbb{R}^d$ be equipped with the Borel σ -algebra $\mathcal{B}(\mathbb{R}^d)$, the standard scalar product

$$\langle \cdot, \cdot \rangle : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}, \quad \langle v, w \rangle = \sum_{i=1}^d v_i w_i, \quad (166)$$

and the associated Euclidean norm

$$\|\cdot\| : \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}, \quad \|v\| = \langle v, v \rangle^{\frac{1}{2}}, \quad (167)$$

- let $\mathbb{S}_{\tilde{d} \times \tilde{d}}(\mathbb{R})$ be the set of symmetric square matrices with real entries for each size $\tilde{d} \in \mathbb{N}_{\geq 1}$,
- let the vector space of matrices $\mathbb{R}^{d \times d}$ be equipped with the Frobenius norm

$$\|\cdot\| : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}_{\geq 0}, \quad \|A\| = \left(\sum_{i,j=1}^d A_{i,j}^2 \right)^{\frac{1}{2}}, \quad (168)$$

- let the functions

$$\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}, \quad (169)$$

$$\mu : \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad (170)$$

be affine linear transformations, i.e. let there be $d+1$ matrices

$$\pi_\sigma := (A_1^{(\sigma)}, \dots, A_d^{(\sigma)}, \bar{A}^{(\sigma)}) \in (\mathbb{R}^{d \times d})^{d+1} \quad (171)$$

such that, for all $x \in \mathbb{R}^d$,

$$\sigma(x) = (A_1^{(\sigma)}x \mid \dots \mid A_d^{(\sigma)}x) + \bar{A}^{(\sigma)}, \quad (172)$$

where $A_i^{(\sigma)}x$ describes the i -th column of the matrix $\sigma(x) \in \mathbb{R}^{d \times d}$, and let there be a matrix-vector pair

$$\pi_\mu := (A^{(\mu)}, \bar{a}^{(\mu)}) \in \mathbb{R}^{d \times d} \times \mathbb{R}^d \quad (173)$$

such that, for all $x \in \mathbb{R}^d$,

$$\mu(x) = A^{(\mu)}x + \bar{a}^{(\mu)}, \quad (174)$$

and, finally,

- let the function

$$\varphi : \mathbb{R}^d \rightarrow \mathbb{R} \quad (175)$$

be continuous and at most polynomially growing, i.e. let φ be continuous and let there be a constant $C \in [1, \infty)$ such that, for all $x \in \mathbb{R}^d$ it holds that

$$|\varphi(x)| \leq C(1 + \|x\|)^C. \quad (176)$$

4.2 Kolmogorov Partial Differential Equations and Viscosity Solutions

In this subsection, we introduce Kolmogorov equations and the notion of *viscosity solutions* for partial differential equations. Towards the end, we answer questions associated with the existence and uniqueness of viscosity solutions for Kolmogorov PDEs. As indicated further down in this section, viscosity solutions represent the appropriate type of weak solution concept for Kolmogorov equations (and several other types of PDEs).

Definition 4.1 (Kolmogorov Partial Differential Equation). *The equation*

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0, \quad (177)$$

$$u(0, x) = \varphi(x), \quad (178)$$

is called Kolmogorov partial differential equation, Kolmogorov equation or Kolmogorov PDE. The functions σ and μ are referred to as coefficient functions (or coefficients) and the function φ is referred to as the boundary condition, initial condition (or initial function) associated with the Kolmogorov equation.

Remark 4.1 (First Properties of Kolmogorov PDEs). It is easy to show, that all Kolmogorov equations are linear, second-order, homogeneous partial differential equations of parabolic type. An explicit, coordinate-based expression for Kolmogorov PDEs is given by

$$\frac{\partial u}{\partial t}(t, x) - \sum_{i,j=1}^d \bar{\sigma}_{i,j}(x) \frac{\partial^2 u}{\partial x_i \partial x_j}(t, x) - \sum_{i=1}^d \mu_i(x) \frac{\partial u}{\partial x_i}(t, x) = 0, \quad (179)$$

$$u(0, x) = \varphi(x), \quad (180)$$

with

$$\bar{\sigma}_{i,j}(x) := \sum_{l=1}^d \frac{\sigma_{i,l}(x) \sigma_{j,l}(x)}{2}. \quad (181)$$

Definition 4.2 (Classical Solutions of Kolmogorov PDEs). *A classical solution function for the Kolmogorov PDE (without boundary condition)*

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0 \quad (182)$$

on the open set $(0, T) \times \mathbb{R}^d$ is defined as a function ¹⁵

$$u^{(\pi_\sigma, \pi_\mu)} : (0, T) \times \mathbb{R}^d \rightarrow \mathbb{R} \quad (183)$$

with the following properties:

- $u^{(\pi_\sigma, \pi_\mu)} \in \mathcal{C}^{1,2}((0, T) \times \mathbb{R}^d, \mathbb{R})$, i.e. $u^{(\pi_\sigma, \pi_\mu)}(t, x)$ is once continuously differentiable in t and twice continuously differentiable in each component of $x = (x_1, \dots, x_d)$, and
- $u^{(\pi_\sigma, \pi_\mu)}$ fulfills Equation 182 for all $(t, x) \in (0, T) \times \mathbb{R}^d$ in the classical sense.

Similarly, a solution function for the Kolmogorov PDE 182 with boundary condition

$$u(0, x) = \varphi(x) \quad (184)$$

on the set $[0, T] \times \mathbb{R}^d$ is defined as a function

$$v^{(\pi_\sigma, \pi_\mu)} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R} \quad (185)$$

with the following properties:

- $v^{(\pi_\sigma, \pi_\mu)}$ is a classical solution function of the Kolmogorov PDE 182 on $(0, T) \times \mathbb{R}^d$ in the sense defined above,
- $v^{(\pi_\sigma, \pi_\mu)} \in \mathcal{C}([0, T] \times \mathbb{R}^d, \mathbb{R})$, and
- $v^{(\pi_\sigma, \pi_\mu)}(0, x) = \varphi(x)$ for all $x \in \mathbb{R}^d$.

Note, that the family of Kolmogorov PDEs contains important equations from various areas of applied mathematics, such as the physical heat equation and the Black-Scholes equation from computational finance. It is therefore of natural interest for practical researchers.

Just like is the case with many other families of PDEs, in general one cannot always expect the existence and uniqueness of classical solutions for Kolmogorov equations. To correct for this, one has to weaken the notion of what is meant by a solution of a Kolmogorov PDE in order to achieve convenient results about the existence and uniqueness of solution functions. This leads to the concept of *viscosity solutions* [52] [53], which have turned out to be the natural solution concept for Kolmogorov equations and a variety of other types of PDEs. First and foremost, viscosity solutions are defined for the class of degenerate elliptic partial differential equations

¹⁵We write $u^{(\pi_\sigma, \pi_\mu)}$ instead of just u in order to remind the reader on the dependence of the solution of the Kolmogorov PDE on specific parameters of the affine-linear coefficients σ and μ .

Definition 4.3 (Degenerate Ellipticity). *Let $\tilde{d} \in \mathbb{N}_{\geq 1}$ be a natural number and $O \subseteq \mathbb{R}^{\tilde{d}}$ be an open set. Furthermore, let*

$$H : O \times \mathbb{R} \times \mathbb{R}^{\tilde{d}} \times \mathbb{S}_{\tilde{d} \times \tilde{d}}(\mathbb{R}) \rightarrow \mathbb{R} \quad (186)$$

be a function. H is called degenerate elliptic if H is continuous and if, for all $\tilde{x} \in O, \tilde{u} \in \mathbb{R}, \tilde{g} \in \mathbb{R}^{\tilde{d}}$ and for all $A, B \in \mathbb{S}_{\tilde{d} \times \tilde{d}}(\mathbb{R})$ for which $B - A$ is positive semidefinite, it holds, that

$$H(\tilde{x}, \tilde{u}, \tilde{g}, B) \leq H(\tilde{x}, \tilde{u}, \tilde{g}, A). \quad (187)$$

The partial differential equation given by

$$H(\tilde{x}, u(\tilde{x}), (\nabla u)(\tilde{x}), (\text{Hess } u)(\tilde{x})) = 0 \quad (188)$$

is called degenerate elliptic if H is degenerate elliptic.

Remark 4.2 (Degenerate Ellipticity of First-Order Equations). According to Definition 4.3, all first order differential equations are automatically degenerate elliptic, since their associated differential operators can be seen as functions which are constant in the argument corresponding to the Hessian.

Definition 4.4 (Viscosity Solutions of Degenerate Elliptic PDEs). *Let $\tilde{d} \in \mathbb{N}_{\geq 1}$ be a natural number and $O \subseteq \mathbb{R}^{\tilde{d}}$ be an open set. Furthermore, let*

$$H : O \times \mathbb{R} \times \mathbb{R}^{\tilde{d}} \times \mathbb{S}_{\tilde{d} \times \tilde{d}}(\mathbb{R}) \rightarrow \mathbb{R} \quad (189)$$

be a degenerate elliptic function. Now consider the partial differential equation

$$H(\tilde{x}, u(\tilde{x}), (\nabla u)(\tilde{x}), (\text{Hess } u)(\tilde{x})) = 0. \quad (190)$$

A function $u^{(H)} : O \rightarrow \mathbb{R}$ is called a viscosity subsolution of 190 if $u^{(H)}$ is upper semicontinuous and if, for all $\tilde{x} \in O$ and for all functions $\phi \in \mathcal{C}^2(O, \mathbb{R})$ with $\phi \geq u^{(H)}$ and $\phi(\tilde{x}) = u^{(H)}(\tilde{x})$, it holds that

$$H(\tilde{x}, \phi(\tilde{x}), (\nabla \phi)(\tilde{x}), (\text{Hess } \phi)(\tilde{x})) \leq 0 \quad (191)$$

Similarly, a function $u^{(H)} : O \rightarrow \mathbb{R}$ is called a viscosity supersolution of 190 if $u^{(H)}$ is lower semicontinuous and if, for all $\tilde{x} \in O$ and for all functions $\phi \in \mathcal{C}^2(O, \mathbb{R})$ with $\phi \leq u^{(H)}$ and $\phi(\tilde{x}) = u^{(H)}(\tilde{x})$, it holds that

$$H(\tilde{x}, \phi(\tilde{x}), (\nabla \phi)(\tilde{x}), (\text{Hess } \phi)(\tilde{x})) \geq 0 \quad (192)$$

Finally, a function $u^{(H)} : O \rightarrow \mathbb{R}$ is called a viscosity solution of 190 if $u^{(H)}$ is both a viscosity subsolution and a viscosity supersolution of 190.

Remark 4.3 (Properties of Viscosity Solutions). Note, that all viscosity solutions are, per definition, both upper semicontinuous and lower semicontinuous and therefore automatically continuous. However, viscosity solutions need not necessarily be differentiable.

Proposition 4.1 (Kolmogorov Equations are Degenerate Elliptic). *For any matrix $A \in \mathbb{S}_{d+1 \times d+1}(\mathbb{R})$, denote with $A^{(1,1)} \in \mathbb{S}_{d \times d}(\mathbb{R})$ is the submatrix extracted from $A \in \mathbb{S}_{d+1 \times d+1}(\mathbb{R})$ by deleting the 1-th row and 1-th column. Similarly, for any vector $g \in \mathbb{R}^{d+1}$, denote with $g^{(1)} \in \mathbb{R}^d$, the subvector extracted from $g \in \mathbb{R}^{d+1}$ by deleting the 1-th entry of g .*

Now let

$$H_{\sigma,\mu,Kol} : (0, T) \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{d+1} \times \mathbb{S}_{d+1 \times d+1}(\mathbb{R}) \rightarrow \mathbb{R} \quad (193)$$

be a function defining a Kolmogorov PDE, i.e.

$$H_{\sigma,\mu,Kol}(\tilde{t}, \tilde{x}, \tilde{u}, \tilde{g}, A) = \tilde{g}_1 - \frac{1}{2} \text{Trace}[\sigma(\tilde{x})[\sigma(\tilde{x})]^* A^{(1,1)}] - \langle \mu(\tilde{x}), \tilde{g}^{(1)} \rangle. \quad (194)$$

Then $H_{\sigma,\mu,Kol}$ is degenerate elliptic (and, therefore, Kolmogorov equations are degenerate elliptic).

Proof. Clearly, $H_{\sigma,\mu,Kol}$ is continuous. Now fix $\tilde{t} \in (0, T)$, $\tilde{x} \in \mathbb{R}^d$, $\tilde{u} \in \mathbb{R}$, $\tilde{g} \in \mathbb{R}^{d+1}$, and choose symmetric matrices $A, B \in \mathbb{S}_{d+1 \times d+1}(\mathbb{R})$ such that $B - A$ is positive semidefinite. We need to show that

$$H_{\sigma,\mu,Kol}(\tilde{t}, \tilde{x}, \tilde{u}, \tilde{g}, B) \leq H_{\sigma,\mu,Kol}(\tilde{t}, \tilde{x}, \tilde{u}, \tilde{g}, A). \quad (195)$$

To see this, first recall, that, since $B - A$ is symmetric and positive semidefinite, the submatrix $B^{(1,1)} - A^{(1,1)}$ is symmetric and positive semidefinite. Second, note that the matrix $\sigma(x)^*(B^{(1,1)} - A^{(1,1)})\sigma(x)$ is symmetric and positive semidefinite. Third, remember, that the trace of a matrix

$$\text{Trace} : \mathbb{R}^{d \times d} \rightarrow \mathbb{R} \quad (196)$$

is a linear operator, for which, for all $M_1, M_2 \in \mathbb{R}^{d \times d}$, it holds that

$$\text{Trace}(M_1 M_2) = \text{Trace}(M_2 M_1). \quad (197)$$

and for which, for each symmetric and positive semidefinite $M \in \mathbb{S}_{d \times d}$ the inequality

$$\text{Trace}(M) \geq 0 \quad (198)$$

must hold. With all of this in mind, we can calculate

$$\begin{aligned} H_{\sigma,\mu,Kol}(\tilde{t}, \tilde{x}, \tilde{u}, \tilde{g}, A) - H_{\sigma,\mu,Kol}(\tilde{t}, \tilde{x}, \tilde{u}, \tilde{g}, B) &= \\ \frac{1}{2} (-\text{Trace}[\sigma(\tilde{x})[\sigma(\tilde{x})]^* A^{(1,1)}] + \text{Trace}[\sigma(\tilde{x})[\sigma(\tilde{x})]^* B^{(1,1)}]) &= \\ \frac{1}{2} (\text{Trace}[\sigma(\tilde{x})[\sigma(\tilde{x})]^* (B^{(1,1)} - A^{(1,1)})]) &= \\ \frac{1}{2} (\text{Trace}[[\sigma(\tilde{x})]^* (B^{(1,1)} - A^{(1,1)})\sigma(\tilde{x})]) &\geq 0, \end{aligned} \quad (199)$$

which proves Inequality 195. $\square$

Remark 4.4 (Consequences of Switching the Sign in the Equation). Note, that degenerate ellipticity is potentially unstable under the operation of switching the sign of the PDE. In particular, the function $-H_{\sigma,\mu,\text{Kol}}$ fails to be degenerate elliptic. Another scenario, which is possible as well in different situations, is that switching the sign of the equation leads to another degenerate elliptic equation, but to a different set of viscosity solutions¹⁶. This illustrates the (perhaps surprising) fact, that two different ways to mathematically express a PDE can be equivalent in the classical sense, but not in the viscosity sense.

Proposition 4.2 (Classical Solutions of Kolmogorov PDEs are Viscosity Solutions). *Let the function*

$$u^{(\pi_\sigma, \pi_\mu)} : (0, T) \times \mathbb{R}^d \rightarrow \mathbb{R} \quad (200)$$

be a classical solution of the Kolmogorov PDE

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0 \quad (201)$$

on the open set $(0, T) \times \mathbb{R}^d$. Then $u^{(\pi_\sigma, \pi_\mu)}$ is a viscosity solution of Equation 201 on $(0, T) \times \mathbb{R}^d$.

Proof. We show, that $u^{(\pi_\sigma, \pi_\mu)}$ must be a viscosity subsolution of Equation 201; the proof, that $u^{(\pi_\sigma, \pi_\mu)}$ is a viscosity supersolution of 201 is analogous.

First, note that $u^{(\pi_\sigma, \pi_\mu)} \in \mathcal{C}^{1,2}((0, T) \times \mathbb{R}^d, \mathbb{R}) \subset \mathcal{C}((0, T) \times \mathbb{R}^d, \mathbb{R})$ must be continuous and therefore also semicontinuous. We now continue with the notation introduced in Proposition 4.1.

Recall the degenerate elliptic function

$$H_{\sigma,\mu,\text{Kol}} : (0, T) \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{d+1} \times \mathbb{S}_{d+1 \times d+1}(\mathbb{R}) \rightarrow \mathbb{R} \quad (202)$$

$$H_{\sigma,\mu,\text{Kol}}(\tilde{t}, \tilde{x}, \tilde{u}, \tilde{g}, \tilde{A}) = \tilde{g}_1 - \frac{1}{2} \text{Trace}[\sigma(\tilde{x})[\sigma(\tilde{x})]^* \tilde{A}^{(1,1)}] - \langle \mu(\tilde{x}), \tilde{g}^{(1)} \rangle$$

defining Equation 201. Now fix $(\tilde{t}, \tilde{x}) \in (0, T) \times \mathbb{R}^d$ and $\phi \in \mathcal{C}^2((0, T) \times \mathbb{R}^d, \mathbb{R})$ with $\phi \geq u^{(\pi_\sigma, \pi_\mu)}$ and $\phi(\tilde{t}, \tilde{x}) = u^{(\pi_\sigma, \pi_\mu)}(\tilde{t}, \tilde{x})$. Our aim is to show, that

$$H_{\sigma,\mu,\text{Kol}}(\tilde{t}, \tilde{x}, \phi(\tilde{t}, \tilde{x}), (\nabla \phi)(\tilde{t}, \tilde{x}), (\text{Hess } \phi)(\tilde{t}, \tilde{x})) \leq 0. \quad (203)$$

First, observe that Equation 201 does not involve the second derivative of $u^{(\pi_\sigma, \pi_\mu)}$ with respect to t ; in addition, note that neither t nor $u^{(\pi_\sigma, \pi_\mu)}$ explicitly appear anywhere in PDE 201. It is therefore possible to express $H_{\sigma,\mu,\text{Kol}}$ via a function

$$h_{\sigma,\mu,\text{Kol}} : \mathbb{R}^d \times \mathbb{R}^{d+1} \times \mathbb{S}_{d \times d}(\mathbb{R}) \rightarrow \mathbb{R} \quad (204)$$

such that, for all $\hat{t} \in (0, T)$, $\hat{x} \in \mathbb{R}^d$, $\hat{u} \in \mathbb{R}$, $\hat{g} \in \mathbb{R}^{d+1}$, $\hat{A} \in \mathbb{S}_{d+1 \times d+1}(\mathbb{R})$ it holds that

$$H_{\sigma,\mu,\text{Kol}}(\hat{t}, \hat{x}, \hat{u}, \hat{g}, \hat{A}) = h_{\sigma,\mu,\text{Kol}}(\hat{x}, \hat{g}, \hat{A}^{(1,1)}). \quad (205)$$

¹⁶ See [54], page 28, for an example of this phenomenon.

It follows from the proof of Proposition 4.1, that $h_{\sigma,\mu,\text{Kol}}$ must inherit the degenerate elliptic antimonotonicity-criterion in its last variable $\tilde{A}$ from $H_{\sigma,\mu,\text{Kol}}$. Since $u^{(\pi_\sigma, \pi_\mu)}$ is a classical solution of Equation 201, it holds for all $(t, x) \in (0, T) \times \mathbb{R}^d$ that

$$h_{\sigma,\mu,\text{Kol}}(x, (\nabla u^{(\pi_\sigma, \pi_\mu)})(t, x), (\text{Hess}_x u^{(\pi_\sigma, \pi_\mu)})(t, x)) = 0. \quad (206)$$

Now notice, that, under our assumptions, the function

$$\phi - u^{(\pi_\sigma, \pi_\mu)} : (0, T) \times \mathbb{R}^d \rightarrow \mathbb{R}, \quad (t, x) \mapsto \phi(t, x) - u^{(\pi_\sigma, \pi_\mu)}(t, x), \quad (207)$$

must have a local minimum at $(\tilde{t}, \tilde{x})$; likewise, the function

$$\phi_{\tilde{t}} - u_{\tilde{t}}^{(\pi_\sigma, \pi_\mu)} : \mathbb{R}^d \rightarrow \mathbb{R}, \quad x \mapsto \phi(\tilde{t}, x) - u^{(\pi_\sigma, \pi_\mu)}(\tilde{t}, x), \quad (208)$$

must have a local minimum at $\tilde{x}$. This implies that

$$(\nabla \phi)(\tilde{t}, \tilde{x}) = (\nabla u^{(\pi_\sigma, \pi_\mu)})(\tilde{t}, \tilde{x}) \quad (209)$$

holds and that the symmetric matrix

$$\begin{aligned} &(\text{Hess}_x \phi)(\tilde{t}, \tilde{x}) - (\text{Hess}_x u^{(\pi_\sigma, \pi_\mu)})(\tilde{t}, \tilde{x}) = \\ &(\text{Hess}_x (\phi - u^{(\pi_\sigma, \pi_\mu)}))(\tilde{t}, \tilde{x}) = (\text{Hess}(\phi_{\tilde{t}} - u_{\tilde{t}}^{(\pi_\sigma, \pi_\mu)}))(\tilde{x}) \in \mathbb{S}_{d \times d}(\mathbb{R}) \end{aligned} \quad (210)$$

must be positive semidefinite. Using all of the above facts, we conclude that

$$\begin{aligned} &H_{\sigma,\mu,\text{Kol}}(\tilde{t}, \tilde{x}, \phi(\tilde{t}, \tilde{x}), (\nabla \phi)(\tilde{t}, \tilde{x}), (\text{Hess} \phi)(\tilde{t}, \tilde{x})) = \\ &h_{\sigma,\mu,\text{Kol}}(\tilde{x}, (\nabla \phi)(\tilde{t}, \tilde{x}), (\text{Hess}_x \phi)(\tilde{t}, \tilde{x})) = \\ &h_{\sigma,\mu,\text{Kol}}(\tilde{x}, (\nabla u^{(\pi_\sigma, \pi_\mu)})(\tilde{t}, \tilde{x}), (\text{Hess}_x \phi)(\tilde{t}, \tilde{x})) \leq \\ &h_{\sigma,\mu,\text{Kol}}(\tilde{x}, (\nabla u^{(\pi_\sigma, \pi_\mu)})(\tilde{t}, \tilde{x}), (\text{Hess}_x u^{(\pi_\sigma, \pi_\mu)})(\tilde{t}, \tilde{x})) = 0. \end{aligned} \quad (211)$$

This proves, that $u^{(\pi_\sigma, \pi_\mu)}$ is a viscosity subsolution of PDE 201. $\square$

Remark 4.5 (Classical Solutions are Viscosity Solutions). By making only minor changes in above proof, one can generalize Proposition 4.2 and show that if $u^{(\pi_\sigma, \pi_\mu)}$ is classical solution of any given degenerate elliptic equation

$$H(\tilde{x}, u(\tilde{x}), (\nabla u)(\tilde{x}), (\text{Hess} u)(\tilde{x})) = 0 \quad (212)$$

on some open set $O \subseteq \mathbb{R}^{\tilde{d}}$, then it is automatically also a viscosity solutions of this equation on O .

Example 4.1 (A Proper Viscosity Solution). In general, viscosity solutions need not be differentiable; therefore, they provide a true generalization of the classical solution concept. To see a simple example of a proper viscosity solution,

which is not a classical solution, consider the following simple 1-dimensional differential problem:

$$|y'(x)| - 1 = 0, \quad x \in (-1, 1). \quad (213)$$

Since Equation 213 is of first order, it is necessarily degenerate elliptic. A viscosity solution of the problem is given by

$$\hat{y}(x) = 1 - |x| \quad (214)$$

Since $\hat{y}$ is not differentiable at $x = 0$, however, it cannot be a classical solution.

We end this subsection by citing an important theorem, which assures us, that Kolmogorov equations do always have a unique at most polynomially growing viscosity solution $u^{(\pi_\sigma, \pi_\mu)}$ if our requirements on σ, μ and φ are indeed fulfilled.

Theorem 4.1 (Existence and Uniqueness of Viscosity Solutions for Kolmogorov PDEs). *Consider the Kolmogorov partial differential equation*

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0 \quad (215)$$

with boundary condition

$$u(0, x) = \varphi(x). \quad (216)$$

Recall our assumptions on σ, μ and φ from Setting 4.1, which we now restate for the convenience of the reader. Let

$$\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}, \quad (217)$$

$$\mu : \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad (218)$$

be affine linear transformations, i.e. let there be $d + 1$ matrices

$$\pi_\sigma := (A_1^{(\sigma)}, \dots, A_d^{(\sigma)}, \bar{A}^{(\sigma)}) \in (\mathbb{R}^{d \times d})^{d+1} \quad (219)$$

such that, for all $x \in \mathbb{R}^d$,

$$\sigma(x) = (A_1^{(\sigma)}x \mid \dots \mid A_d^{(\sigma)}x) + \bar{A}^{(\sigma)}, \quad (220)$$

where $A_i^{(\sigma)}x$ describes the i -th column of the matrix $\sigma(x) \in \mathbb{R}^{d \times d}$, and let there be a matrix-vector pair

$$\pi_\mu := (A^{(\mu)}, \bar{a}^{(\mu)}) \in \mathbb{R}^{d \times d} \times \mathbb{R}^d \quad (221)$$

such that, for all $x \in \mathbb{R}^d$,

$$\mu(x) = A^{(\mu)}x + \bar{a}^{(\mu)}, \quad (222)$$

and, moreover, let

$$\varphi : \mathbb{R}^d \rightarrow \mathbb{R} \quad (223)$$

be continuous and at most polynomially growing, i.e. let φ be continuous and let there be a constant $C \in [1, \infty)$ such that, for all $x \in \mathbb{R}^d$ it holds that

$$|\varphi(x)| \leq C(1 + \|x\|)^C. \quad (224)$$

Then there exists exactly one function

$$u^{(\pi_\sigma, \pi_\mu)} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R} \quad (225)$$

on the set $[0, T] \times \mathbb{R}^d$ which has the following properties:

- $u^{(\pi_\sigma, \pi_\mu)} \in \mathcal{C}([0, T] \times \mathbb{R}^d, \mathbb{R})$ is continuous on $[0, T] \times \mathbb{R}^d$,
- for all $x \in \mathbb{R}^d$, it holds that $u^{(\pi_\sigma, \pi_\mu)}(0, x) = \varphi(x)$,
- $u^{(\pi_\sigma, \pi_\mu)}|_{(0, T) \times \mathbb{R}^d}$ is a viscosity solution of Kolmogorov equation 273 on the open set $(0, T) \times \mathbb{R}^d$, and
- $\inf_{q \in (0, \infty)} \sup_{(t, x) \in [0, T] \times \mathbb{R}^d} \frac{|u^{(\pi_\sigma, \pi_\mu)}(t, x)|}{1 + \|x\|^q} < \infty$.

Proof. For a detailed proof, take a look at Corollary 2.23 in the article of Grohs, Hornung, Jentzen and Wurstemberger [10]. It is easy to show, that σ and μ fulfill

$$\mu(\lambda x + y) + \lambda \mu(0) = \lambda \mu(x) + \mu(y) \quad (226)$$

and

$$\sigma(\lambda x + y) + \lambda \sigma(0) = \lambda \sigma(x) + \sigma(y) \quad (227)$$

for all $x, y \in \mathbb{R}^d$ and $\lambda \in \mathbb{R}$, if and only if they are affine linear transformations in the sense defined in Setting 4.1.

Since we assume, that σ and μ are affine linear functions, this implies, that, in our setting, all requirements of Corollary 2.23 in [10] are fulfilled. $\square$

Remark 4.6 (Growth Condition for Uniqueness). The growth condition

$$\inf_{q \in (0, \infty)} \sup_{(t, x) \in [0, T] \times \mathbb{R}^d} \frac{|u^{(\pi_\sigma, \pi_\mu)}(t, x)|}{1 + \|x\|^q} < \infty \quad (228)$$

from Theorem 4.1 is necessary in order to obtain uniqueness of the solution function. Amongst other things, Condition 228 excludes a variety of possible poorly behaved pathological solutions. For the 1-dimensional heat equation, for example, there are infinitely many solutions which violate Condition 228, exhibit finite-time blowup and do not represent the physical phenomenon being modelled. In the context of Kolmogorov PDEs, the unique viscosity solution $u^{(\pi_\sigma, \pi_\mu)}$ described in Theorem 4.1 is usually the one which is relevant for practical modelling purposes.

Remark 4.7 (Extending the Time Interval for Kolmogorov PDEs). Let $\tilde{T} > T$ and let $u^{(\pi_\sigma, \pi_\mu, T)}$ and $u^{(\pi_\sigma, \pi_\mu, \tilde{T})}$ be the unique viscosity solutions of the same Kolmogorov equation on the sets $[0, T] \times \mathbb{R}^d$ and $[0, \tilde{T}] \times \mathbb{R}^d$. Then it immediately follows from the properties of $u^{(\pi_\sigma, \pi_\mu, T)}$ and $u^{(\pi_\sigma, \pi_\mu, \tilde{T})}$ (see Theorem 4.1), that

$$u^{(\pi_\sigma, \pi_\mu, T)} = u^{(\pi_\sigma, \pi_\mu, \tilde{T})}|_{[0, T] \times \mathbb{R}^d}. \quad (229)$$

For an even deeper dive into the theory of viscosity solutions of Kolmogorov partial differential equations, we recommend the article of Hairer, Hutzenthaler and Jentzen [53].

4.3 Stochastic Differential Equations and their Connection to Kolmogorov PDEs via the Feynman-Kac Formula

In this section, we discuss properties such as existence of uniqueness of solutions of a particular family of stochastic differential equations, whose solution processes exhibit an important and deep link to Kolmogorov equations, described by the Feynman-Kac formula. First, we extend our basic setting by including a stochastic modelling framework.

Setting 4.2. In addition to Setting 4.1, for the rest of Section 4,

- for any topological space $(U, \mathcal{O})$, let $\mathcal{B}(U)$ be the Borel σ -algebra on U ,
- let $\mathbb{R}^d$ and $[-\infty, \infty]^d$ be equipped with their canonical Borel σ -algebras $\mathcal{B}(\mathbb{R}^d)$ and $\mathcal{B}([-\infty, \infty]^d)$,
- let $x_0 \in \mathbb{R}^d$,
- let $(\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P})$ be a suitable, complete probability space with a complete, right-continuous filtration $(\mathcal{G}_t)_{t \in [0, T]}$, and
- let $B : [0, T] \times \Omega \rightarrow \mathbb{R}^d$, $(t, \omega) \mapsto B_t(\omega)$, be a standard d -dimensional $(\mathcal{G}_t)_{t \in [0, T]}$ -adapted Brownian motion on $\mathbb{R}^d$, such that $B_0 = 0$ and such that, for every pair $(t, s) \in \mathbb{R}^2$ with $0 \leq t < s$, the random variable $B_s - B_t$ is independent of $\mathcal{G}_t$.

Remark 4.8 (Regularity Assumptions on our Probability Space). Completeness of $(\Omega, \mathcal{G}, \mathbb{P})$, as well as right-continuity and completeness of the filtration $(\mathcal{G}_t)_{t \in [0, T]}$, are standard requirements in the theory of time-continuous stochastic processes and are in fact often referred to as the *usual hypotheses*. This is not restrictive; one can always replace a probability space with its canonical completion. Similarly, it is always possible, to replace a given filtration with a closely related right-continuous filtration and then take its completion in a canonical way. Some fundamental theorems in stochastic calculus, such as certain statements related to the regularity of the paths of important types of stochastic processes, require the usual hypotheses. For more information, consider the book of Le Gall [55].

Definition 4.5 (Stochastic Differential Equation). *The equation*

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T], \quad (230)$$

is called *stochastic differential equation*, or, in short, *SDE*. The vector $x_0 \in \mathbb{R}^d$ is called *initial value of the SDE*. The functions $\mu : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is referred to as *drift coefficient* and the function $\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}$ is referred to as *diffusion coefficient* of the SDE. The set $[0, T]$ is called *time interval of the SDE*.

Remark 4.9 (Multidimensional Integration). Both integral expressions in Equation 230 are to be read componentwise:

$$\begin{aligned} \int_0^t \mu(S_s) ds &:= \left(\int_0^t \mu_i(S_s) ds \right)_{i=1}^d, \\ \int_0^t \sigma(S_s) dB_s &:= \left(\sum_{j=1}^d \int_0^t \sigma_{i,j}(S_s) dB_s^{(j)} \right)_{i=1}^d. \end{aligned} \quad (231)$$

Here, $(B_s^{(j)})_{s \in [0, T]}$ denotes the j -th 1-dimensional component Brownian motion process of the d -dimensional Brownian motion $(B_s)_{s \in [0, T]}$.

Remark 4.10 (Differential Notation for SDEs). The name *stochastic differential equation*, rather than the less common (but perhaps more intuitive) name *stochastic integral equation*, is motivated by the differential notation

$$dS_t = S_t dt + S_t dB_t, \quad S_0 = x_0, \quad t \in [0, T], \quad (232)$$

for Equation 230. This type of notation, however, has no strict mathematical meaning on its own and always implicitly refers to an equation defined via one Lebesgue integral and one Ito integral.

Definition 4.6 (Solution of a Stochastic Differential Equation). *A solution process for the stochastic differential equation*

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T], \quad (233)$$

is a function

$$S^{(x_0, \pi_\sigma, \pi_\mu)} : [0, T] \times \Omega \rightarrow \mathbb{R}^d, \quad (t, \omega) \mapsto S_t^{(x_0, \pi_\sigma, \pi_\mu)}(\omega), \quad (234)$$

satisfying the following properties:

- $S^{(x_0, \pi_\sigma, \pi_\mu)}$ is a $(\mathcal{G}_t)_{t \in [0, T]}$ -adapted stochastic process,
- $S^{(x_0, \pi_\sigma, \pi_\mu)}$ has $\mathbb{P}$ -a.s. continuous sample paths, i.e. there exists an event $A \in \mathcal{G}$ with $\mathbb{P}(A) = 1$ such that, for all $\tilde{\omega} \in A$, the map $t \mapsto S_t^{(x_0, \pi_\sigma, \pi_\mu)}(\tilde{\omega})$ is continuous, and
- for every $t \in [0, T]$, Equation 236 holds $\mathbb{P}$ -a.s. for $S_t^{(x_0, \pi_\sigma, \pi_\mu)} : \Omega \rightarrow \mathbb{R}^d$.

We now proceed by presenting the main existence and uniqueness result for solution processes of SDEs; but before, we introduce some auxiliary definitions and functions, which are useful in subsequent parts of this section in order to deal with some upcoming technicalities.

Definition 4.7 (Stochastic Processes with Continuous Sample Paths). *The set*

$$\mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d) := \{S : [0, T] \times \Omega \rightarrow \mathbb{R}^d \mid \quad (235)$$

S is $(\mathcal{G}_t)_{t \in [0, T]}$ -adapted and has $\mathbb{P}$ -a.s. continuous sample paths

is (perhaps unsurprisingly) called the set of d -dimensional stochastic processes with continuous sample paths in $\mathbb{R}^d$ over $(\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P})$.

Remark 4.11 (Space of Solution Processes). Clearly, from Definition 4.6 it follows, that any solution process of

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T], \quad (236)$$

must be contained in $\mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$.

Definition 4.8 (Indistinguishability of Stochastic Processes). *Two processes $S, \bar{S} \in \mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ are called indistinguishable, if there exists an event $A \in \mathcal{G}$ with $\mathbb{P}(A) = 1$ such that, for all $\tilde{\omega} \in A$, the sample path maps*

$$[0, T] \ni t \mapsto S_t(\tilde{\omega}) \in \mathbb{R}^d \quad (237)$$

and

$$[0, T] \ni t \mapsto \bar{S}_t(\tilde{\omega}) \in \mathbb{R}^d \quad (238)$$

are identical.

Remark 4.12 (Almost Surely Continuous Modifications are Indistinguishable). Two processes $S, \bar{S} \in \mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ are called modifications of each other, if, for all $t \in [0, T]$, it holds that

$$\mathbb{P}(S_t = \bar{S}_t) = 1. \quad (239)$$

It is a well-known result from stochastic calculus, that the processes $S, \bar{S} \in \mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ are modifications of each other if and only if they are indistinguishable. This equivalence, however, breaks down, when the assumption, that S and $\bar{S}$ are $\mathbb{P}$ -a.s. continuous, is dropped; in that case, indistinguishability is the stronger property.

Definition 4.9 (Componentwise Upper Limit). *Let $(\tilde{\Omega}, \tilde{\mathcal{G}})$ be a measurable space and, for $k \in \mathbb{N}_{\geq 0}$, let*

$$h^{(k)} : \tilde{\Omega} \rightarrow \mathbb{R}^d \quad (240)$$

be a member of a sequence $(h^{(k)})_{k \in \mathbb{N}_{\geq 0}}$ of $\tilde{\mathcal{G}}/\mathcal{B}(\mathbb{R}^d)$ -measurable functions. Then the function

$$\begin{aligned} & \limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} h^{(k)} : \tilde{\Omega} \rightarrow [-\infty, \infty]^d, \\ & \limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} h^{(k)}(\tilde{\omega}) := (\limsup_{k \rightarrow \infty} h_1^{(k)}(\tilde{\omega}), \dots, \limsup_{k \rightarrow \infty} h_d^{(k)}(\tilde{\omega})), \end{aligned} \quad (241)$$

is called componentwise upper limit of $(h^{(k)})_{k \in \mathbb{N}_{\geq 0}}$.

Definition 4.10 (Finiteness Function). *The function*

$$r_{\mathbb{R}^d} : [-\infty, \infty]^d \rightarrow \mathbb{R}^d, \quad r_{\mathbb{R}^d}(x) = \begin{cases} x, & x \in \mathbb{R}^d, \\ 0, & x \in [-\infty, \infty]^d \setminus \mathbb{R}^d. \end{cases} \quad (242)$$

is called d -dimensional finiteness function.

Definition 4.11 (Finitely Retracted Componentwise Upper Limit). *Let $(\tilde{\Omega}, \tilde{\mathcal{G}})$ be a measurable space and, for $k \in \mathbb{N}_{\geq 0}$, let*

$$h^{(k)} : \tilde{\Omega} \rightarrow \mathbb{R}^d \quad (243)$$

be a member of a sequence $(h^{(k)})_{k \in \mathbb{N}_{\geq 0}}$ of $\tilde{\mathcal{G}}/\mathcal{B}(\mathbb{R}^d)$ -measurable functions. Then the function

$$\begin{aligned} & \limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} \text{fin } h^{(k)} : \tilde{\Omega} \rightarrow \mathbb{R}^d, \\ & \limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} \text{fin } h^{(k)}(\tilde{\omega}) := r_{\mathbb{R}^d} \left(\limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} h^{(k)}(\tilde{\omega}) \right), \end{aligned} \quad (244)$$

is called finitely retracted componentwise upper limit of $(h^{(k)})_{k \in \mathbb{N}_{\geq 0}}$.

Lemma 4.1 (Measurability of Finitely Retracted Componentwise Upper Limit). *Let $(\tilde{\Omega}, \tilde{\mathcal{G}})$ be a measurable space and, for $k \in \mathbb{N}_{\geq 0}$, let*

$$h^{(k)} : \tilde{\Omega} \rightarrow \mathbb{R}^d \quad (245)$$

be a sequence of $\tilde{\mathcal{G}}/\mathcal{B}(\mathbb{R}^d)$ -measurable functions. Then the finitely retracted componentwise upper limit

$$\limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} \text{fin } h^{(k)} : \tilde{\Omega} \rightarrow \mathbb{R}^d \quad (246)$$

of $(h^{(k)})_{k \in \mathbb{N}_{\geq 0}}$ is $\tilde{\mathcal{G}}/\mathcal{B}(\mathbb{R}^d)$ -measurable.

Proof. It is trivial to show, that

$$r_{\mathbb{R}^d} : [-\infty, \infty]^d \rightarrow \mathbb{R}^d \quad (247)$$

is $\mathcal{B}([-\infty, \infty]^d)/\mathcal{B}(\mathbb{R}^d)$ -measurable. Also notice, that for the case $d = 1$, it is a well known fact from basic measure theory, that

$$\limsup_{k \rightarrow \infty}^{(\mathbb{R})} h^{(k)} : \tilde{\Omega} \rightarrow [-\infty, \infty] \quad (248)$$

is $\tilde{\mathcal{G}}/\mathcal{B}([-\infty, \infty])$ -measurable. This implies in particular, that for general $d \in \mathbb{N}_{\geq}$, all component functions

$$\limsup_{k \rightarrow \infty} h_1^{(k)}, \dots, \limsup_{k \rightarrow \infty} h_d^{(k)} : \tilde{\Omega} \rightarrow [-\infty, \infty] \quad (249)$$

of

$$\limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} h^{(k)} = (\limsup_{k \rightarrow \infty} h_1^{(k)}, \dots, \limsup_{k \rightarrow \infty} h_d^{(k)}) : \tilde{\Omega} \rightarrow [-\infty, \infty]^d \quad (250)$$

are $\tilde{\mathcal{G}}/\mathcal{B}([-\infty, \infty])$ -measurable. Now remember, that $\mathcal{B}([-\infty, \infty]^d)$ is generated by sets of the form

$$A_1 \times \dots \times A_d \in \mathcal{B}([-\infty, \infty]^d) \quad (251)$$

with $A_1, \dots, A_d \in \mathcal{B}([-\infty, \infty])$ ¹⁷. Furthermore, remember, that it is sufficient to check measurability of functions only on generators of σ -algebras. With this in mind, we choose $A_1, \dots, A_d \in \mathcal{B}([-\infty, \infty])$ and conclude

$$\left(\limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} h^{(k)} \right)^{-1} (A_1 \times \dots \times A_d) = \bigcap_{i=1}^d \left(\limsup_{k \rightarrow \infty} h_i^{(k)} \right)^{-1} (A_i) \in \tilde{\mathcal{G}}, \quad (252)$$

which shows, that

$$\limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} h^{(k)} : \tilde{\Omega} \rightarrow [-\infty, \infty]^d \quad (253)$$

is $\tilde{\mathcal{G}}/\mathcal{B}([-\infty, \infty]^d)$ -measurable. Since

$$\limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} \text{fin } h^{(k)} = r_{\mathbb{R}^d} \circ \limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} h^{(k)} \quad (254)$$

and measurability of functions is stable under composition, $\limsup_{k \rightarrow \infty}^{(\mathbb{R}^d)} \text{fin } h^{(k)}$ must be $\tilde{\mathcal{G}}/\mathcal{B}(\mathbb{R}^d)$ -measurable, which completes the proof of the statement. $\square$

Theorem 4.2 (Existence and Uniqueness of Solutions of SDEs). *Let*

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T], \quad (255)$$

be a stochastic differential equation. Recall our assumptions on σ and μ from Setting 4.1, which we now restate for the convenience of the reader. Let

$$\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}, \quad (256)$$

$$\mu : \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad (257)$$

be affine linear transformations, i.e. let there be $d+1$ matrices

$$\pi_\sigma := (A_1^{(\sigma)}, \dots, A_d^{(\sigma)}, \bar{A}^{(\sigma)}) \in (\mathbb{R}^{d \times d})^{d+1} \quad (258)$$

such that, for all $x \in \mathbb{R}^d$,

$$\sigma(x) = (A_1^{(\sigma)} x \mid \dots \mid A_d^{(\sigma)} x) + \bar{A}^{(\sigma)}, \quad (259)$$

¹⁷A subtle fact: this is true, because $[-\infty, \infty]^d$ is a second-countable topological space, which is a sufficient requirement for the σ -algebras $\mathcal{B}([-\infty, \infty]^d)$ and $\bigotimes_{i=1}^d \mathcal{B}([-\infty, \infty])$ to be the same.

where $A_i^{(\sigma)}x$ describes the i -th column of the matrix $\sigma(x) \in \mathbb{R}^{d \times d}$, and let there be a matrix-vector pair

$$\pi_\mu := (A^{(\mu)}, \bar{a}^{(\mu)}) \in \mathbb{R}^{d \times d} \times \mathbb{R}^d \quad (260)$$

such that, for all $x \in \mathbb{R}^d$,

$$\mu(x) = A^{(\mu)}x + \bar{a}^{(\mu)}. \quad (261)$$

Then there exists an up to indistinguishability unique solution process

$$S^{(x_0, \pi_\sigma, \pi_\mu)} : [0, T] \times \Omega \rightarrow \mathbb{R}^d \quad (262)$$

for the SDE in 255.

Proof. Consider Theorem 8.3 in the book of Le Gall [55] for an in-depth proof of the above statement involving all technical details. Below, we present a sketch of the key arguments of the proof.

First, notice, that the fact, that σ and μ are affine linear functions, implies, that σ and μ are globally Lipschitz continuous and at most linearly growing. More specifically, one can show with elementary methods, that there must be a constant $K \in (0, \infty)$, such that, for all $x, y \in \mathbb{R}^d$, it holds that

$$\|\sigma(x) - \sigma(y)\| + \|\mu(x) - \mu(y)\| \leq K(\|x - y\|) \quad (263)$$

and

$$\|\sigma(x)\| + \|\mu(x)\| \leq K(1 + \|x\|)^{18}. \quad (264)$$

This means, that the necessary prerequisites for Theorem 8.3 in [55] are fulfilled.

In order to show the existence of a solution for the SDE given in 255, it is possible to use a stochastic version of Picard's approximation method¹⁹. We recursively define a sequence of stochastic processes $(S_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}})_{k \in \mathbb{N}_{\geq 0}}$ via

$$S_t^{(x_0, \pi_\sigma, \pi_\mu)^{(0)}} := x_0, \quad (265)$$

$$S_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k+1)}} := x_0 + \int_0^t \mu(S_s^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}) ds + \int_0^t \sigma(S_s^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}) dB_s. \quad (266)$$

By induction, one can show, that, for all $k \in \mathbb{N}_{\geq 0}$, the process $S^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}$ is well-defined and it holds, that

$$S^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}} \in \mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d). \quad (267)$$

Note that, if $\tilde{\omega}$ is in the $\mathbb{P}$ -null set on which $t \mapsto S_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\tilde{\omega})$ might be discontinuous (and therefore possibly not Lebesgue-integrable), the right-hand

¹⁸For a detailed proof of these facts about affine linear functions, take a look at Corollary 2.9 and Corollary 2.10 in the article of Grohs, Hornung, Jentzen and Wurstemberger [10]

¹⁹The deterministic version of the Picard iteration method is already used in the famous Picard-Lindelöf theorem, which proves the existence of unique solutions of ordinary differential equations associated with Lipschitz continuous vector fields.

side of Recursion 291 is tacitly set to 0. Also, note that the Ito integral in Recursion 291 is here seen a concrete $\mathbb{P}$ -a.s. continuous stochastic process in t rather than as an equivalence class of indistinguishable stochastic processes. Now, one can proceed by proving, that there exists an event $A^{(x_0, \pi_\sigma, \pi_\mu)} \in \mathcal{G}$ with probability $\mathbb{P}(A^{(x_0, \pi_\sigma, \pi_\mu)}) = 1$, such that, for all $\tilde{\omega} \in A^{(x_0, \pi_\sigma, \pi_\mu)}$ and all $k \in \mathbb{N}_{\geq 0}$, the sample path function

$$S^{(x_0, \pi_\sigma, \pi_\mu)(k)}(\tilde{\omega}) : [0, T] \rightarrow \mathbb{R}^d, \quad t \mapsto S_t^{(x_0, \pi_\sigma, \pi_\mu)(k)}(\tilde{\omega}), \quad (268)$$

is continuous and the sequence of sample path functions $(S^{(x_0, \pi_\sigma, \pi_\mu)(k)}(\tilde{\omega}))_{k \in \mathbb{N}_{\geq 0}}$ converges uniformly on $[0, T]$. This, along with Lemma 4.1 implies, that the function

$$S^{(x_0, \pi_\sigma, \pi_\mu)} : [0, T] \times \Omega \rightarrow \mathbb{R}^d, \quad S_t^{(x_0, \pi_\sigma, \pi_\mu)}(\omega) := \limsup_{k \rightarrow \infty} S_t^{(x_0, \pi_\sigma, \pi_\mu)(k)}(\omega), \quad (269)$$

is a stochastic process²⁰ in $\mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ such that, for all $\tilde{\omega} \in A^{(x_0, \pi_\sigma, \pi_\mu)}$ and all $t \in [0, T]$, it holds that

$$S_t^{(x_0, \pi_\sigma, \pi_\mu)}(\tilde{\omega}) = \lim_{k \rightarrow \infty} S_t^{(x_0, \pi_\sigma, \pi_\mu)(k)}(\tilde{\omega}). \quad (270)$$

Now, fixing $t \in [0, T]$ and passing to the limit $k \rightarrow \infty$ in recursion

$$S_t^{(x_0, \pi_\sigma, \pi_\mu)(k+1)} = x_0 + \int_0^t \mu(S_s^{(x_0, \pi_\sigma, \pi_\mu)(k)}) ds + \int_0^t \sigma(S_s^{(x_0, \pi_\sigma, \pi_\mu)(k)}) dB_s \quad (271)$$

shows, that $S^{(x_0, \pi_\sigma, \pi_\mu)}$ is a solution process of the SDE in 255. The fact, that μ and σ are both globally Lipschitz, assures, that the stochastic integrals on the right-hand side of Recursion 271 converge in probability to the desired values.

Now let $\bar{S}^{(x_0, \pi_\sigma, \pi_\mu)}$ be another solution process of the SDE in 255. It follows from a version of Gronwall's Lemma (see Le Gall [55], Lemma 8.4), that, for all $t \in [0, T]$, it holds that

$$\mathbb{E}[\|S_t^{(x_0, \pi_\sigma, \pi_\mu)} - \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)}\|^2] = 0. \quad (272)$$

This implies, that $S^{(x_0, \pi_\sigma, \pi_\mu)(k+1)}$ and $\bar{S}^{(x_0, \pi_\sigma, \pi_\mu)(k+1)}$ are modifications of each other and since both processes are $\mathbb{P}$ -a.s. continuous, this is sufficient to conclude, that $S^{(x_0, \pi_\sigma, \pi_\mu)}$ and $\bar{S}^{(x_0, \pi_\sigma, \pi_\mu)}$ are indistinguishable. $\square$

Remark 4.13 (Extending the Time Interval for SDEs). Let $\tilde{T} > T$ and let $S^{(x_0, \pi_\sigma, \pi_\mu)(T)}$ and $S^{(x_0, \pi_\sigma, \pi_\mu)(\tilde{T})}$ be the (up to indistinguishability) unique solution processes of the same SDE on the time intervals $[0, T]$ and $[0, \tilde{T}]$, respectively. Then it immediately follows from the properties of $S^{(x_0, \pi_\sigma, \pi_\mu)(T)}$ and $S^{(x_0, \pi_\sigma, \pi_\mu)(\tilde{T})}$ (see Theorem 4.2 and Definition 4.6), that $S^{(x_0, \pi_\sigma, \pi_\mu)(T)}$ and $S^{(x_0, \pi_\sigma, \pi_\mu)(\tilde{T})}$ are indistinguishable on $[0, T]$.

²⁰Note that, without Lemma 4.1, it is not immediately obvious, that $S^{(x_0, \pi_\sigma, \pi_\mu)}$ is indeed $(\mathcal{G}_t)_{t \in [0, T]}$ -adapted.

We are now finally ready to present the Feynman-Kac formula, which establishes an important link between stochastic differential equations and Kolmogorov partial differential equations and merges the results from Theorem 4.1 and Theorem 4.2.

Theorem 4.3 (Feynman-Kac Formula). *Let*

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0 \quad (273)$$

be a Kolmogorov partial differential equation with boundary condition

$$u(0, x) = \varphi(x) \quad (274)$$

and let

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T], \quad (275)$$

be an associated stochastic differential equation.

Now recall our assumptions on σ, μ and φ from Setting 4.1, which we now restate for the convenience of the reader. Let

$$\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}, \quad (276)$$

$$\mu : \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad (277)$$

be affine linear transformations, i.e. let there be $d+1$ matrices

$$\pi_\sigma := (A_1^{(\sigma)}, \dots, A_d^{(\sigma)}, \bar{A}^{(\sigma)}) \in (\mathbb{R}^{d \times d})^{d+1} \quad (278)$$

such that, for all $x \in \mathbb{R}^d$,

$$\sigma(x) = (A_1^{(\sigma)} x \mid \dots \mid A_d^{(\sigma)} x) + \bar{A}^{(\sigma)}, \quad (279)$$

where $A_i^{(\sigma)} x$ describes the i -th column of the matrix $\sigma(x) \in \mathbb{R}^{d \times d}$, and let there be a matrix-vector pair

$$\pi_\mu := (A^{(\mu)}, \bar{a}^{(\mu)}) \in \mathbb{R}^{d \times d} \times \mathbb{R}^d \quad (280)$$

such that, for all $x \in \mathbb{R}^d$,

$$\mu(x) = A^{(\mu)} x + \bar{a}^{(\mu)}, \quad (281)$$

and, moreover, let

$$\varphi : \mathbb{R}^d \rightarrow \mathbb{R} \quad (282)$$

be continuous and at most polynomially growing, i.e. let φ be continuous and let there be a constant $C \in [1, \infty)$ such that, for all $x \in \mathbb{R}^d$ it holds that

$$|\varphi(x)| \leq C(1 + \|x\|)^C. \quad (283)$$

Then there exists exactly one function

$$u^{(\pi_\sigma, \pi_\mu)} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R} \quad (284)$$

on the set $[0, T] \times \mathbb{R}^d$ which has the following properties:

- $u^{(\pi_\sigma, \pi_\mu)} \in \mathcal{C}([0, T] \times \mathbb{R}^d, \mathbb{R})$ is continuous on $[0, T] \times \mathbb{R}^d$,
- for all $x \in \mathbb{R}^d$, it holds that $u^{(\pi_\sigma, \pi_\mu)}(0, x) = \varphi(x)$,
- $u^{(\pi_\sigma, \pi_\mu)}|_{(0, T) \times \mathbb{R}^d}$ is a viscosity solution of Kolmogorov equation 273 on the open set $(0, T) \times \mathbb{R}^d$, and
- $\inf_{q \in (0, \infty)} \sup_{(t, x) \in [0, T] \times \mathbb{R}^d} \frac{|u^{(\pi_\sigma, \pi_\mu)}(t, x)|}{1 + \|x\|^q} < \infty$.

Furthermore, there exists an up to indistinguishability unique solution process

$$S^{(x_0, \pi_\sigma, \pi_\mu)} : [0, T] \times \Omega \rightarrow \mathbb{R}^d \quad (285)$$

for the SDE in 275. A link between $u^{(\pi_\sigma, \pi_\mu)}$ and $S^{(x_0, \pi_\sigma, \pi_\mu)}$ is given by the Feynman-Kac formula, which assures, that, for all $t \in [0, T]$, it holds that

$$u^{(\pi_\sigma, \pi_\mu)}(t, x_0) = \mathbb{E}[\varphi(S_t^{(x_0, \pi_\sigma, \pi_\mu)})]. \quad (286)$$

Proof. For a proof of the existence and uniqueness claims associated with the process $S^{(x_0, \pi_\sigma, \pi_\mu)}$, take a look at the proof of Theorem 4.2.

For a proof of the existence and uniqueness claims associated with $u^{(\pi_\sigma, \pi_\mu)}$, consider Corollary 2.23 in the article of Grohs, Hornung, Jentzen and Wurstemberger [10] as well as the proof of Theorem 4.1.

For a proof of the Feynman-Kac formula, again consider Corollary 2.23 in the article of Grohs, Hornung, Jentzen and Wurstemberger [10]. Note that, Corollary 2.23 in [10] only assures the validity of the Feynman-Kac formula 286 for the special case $t = T$. In our setting, however, it follows from Theorems 4.1 and 4.2 as well as Remarks 4.7 and 4.13, that if Equation 286 is valid for $t = T$, it must in fact be valid for all $t \in [0, T]$. $\square$

4.4 Kolmogorov Equations with Variable Coefficient Parameters as a Statistical Learning Problem

Now, in the final section of this chapter, we are ready for our theoretical attempt to rephrase Kolmogorov partial differential equations as statistical learning problems. We start by extending our framework to include random variables which model our training data.

Setting 4.3. In addition to Setting 4.1 and Setting 4.3, for the rest of Section 4,

- for any topological space $(U, \mathcal{O})$ and nonempty set $\tilde{U} \subseteq U$, let $\mathcal{O}_U(\tilde{U})$ be the trace topology induced on $\tilde{U}$ by $(U, \mathcal{O})$,
- for any metric space (U, h) , whose metric $h(x, y) = \|\tilde{x} - \tilde{y}\|$ is induced by a norm $\|\tilde{\cdot}\| : U \rightarrow \mathbb{R}_{\geq 0}$, let $\mathcal{O}_{\|\tilde{\cdot}\|}(U)$ be the topology induced by h on U ,

- let $v, w \in \mathbb{R}$ be fixed real numbers with $a < b$ and let $[v, w]^d \subseteq \mathbb{R}^d$ be a compact subspace of the topological space $(\mathbb{R}^d, \mathcal{O}_{\|\cdot\|}(\mathbb{R}^d))$ in form of a d -dimensional hypercube,
- let the space of matrices $\mathbb{R}^{d \times d}$ be equipped with the topology $\mathcal{O}_{\|\cdot\|}(\mathbb{R}^{d \times d})$ induced by the Frobenius norm $\|\cdot\| : \mathbb{R}^{d \times d} \rightarrow \mathbb{R}_{\geq 0}$ ²¹,
- let $D_{\Pi_\sigma} \subseteq (\mathbb{R}^{d \times d})^{d+1}$ be a compact subspace of the topological space of sequences of $d+1$ matrices $((\mathbb{R}^{d \times d})^{d+1}, \bigotimes_{i=1}^{d+1} \mathcal{O}_{\|\cdot\|}(\mathbb{R}^{d \times d}))$,
- let $D_{\Pi_\mu} \subseteq \mathbb{R}^{d \times d} \times \mathbb{R}^d$ be a compact subspace of the topological space of matrix-vector pairs $(\mathbb{R}^{d \times d} \times \mathbb{R}^d, \mathcal{O}_{\|\cdot\|}(\mathbb{R}^{d \times d}) \otimes \mathcal{O}_{\|\cdot\|}(\mathbb{R}^d))$,
- let $[0, T] \subseteq \mathbb{R}$ be a compact subspace of the topological space $(\mathbb{R}, \mathcal{O}_{|\cdot|}(\mathbb{R}))$ in form of a compact interval,
- let $[v, w]^d, D_{\Pi_\sigma}, D_{\Pi_\mu}$ and $[0, T]$ be equipped with their canonical trace topologies as well as with their associated canonical Borel σ -algebras $\mathcal{B}([v, w]^d), \mathcal{B}(D_{\Pi_\sigma}), \mathcal{B}(D_{\Pi_\mu})$ and $\mathcal{B}([0, T])$,
- let $D_{(X_0, \Pi_\sigma, \Pi_\mu)} := [v, w]^d \times D_{\Pi_\sigma} \times D_{\Pi_\mu}$ be the (necessarily compact ²²) product topological space of $[v, w]^d, D_{\Pi_\sigma}$ and D_{Π_μ} , equipped with the canonical product topology $\mathcal{O}_{\mathbb{R}^d}([v, w]^d) \otimes \mathcal{O}_{(\mathbb{R}^{d \times d})^{d+1}}(D_{\Pi_\sigma}) \otimes \mathcal{O}_{\mathbb{R}^{d \times d} \times \mathbb{R}^d}(D_{\Pi_\mu})$ and the associated canonical Borel σ -algebra $\mathcal{B}(D_{(X_0, \Pi_\sigma, \Pi_\mu)})$,
- let $\Omega \times [0, T]$ be equipped with its canonical product σ -algebra $\mathcal{G} \otimes \mathcal{B}([0, T])$,
- let $D_\Pi := [v, w]^d \times D_{\Pi_\sigma} \times D_{\Pi_\mu} \times [0, T]$ be the (necessarily compact) product topological space of $[v, w]^d, D_{\Pi_\sigma}, D_{\Pi_\mu}$ and $[0, T]$, equipped with the canonical product topology $\mathcal{O}_{\mathbb{R}^d}([v, w]^d) \otimes \mathcal{O}_{(\mathbb{R}^{d \times d})^{d+1}}(D_{\Pi_\sigma}) \otimes \mathcal{O}_{\mathbb{R}^{d \times d} \times \mathbb{R}^d}(D_{\Pi_\mu}) \otimes \mathcal{O}_{\mathbb{R}}([0, T])$ and the associated canonical Borel σ -algebra $\mathcal{B}(D_\Pi)$ ²³,
- let $\Pi := (X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T}) : \Omega \rightarrow D_\Pi$ be a $\mathcal{G}_0/\mathcal{B}(D)$ -measurable random variable, called *predictor variable*, whereby $X_0 : \Omega \rightarrow [v, w]^d$ consists of a $\mathcal{G}_0/\mathcal{B}([v, w]^d)$ -measurable random vector, $\Pi_\sigma : \Omega \rightarrow D_{\Pi_\sigma}$ consists of a $\mathcal{G}_0/\mathcal{B}(D_{\Pi_\sigma})$ -measurable sequence of $d+1$ random matrices, $\Pi_\mu : \Omega \rightarrow D_{\Pi_\mu}$ consists of a $\mathcal{G}_0/\mathcal{B}(D_{\Pi_\mu})$ -measurable pair of a random matrix and a random vector, and $\mathcal{T} : \Omega \rightarrow [0, T]$ is a $\mathcal{G}_0/\mathcal{B}([0, T])$ -measurable random nonnegative number, and where the random variables X_0, Π_σ, Π_μ and $\mathcal{T}$ are pairwise independent, and
- let us strengthen our assumptions on

$$\varphi : \mathbb{R}^d \rightarrow \mathbb{R}, \quad (287)$$

²¹It is trivial to see, that $\mathbb{R}^{d \times d}$ are isomorphic $\mathbb{R}^{d^2}$ in a topological sense.

²²It is a basic result from topology, that canonical products of compact topological spaces are again compact.

²³Note, that, since all involved topological spaces are second-countable, $\mathcal{B}(D_\Pi)$ is necessarily identical to the canonical product of the individual Borel σ -algebras $\mathcal{B}([v, w]^d) \otimes \mathcal{B}(D_{\Pi_\sigma}) \otimes \mathcal{B}(D_{\Pi_\mu}) \otimes \mathcal{B}([0, T])$. An analogous statement holds for $\mathcal{B}(D_{(X_0, \Pi_\sigma, \Pi_\mu)})$

by assuming that φ is continuous and bounded (instead of just continuous and at most polynomially growing), i.e. let us assume that φ is continuous and that there exists a constant $C_\varphi \in (0, \infty)$, such that, for all $x \in \mathbb{R}^d$, it holds that

$$|\varphi(x)| \leq C_\varphi, \quad (288)$$

and

- for $p \in \mathbb{N}_{\geq 0}$, let

$$\mathcal{L}^p(\Omega, \mathbb{R}, \mathbb{P}) := \{f : \Omega \rightarrow \mathbb{R} \mid f \text{ is } \mathcal{G}/\mathcal{B}(\mathbb{R})\text{-measurable and } \int_{\Omega} |f|^p d\mathbb{P} < \infty\} \quad (289)$$

be the space of $\mathcal{G}/\mathcal{B}(\mathbb{R})$ -measurable functions which are p -th power integrable with respect to the measure P on $(\Omega, \mathcal{G})$.

Remark 4.14 (Measurability of Picard Iteration Processes with Variable Initial Data). Assume the setting and notation of Theorem 4.2 (and its proof) and consider again the sequence of Picard iteration processes in the space $\mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ given by

$$S_t^{(x_0, \pi_\sigma, \pi_\mu)^{(0)}} := x_0, \quad (290)$$

$$S_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k+1)}} := x_0 + \int_0^t \mu(S_s^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}) ds + \int_0^t \sigma(S_s^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}) dB_s. \quad (291)$$

Now, temporarily, set $d = 1$. Since σ and μ are affine-linear functions, one can show in this case via induction, that each process $S_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}$ can be expressed in an (up to indistinguishability) equivalent form

$$\bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}} = \sum_{i=1}^{q_k} c_i^{(k)}(x_0, \pi_\sigma, \pi_\mu) h_i^{(k)}(t) + g_i^{(k)}(x_0, \pi_\sigma, \pi_\mu) H_t^{(i)^{(k)}} \quad (292)$$

for some $q_k \in \mathbb{N}$, some continuous functions

$$\begin{aligned} c_{(1)}^{(k)}, \dots, c_{(q_k)}^{(k)} &: D_{(X_0, \Pi_\sigma, \Pi_\mu)} \rightarrow \mathbb{R}, \\ g_{(1)}^{(k)}, \dots, g_{(q_k)}^{(k)} &: D_{(X_0, \Pi_\sigma, \Pi_\mu)} \rightarrow \mathbb{R}, \\ h_{(1)}^{(k)}, \dots, h_{(q_k)}^{(k)} &: [0, T] \rightarrow \mathbb{R}, \end{aligned} \quad (293)$$

as well as some $\mathbb{P}$ -a.s. continuous processes

$$H^{(1)^{(k)}}, \dots, H^{(q_k)^{(k)}} \in \mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}), \quad (294)$$

which are independent of the parameters $(x_0, \pi_\sigma, \pi_\mu)$. Again, we silently agree to set $\bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\tilde{\omega}) = 0$, if $\tilde{\omega} \in \Omega$ is in the $\mathbb{P}$ -null set on which at least one of the sample paths $t \mapsto H_t^{(i)^{(k)}}(\tilde{\omega})$ is discontinuous. Note, that $q_k \in \mathbb{N}$, the functions $c_{(i)}^{(k)}, g_{(i)}^{(k)}, h_{(i)}^{(k)}$ and the processes $H^{(i)^{(k)}}$ are not necessarily unique,

but are treated as fixed entities from now on. Also note, that, when writing the Picard iteration processes in this form, it still holds, that the limiting process

$$\Omega \times [0, T] \ni (\omega, t) \mapsto \limsup_{k \rightarrow \infty} \text{fin} \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\omega) \in \mathbb{R} \quad (295)$$

is a solution of the SDE from Theorem 4.2.

It is now obvious from the representation of $\bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}$ given in Equation 292, that, for all fixed $k \in \mathbb{N}_{\geq 1}$ and $(\omega, t) \in \Omega \times [0, T]$, the map

$$D_{(X_0, \Pi_\sigma, \Pi_\mu)} \ni (x_0, \pi_\sigma, \pi_\mu) \mapsto \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\omega) \in \mathbb{R} \quad (296)$$

is continuous. On the other hand, for all fixed $k \in \mathbb{N}_{\geq 1}$ and $(x_0, \pi_\sigma, \pi_\mu) \in D_{(X_0, \Pi_\sigma, \Pi_\mu)}$, the process $\bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}} \in \mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R})$ is $\mathbb{P}$ -a.s. continuous as well as $\mathcal{G}_t$ -adapted, and therefore automatically *measurable*²⁴, which means, that the map

$$\Omega \times [0, T] \ni (\omega, t) \mapsto \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\omega) \in \mathbb{R} \quad (297)$$

is $(\mathcal{G} \otimes \mathcal{B}([0, T]))/\mathcal{B}(\mathbb{R})$ -measurable. Combining the continuity of 296 and the measurability of 297, we can conclude, that the map

$$(\Omega \times [0, T]) \times D_{(X_0, \Pi_\sigma, \Pi_\mu)} \ni ((\omega, t), (x_0, \pi_\sigma, \pi_\mu)) \mapsto \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\omega) \in \mathbb{R} \quad (298)$$

is a Carathodory function, and therefore $((\mathcal{G} \otimes \mathcal{B}([0, T])) \otimes \mathcal{B}(D_{(X_0, \Pi_\sigma, \Pi_\mu)}))/\mathcal{B}(\mathbb{R})$ -measurable²⁵.

With similar arguments (but at the price of a much heavier notation), one can show, that it also holds for general $d \in \mathbb{N}_{\geq 1}$, that the map

$$(\Omega \times [0, T]) \times D_{(X_0, \Pi_\sigma, \Pi_\mu)} \ni ((\omega, t), (x_0, \pi_\sigma, \pi_\mu)) \mapsto \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\omega) \in \mathbb{R}^d \quad (299)$$

is $((\mathcal{G} \otimes \mathcal{B}([0, T])) \otimes \mathcal{B}(D_{(X_0, \Pi_\sigma, \Pi_\mu)}))/\mathcal{B}(\mathbb{R}^d)$ -measurable. Using Lemma 4.1, we can even conclude, that the map

$$(\Omega \times [0, T]) \times D_{(X_0, \Pi_\sigma, \Pi_\mu)} \ni ((\omega, t), (x_0, \pi_\sigma, \pi_\mu)) \mapsto \limsup_{k \rightarrow \infty} \text{fin} \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\omega) \in \mathbb{R}^d \quad (300)$$

is $((\mathcal{G} \otimes \mathcal{B}([0, T])) \otimes \mathcal{B}(D_{(X_0, \Pi_\sigma, \Pi_\mu)}))/\mathcal{B}(\mathbb{R}^d)$ -measurable.

Definition 4.12 (Solution Process with Variable Initial Data). *Assume the setting of Theorem 4.2 (and its proof) and of Remark 4.14. Then we define a function $\bar{S}$ via*

$$\bar{S} : \Omega \times D_\Pi \rightarrow \mathbb{R}^d, \quad \bar{S}(\omega, (x_0, \pi_\sigma, \pi_\mu, t)) = \limsup_{k \rightarrow \infty} \text{fin} \bar{S}_t^{(x_0, \pi_\sigma, \pi_\mu)^{(k)}}(\omega). \quad (301)$$

$\bar{S}$ is called solution process with variable initial data.

²⁴See Le Gall [55], Proposition 3.4.

²⁵ This is an application of Lemma 4.51 in the book of Aliprantis [12]. Note, that the necessary assumptions of Lemma 4.51 are fulfilled, since $D_{(X_0, \Pi_\sigma, \Pi_\mu)}$ is a separable metrizable space and $\mathbb{R}$ is a metrizable space. For more information on Carathodory functions, consider Section 4.10 in [12].

Proposition 4.3 (Properties of Solution Process with Variable Initial Data).
The solution process with variable initial data

$$\bar{S} : \Omega \times D_{\Pi} \rightarrow \mathbb{R}^d, \quad \bar{S}(\omega, (x_0, \pi_{\sigma}, \pi_{\mu}, t)) = \limsup_{k \rightarrow \infty} \text{fin}_{(\mathbb{R}^d)} \bar{S}_t^{(x_0, \pi_{\sigma}, \pi_{\mu})(k)}(\omega), \quad (302)$$

is a $(\mathcal{G} \otimes \mathcal{B}(D_{\Pi}))/\mathcal{B}(\mathbb{R}^d)$ -measurable function, such that, for each fixed triple $(\tilde{x}_0, \tilde{\pi}_{\sigma}, \tilde{\pi}_{\mu}) \in D_{(X_0, \Pi_{\sigma}, \Pi_{\mu})}$, the process

$$\Omega \times [0, T] \ni (\omega, t) \mapsto \bar{S}(\omega, (\tilde{x}_0, \tilde{\pi}_{\sigma}, \tilde{\pi}_{\mu}, t)) \in \mathbb{R}^d \quad (303)$$

is an element of $\mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ and a solution process for the stochastic differential equation

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T], \quad (304)$$

where σ and μ are affine linear functions defined by the parameters $\tilde{\pi}_{\sigma}$ and $\tilde{\pi}_{\mu}$ (see Setting 4.1).

Proof. The mapping

$$\Omega \times D_{\Pi} \ni (\omega, (x_0, \pi_{\sigma}, \pi_{\mu}, t)) \mapsto \bar{S}(\omega, (x_0, \pi_{\sigma}, \pi_{\mu}, t)) \in \mathbb{R}^d \quad (305)$$

can be sequentially decomposed into the mappings

$$\Omega \times D_{\Pi} \ni (\omega, (x_0, \pi_{\sigma}, \pi_{\mu}, t)) \mapsto ((\omega, t), (x_0, \pi_{\sigma}, \pi_{\mu})) \in (\Omega \times [0, T]) \times D_{(X_0, \Pi_{\sigma}, \Pi_{\mu})}, \quad (306)$$

and

$$(\Omega \times [0, T]) \times D_{(X_0, \Pi_{\sigma}, \Pi_{\mu})} \ni ((\omega, t), (x_0, \pi_{\sigma}, \pi_{\mu})) \mapsto \bar{S}(\omega, (x_0, \pi_{\sigma}, \pi_{\mu}, t)) \in \mathbb{R}^d. \quad (307)$$

Clearly, mapping 306 is $(\mathcal{G} \otimes \mathcal{B}(D_{\Pi}))/((\mathcal{G} \otimes \mathcal{B}([0, T])) \otimes \mathcal{B}(D_{(X_0, \Pi_{\sigma}, \Pi_{\mu})}))$ -measurable. Moreover, the arguments in Remark 4.14 imply, that mapping 307 is $((\mathcal{G} \otimes \mathcal{B}([0, T])) \otimes \mathcal{B}(D_{(X_0, \Pi_{\sigma}, \Pi_{\mu})}))/\mathcal{B}(\mathbb{R}^d)$ -measurable. Since the composition of measurable maps is again measurable, it follows, that $\bar{S}$ is $(\mathcal{G} \otimes \mathcal{B}(D_{\Pi}))/\mathcal{B}(\mathbb{R}^d)$ -measurable. Furthermore, it follows from the proof of Theorem 4.2 and Remark 4.14, that, for each triple $(\tilde{x}_0, \tilde{\pi}_{\sigma}, \tilde{\pi}_{\mu}) \in D_{(X_0, \Pi_{\sigma}, \Pi_{\mu})}$, the sequence of Picard iteration processes in $\mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ given by

$$(\Omega \times [0, T]) \times D_{(X_0, \Pi_{\sigma}, \Pi_{\mu})} \ni ((\omega, t), (x_0, \pi_{\sigma}, \pi_{\mu})) \mapsto \bar{S}_t^{(x_0, \pi_{\sigma}, \pi_{\mu})(k)}(\omega) \in \mathbb{R}^d \quad (308)$$

converges $\mathbb{P}$ -a.s. uniformly in t towards a solution process of SDE 304. This implies, that the limiting process

$$\Omega \times [0, T] \ni (\omega, t) \mapsto \limsup_{k \rightarrow \infty} \text{fin}_{(\mathbb{R}^d)} \bar{S}_t^{(x_0, \pi_{\sigma}, \pi_{\mu})(k)}(\omega) := \bar{S}(\omega, (\tilde{x}_0, \tilde{\pi}_{\sigma}, \tilde{\pi}_{\mu}, t)) \in \mathbb{R}^d \quad (309)$$

is also an element of $\mathcal{S}_{\text{cont}}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$, as well as a solution process of SDE 304. $\square$

Definition 4.13. *The function*

$$\Omega \ni \omega \mapsto \varphi(\bar{S}(\omega, \Pi(\omega))) \in \mathbb{R} \quad (310)$$

is called target variable and is also denoted as $Y_{\text{target}}(\omega) := \varphi(\bar{S}(\omega, \Pi(\omega)))$.

Proposition 4.4 (Properties of Target Variable). *The target variable Y_{target} is a $\mathcal{G}/\mathcal{B}(\mathbb{R})$ -measurable random variable with finite expectation and finite variance, i.e. $Y_{\text{target}} \in \mathcal{L}^2(\Omega, \mathbb{R}, \mathbb{P})$.*

Proof. The $\mathcal{G}/\mathcal{B}(\mathbb{R})$ -measurability of

$$\Omega \ni \omega \mapsto \varphi(\bar{S}(\omega, \Pi(\omega))) \in \mathbb{R} \quad (311)$$

follows from the continuity (and therefore $\mathcal{B}(\mathbb{R}^d)/\mathcal{B}(\mathbb{R})$ -measurability) of φ , the $(\mathcal{G} \otimes \mathcal{B}(D_\Pi))/\mathcal{B}(\mathbb{R}^d)$ -measurability of $\bar{S}$, and the $\mathcal{G}/(\mathcal{G} \otimes \mathcal{B}(D_\Pi))$ -measurability of the map

$$\Omega \ni \omega \mapsto (\omega, \Pi(\omega)). \quad (312)$$

Finally, the finiteness of the variance and expectation of $\omega \mapsto \varphi(\bar{S}(\omega, \Pi(\omega)))$ is a direct consequence of our boundedness-assumption on φ . $\square$

We are now almost ready to propose the main theorem 4.4 of Section 4, whose aim it is, to give a theoretical justification for solving Kolmogorov PDEs with variable x_0, σ, μ via statistical learning. We also propose an (as of yet unproven) technical Lemma, which, under certain additional continuity assumptions on $\bar{S}$, can potentially be used for a proof of the main theorem 4.4.

Lemma 4.2 (Conditional Expectation of Product Space Function). *Let*

- $\tilde{D}$ be a compact, separable, metrizable topological space with Borel σ -algebra $\mathcal{B}(\tilde{D})$,
- $\tilde{g} : \Omega \times \tilde{D} \rightarrow \mathbb{R}$ be a bounded $(\mathcal{G} \otimes \mathcal{B}(\tilde{D}))/\mathcal{B}(\mathbb{R})$ -measurable function, such that, for all fixed $\omega \in \Omega$, the map $\tilde{D} \ni \pi \mapsto \tilde{g}(\omega, \pi) \in \mathbb{R}$ is continuous²⁶,
- $\tilde{\Pi} : \Omega \rightarrow \tilde{D}$ be a $\mathcal{G}/\mathcal{B}(\tilde{D})$ -measurable random variable,
- $\tilde{Y} : \Omega \rightarrow \mathbb{R}$ be a $\mathcal{G}/\mathcal{B}(\mathbb{R})$ -measurable random variable, defined by $\tilde{Y}(\omega) := \tilde{g}(\omega, \tilde{\Pi}(\omega))$,
- $\tilde{s} : \tilde{D} \rightarrow \mathcal{L}^2(\Omega, \mathbb{R}, \mathbb{P})$ be defined by $\tilde{s}(\pi)(\omega) := \tilde{g}(\omega, \pi)$,
- $\tilde{E} : \tilde{D} \rightarrow \mathbb{R}$ be defined by $\tilde{E}(\pi) = \mathbb{E}[\tilde{s}(\pi)]$, and
- for all $\pi \in \tilde{D}$, let $\tilde{s}(\pi)$ be independent of $\tilde{\Pi}$.

Then $\tilde{E}$ is $\mathcal{B}(\tilde{D})/\mathcal{B}(\mathbb{R})$ -measurable and $\tilde{E} \circ \tilde{\Pi}$ is a version of the $\mathbb{P}$ -unique conditional expectation $\mathbb{E}[\tilde{Y} \mid \tilde{\Pi}]$.

²⁶In other words, let $\tilde{g}$ be a bounded Carathodory function.

Proof (Incomplete). First, note, that $\tilde{E} : \tilde{D} \rightarrow \mathbb{R}$ can be written as

$$\tilde{D} \ni \pi \mapsto \int_{\Omega} \tilde{g}(\omega, \pi) d\mathbb{P}(\omega) \in \mathbb{R} \quad (313)$$

Since g is bounded, and therefore integrable on the finite product measure space $(\Omega \times \tilde{D}, \mathcal{G} \otimes \mathcal{B}(\tilde{D}), \mathbb{P} \otimes \mathbb{P}_{\tilde{\Pi}})$ ²⁷, Fubini's theorem guarantees the $\mathcal{B}(\tilde{D})/\mathcal{B}(\mathbb{R})$ -measurability of the map given in 313, and therefore the $\mathcal{B}(\tilde{D})/\mathcal{B}(\mathbb{R})$ -measurability of $\tilde{E}$.

A proof for the claim, that $\tilde{E} \circ \tilde{\Pi}$ is a version of the $\mathbb{P}$ -unique conditional expectation $\mathbb{E}[\tilde{Y} \mid \tilde{\Pi}]$ is still outstanding and needs to be delivered to complete the proof. □

Theorem 4.4 (Kolmogorov Equations with Variable Coefficient Parameters as Mathematical Learning Problems). *Let*

$$\sigma : \mathbb{R}^d \rightarrow \mathbb{R}^{d \times d}, \quad (314)$$

$$\mu : \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad (315)$$

be affine linear transformations with parameters in D_{π_σ} and D_{π_μ} , i.e. let there be $d + 1$ matrices

$$\pi_\sigma := (A_1^{(\sigma)}, \dots, A_d^{(\sigma)}, \bar{A}^{(\sigma)}) \in D_{\pi_\sigma} \quad (316)$$

such that, for all $x \in \mathbb{R}^d$,

$$\sigma(x) = (A_1^{(\sigma)}x \mid \dots \mid A_d^{(\sigma)}x) + \bar{A}^{(\sigma)}, \quad (317)$$

where $A_i^{(\sigma)}x$ describes the i -th column of the matrix $\sigma(x) \in \mathbb{R}^{d \times d}$, and let there be a matrix-vector pair

$$\pi_\mu := (A^{(\mu)}, \bar{a}^{(\mu)}) \in D_{\pi_\mu} \quad (318)$$

such that, for all $x \in \mathbb{R}^d$,

$$\mu(x) = A^{(\mu)}x + \bar{a}^{(\mu)}, \quad (319)$$

Moreover, let

$$\varphi : \mathbb{R}^d \rightarrow \mathbb{R} \quad (320)$$

be continuous and bounded (as assumed in Setting 4.3), i.e. let φ be continuous and let there be a constant $C_\varphi \in (0, \infty)$ such that, for all $x \in \mathbb{R}^d$ it holds that

$$|\varphi(x)| \leq C_\varphi. \quad (321)$$

Consider the Kolmogorov partial differential equation

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0 \quad (322)$$

²⁷Here, $\mathbb{P}_{\tilde{\Pi}} := \mathbb{P} \circ \tilde{\Pi}^{-1}$ is the pushforward measure of $\tilde{\Pi}$ induced on $(\tilde{D}, \mathcal{B}(\tilde{D}))$.

with boundary condition

$$u(0, x) = \varphi(x). \quad (323)$$

For each fixed pair of parameters $(\pi_\sigma, \pi_\mu) \in D_{\pi_\sigma} \times D_{\pi_\mu}$ associated with σ and μ and each fixed initial function φ with the above properties, there exists exactly one function

$$u^{(\pi_\sigma, \pi_\mu)} : [0, T] \times \mathbb{R}^d \rightarrow \mathbb{R} \quad (324)$$

on the set $[0, T] \times \mathbb{R}^d$ which has the following properties:

- $u^{(\pi_\sigma, \pi_\mu)} \in \mathcal{C}([0, T] \times \mathbb{R}^d, \mathbb{R})$ is continuous on $[0, T] \times \mathbb{R}^d$,
- for all $x \in \mathbb{R}^d$, it holds that $u^{(\pi_\sigma, \pi_\mu)}(0, x) = \varphi(x)$,
- $u^{(\pi_\sigma, \pi_\mu)}|_{(0, T) \times \mathbb{R}^d}$ is a viscosity solution of Kolmogorov equation 273 on the open set $(0, T) \times \mathbb{R}^d$, and
- $\inf_{q \in (0, \infty)} \sup_{(t, x) \in [0, T] \times \mathbb{R}^d} \frac{|u^{(\pi_\sigma, \pi_\mu)}(t, x)|}{1 + \|x\|^q} < \infty$.

Next, let

$$\bar{S} : \Omega \times D_\Pi \rightarrow \mathbb{R}^d \quad (325)$$

be the $(\mathcal{G} \otimes \mathcal{B}(D_\Pi)) / \mathcal{B}(\mathbb{R}^d)$ -measurable solution process with variable initial data from Definition 4.12, such that, for each fixed triple $(x_0, \pi_\sigma, \pi_\mu) \in D_{(x_0, \pi_\sigma, \pi_\mu)}$, the process

$$\Omega \times [0, T] \ni (\omega, t) \mapsto \bar{S}(\omega, (x_0, \pi_\sigma, \pi_\mu, t)) \in \mathbb{R}^d \quad (326)$$

is an element of $\mathcal{S}_{cont}((\Omega, \mathcal{G}, (\mathcal{G}_t)_{t \in [0, T]}, \mathbb{P}), \mathbb{R}^d)$ and a solution process for the stochastic differential equation

$$S_t = x_0 + \int_0^t \mu(S_s) ds + \int_0^t \sigma(S_s) dB_s, \quad t \in [0, T], \quad (327)$$

where σ and μ are affine-linear functions as above, associated with the paramters π_σ and π_μ .

Now, recall the definition of the mathematical learning problem 2.2, the definition of a regression function 2.3 as well as Theorem 2.1 and Remark 2.9 from Section 2. The $\mathcal{G} / \mathcal{B}(\mathbb{R})$ -measurable, square-integrable target variable $Y_{target} \in \mathcal{L}^2(\Omega, \mathbb{R}, \mathbb{P})$ defined by

$$Y_{target} : \Omega \rightarrow \mathbb{R}, \quad Y_{target}(\omega) = \varphi(\bar{S}(\omega, \Pi(\omega))) \quad (328)$$

and the $\mathcal{G} / \mathcal{B}(D_\Pi)$ -measurable predictor variable

$$\Pi := (X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T}) : \Omega \rightarrow D_\Pi \quad (329)$$

form a mathematical learning problem via

$$Z = (\Pi, Y_{target}). \quad (330)$$

Furthermore, the Kolmogorov PDE solution map with variable coefficient parameters

$$\bar{u} : D_\Pi \rightarrow \mathbb{R}, \quad \bar{u}(x_0, \pi_\sigma, \pi_\mu, t) = u^{(\pi_\sigma, \pi_\mu)}(t, x_0), \quad (331)$$

is the $\mathbb{P}_\Pi$ -unique regression function which solves the mathematical learning problem associated with Z , i.e. $\bar{u}$ is $\mathcal{B}(D_\Pi)/\mathcal{B}(\mathbb{R})$ -measurable and it holds for $\mathbb{P}_\Pi$ -almost all $(x_0, \pi_\sigma, \pi_\mu, t) \in D_\Pi$, that

$$\bar{u}((x_0, \pi_\sigma, \pi_\mu, t)) = \mathbb{E}[Y_{\text{target}} \mid \Pi = (x_0, \pi_\sigma, \pi_\mu, t)]. \quad (332)$$

Suggested Proof (Based on Assumptions (†)).

Note: In this proof, we assume the validity of Lemma 4.2, which remains unproven at this point. Furthermore, as an additional property for $\bar{S}$, we assume, that $\bar{S}$ is a Carathodory function, i.e. for all $\omega \in \Omega$, we assume, that the map $D_\Pi \ni \pi \mapsto \bar{S}(\omega, \pi) \in \mathbb{R}^d$ is continuous. Rigorous justifications for the validity both of these assumptions still need to be delivered in order to formally complete the chain of arguments below. We assign the symbol (†) to both of these assumptions to mark where they are used in the proof.

For a proof of the existence and uniqueness claims associated with the viscosity solutions $u^{(\pi_\sigma, \pi_\mu)}$ of Kolmogorov PDEs 322, consider Theorem 4.1.

For a proof of the existence and uniqueness claims associated with the solution processes of stochastic differential equations of the form 327, consider Theorem 4.2.

For the construction (and therefore proof of existence) of the solution process with variable initial data $\bar{S}$ for stochastic differential equations of the form 327 as well as proofs of the essential properties of $\bar{S}$, take a look at Remark 4.14, Definition 4.12 and Proposition 4.3.

By assumption, the predictor variable Π is $\mathcal{G}/\mathcal{B}(D_\Pi)$ -measurable. The claim, that Y_{target} is $\mathcal{G}/\mathcal{B}(\mathbb{R})$ -measurable and an element of $\mathcal{L}^2(\Omega, \mathbb{R}, \mathbb{P})$, follows from Proposition 4.4. Since (D_Π) is compact and equipped with its Borel σ -algebra $\mathcal{B}(D_\Pi)$, all assumptions from Section 2 are fulfilled and $Z = (\Pi, Y_{\text{target}})$ defines a mathematical learning problem. The square-integrability of $Y_{\text{target}} \in \mathcal{L}^2(\Omega, \mathbb{R}, \mathbb{P})$, along with Proposition 2.3, guarantees, that there exists a $\mathbb{P}_\Pi$ -unique $\mathcal{B}(D_\Pi)/\mathcal{B}(\mathbb{R})$ -measurable function

$$\hat{f}_Z : D_\Pi \rightarrow \mathbb{R} \quad (333)$$

such that

$$\hat{f}_Z \circ \Pi : \Omega \rightarrow \mathbb{R} \quad (334)$$

is a version of the $\mathbb{P}$ -unique conditional expectation $\mathbb{E}[Y_{\text{target}} \mid \Pi]$. Furthermore, Theorem 2.1 reveals, that $\hat{f}_Z$ is then the $\mathbb{P}_\Pi$ -unique regression function, which minimizes the least squares error

$$\mathcal{E}_Z(f) := \mathbb{E}[|f(\Pi) - Y_{\text{target}}|^2] \quad (335)$$

amongst all $\mathcal{B}(D_\Pi)/\mathcal{B}(\mathbb{R})$ -measurable functions $f : D_\Pi \rightarrow \mathbb{R}$, and, in this sense, solves the mathematical learning problem associated with Z . It remains to show, that $\bar{u}$ is $\mathcal{B}(D_\Pi)/\mathcal{B}(\mathbb{R})$ -measurable and that $\bar{u} \circ \Pi$ is a version of the $\mathbb{P}$ -unique conditional expectation $\mathbb{E}[Y_{\text{target}} \mid \Pi]$, since this implies, that we can set $\hat{f}_Z := \bar{u}$.

To see, that this is the case, let us first define an auxiliary functions s and E via

$$s : D_\Pi \rightarrow \mathcal{L}(\Omega, \mathbb{R}, \mathbb{P}), \quad s(\pi)(\omega) = \varphi(\bar{S}(\omega, \pi)), \quad (336)$$

and

$$E : D_\Pi \rightarrow \mathbb{R}, \quad E(\pi) = \mathbb{E}[s(\pi)]. \quad (337)$$

Note, that, for all $\pi = (x_0, \pi_\sigma, \pi_\mu, t) \in D_\Pi$, an application of the Feynman-Kac Formula 4.3 leads to

$$\begin{aligned} \bar{u}(\pi) &= \bar{u}(x_0, \pi_\sigma, \pi_\mu, t) = u^{(\pi_\sigma, \pi_\mu)}(t, x_0) \\ &= \mathbb{E}[\varphi(\bar{S}(\cdot, (x_0, \pi_\sigma, \pi_\mu, t)))] = \mathbb{E}[s(\pi)] = E(\pi), \end{aligned} \quad (338)$$

which means, that $\bar{u}$ and E are in fact identical.

Now note, that, for all $\pi = (x_0, \pi_\sigma, \pi_\mu, t) \in D_\Pi$, the random variables $s(\pi)$ and Π are independent of each other. To see this, remember that Π is per definition $\mathcal{G}_0/\mathcal{B}(D_\Pi)$ measurable (see Setting 4.2), which means, that $\sigma(\Pi) \subseteq \mathcal{G}_0$. On the other hand, $s(\pi)$ can be shown²⁸ to be $\sigma(B_t)^*/\mathcal{B}(\mathbb{R})$ -measurable, whereby $(\sigma(B_t)^*)_{t \in [0, T]}$ is the canonically completion of the natural filtration $(\sigma(B_t))_{t \in [0, T]}$ generated by the Brownian motion $(B_t)_{t \in [0, T]}$. Since $\sigma(B_t)$ is independent of $\mathcal{G}_0$ (see Setting 4.2), its completion $\sigma(B_t)^*$ is independent of $\mathcal{G}_0$ as well²⁹, which proves the independence of $s(\pi)$ and Π .

This independence, along with our assumptions $(\dagger)$, imply, that we can use Lemma 4.2 with $\tilde{D} := D_\Pi$, $\tilde{g} := \varphi \circ \bar{S}$, $\tilde{\Pi} := \Pi$, $\tilde{Y} := Y_{\text{target}}$, $\tilde{s} := s$ and $\tilde{E} := E$. Lemma 4.2 assures us, that E (and therefore $\bar{u}$) is $\mathcal{B}(D_\Pi)/\mathcal{B}(\mathbb{R})$ -measurable and that $E \circ \Pi$ is a version of the $\mathbb{P}$ -unique conditional expectation $\mathbb{E}[Y_{\text{target}} \mid \Pi]$, which completes the proof. $\square$

²⁸See Theorem 8.3 in the book of Le Gall [55].

²⁹This can be shown, using elementary methods from measure and probability theory, such as the fact that two σ -algebras with independent $\cap$ -stable generators are again independent.

5 Numerical Experiments

This chapter is dedicated towards numerical experiments for the empirical investigation of the validity of our conjectured learning theorem 4.4 for Kolmogorov equations with variable (affine-linear) initial coefficients. We shall put the claims of Theorem 4.4 to the test, in both a low- and a high-dimensional setting.

Setting 5.1. In this chapter, we again assume Setting 4.1, Setting 4.2 and Setting 4.3 from the previous chapter.

5.1 Algorithm

We start with a proposition, which shows, that the mathematical learning problem described in Theorem 4.4 is compatible with the hypothesis class of neural networks from a theoretical point of view.

Proposition 5.1 (Boundedness of the Kolmogorov PDE Learning Problem with Neural Networks). *Let $\mathcal{N}_{\rho,l,a,R}(D_\Pi, \mathbb{R})$ be a hypothesis class of neural networks over D_Π (see Definition 3.6). Then there exists a constant $M > 0$, such that, for all $\Phi_{\rho,l,a}(\theta_R, \cdot) \in \mathcal{N}_{\rho,l,a,R}(D_\Pi, \mathbb{R}^n)$, it holds that*

$$|\Phi_{\rho,l,a}(\theta_R, \Pi) - Y_{\text{target}}| \leq M. \quad (339)$$

Proof. By the triangle inequality and the boundedness of φ , we obtain

$$|\Phi_{\rho,l,a}(\theta_R, \Pi) - Y_{\text{target}}| \leq |\Phi_{\rho,l,a}(\theta_R, \Pi)| + |Y_{\text{target}}| \leq |\Phi_{\rho,l,a}(\theta_R, \Pi)| + C_\varphi \quad (340)$$

For $|\Phi_{\rho,l,a}(\theta_R, \Pi)|$, on the other hand, we can further estimate

$$|\Phi_{\rho,l,a}(\theta_R, \Pi)| \leq \|\Phi_{\rho,l,a}(\theta_R, \cdot)\|_\infty. \quad (341)$$

Since Proposition 3.1 assures, that $\mathcal{N}_{\rho,l,a,R}(D_\Pi, \mathbb{R})$ is a compact subspace of $(\mathcal{C}(D_\Pi, \mathbb{R}), \|\cdot\|_\infty)$ and since the maximum norm $\|\cdot\|_\infty : \mathcal{C}(D_\Pi, \mathbb{R}) \rightarrow \mathbb{R}_{\geq 0}$ is a continuous map, we can conclude, that $\|\cdot\|_\infty$ is bounded on $\mathcal{N}_{\rho,l,a,R}(D_\Pi, \mathbb{R})$ by a constant $\tilde{M} \geq 0$. This leads to the estimate

$$\|\Phi_{\rho,l,a}(\theta_R, \cdot)\|_\infty \leq \tilde{M}, \quad (342)$$

which completes the proof. $\square$

Remark 5.1 (Consistency of the Kolmogorov PDE Learning Problem with Neural Networks). Proposition 5.1, guarantees the consistency of the Kolmogorov PDE learning problem with neural networks, as described in Theorem 2.3. Furthermore, note that the target variable Y_{target} from Theorem 4.4 and the set D_Π fulfill the standard regularity requirements for statistical learning outlined in Remark 2.8.

Inspired by Theorem 4.4, we now propose the following algorithm for the numerical solution of a class of Kolmogorov partial differential equations with variable, affine-linear coefficients by means of deep learning, which we shall then test empirically.

Remark 5.2 (Solving a Class of Kolmogorov PDEs with Deep Artificial Neural Networks). The following strategy combines the theory and formal framework of Chapters 2, 3 and 4 (and in particular Theorem 4.4) in order to propose an algorithm for the numerical solution of Kolmogorov PDEs of the form

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0, \quad (343)$$

$$u(0, x) = \varphi(x), \quad (344)$$

with variable, affine-linear coefficient functions σ and μ and a continuous, bounded initial function φ using artificial neural networks as a hypothesis class.

1. **Setting the Hyperparameters $\mathbf{d}, \mathbf{T}, \mathbf{v}, \mathbf{w}, \mathbf{D}_{\Pi_\sigma}, \mathbf{D}_{\Pi_\mu}$ of the Kolmogorov PDE.** In order to determine the class of equations one wants to solve across a particular space-time region, one first chooses concrete values for the spatial dimension $d \in \mathbb{N}_{\geq 1}$, for the maximum time $T > 0$, for the boundary values $w > v > 0$ of the spatial hypercube $[v, w]^d$, and for the compact sets $D_{\Pi_\sigma} \subseteq (\mathbb{R}^{d \times d})^{d+1}$ and $D_{\Pi_\mu} \subseteq \mathbb{R}^{d \times d} \times \mathbb{R}^d$.
2. **Fixing the Distribution of the Predictor Variable Π .** Next, one needs to define distributions for the (pairwise independent) components

$$X_0 : \Omega \rightarrow [v, w]^d, \quad (345)$$

$$\Pi_\sigma : \Omega \rightarrow D_{\Pi_\sigma}, \quad (346)$$

$$\Pi_\mu : \Omega \rightarrow D_{\Pi_\mu}, \quad (347)$$

$$\mathcal{T} : \Omega \rightarrow [0, T] \quad (348)$$

of the predictor variable $\Pi = (X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T})$.

A natural choice, which we exclusively use in our approach, and which ensures an even learning process of the algorithm across the chosen space-time region and set of possible coefficient functions, is the uniform distribution. We choose 6 compact, nonempty intervals: the two intervals

$$I_{\mathcal{T}} := [0, T] \quad (349)$$

and

$$I_{X_0} := [v, w], \quad (350)$$

as well as four new compact, nonempty intervals

$$I_{\Pi_\sigma}, I_{\Pi_{\sigma, \text{add}}}, I_{\Pi_\mu}, I_{\Pi_{\mu, \text{add}}} \subseteq \mathbb{R}. \quad (351)$$

Now, $\mathcal{T}$ can be uniformly drawn from $I_{\mathcal{T}}$, the entries of X_0 can be independently and uniformly drawn from I_{X_0} , the entries of the matrices $\Pi_{\sigma, 1}, \dots, \Pi_{\sigma, d}$ can be drawn independently and uniformly from I_{Π_σ} , the entries of the matrix $\Pi_{\sigma, d+1}$ can be drawn independently and uniformly from

$I_{\Pi_\sigma, \text{add}}$, the entries of the matrix $\Pi_{\mu,1}$ can be independently and uniformly drawn from I_{Π_μ} , and the entries of the vector $\Pi_{\mu,2}$ can be independently and uniformly drawn from $I_{\Pi_\sigma, \text{add}}$.

Note, that by choosing the 4 intervals I_{Π_σ} , $I_{\Pi_\sigma, \text{add}}$, I_{Π_μ} , and $I_{\Pi_\mu, \text{add}}$, one also implicitly chooses the compact sets $D_{\Pi_\sigma} \subseteq (\mathbb{R}^{d \times d})^{d+1}$ and $D_{\Pi_\mu} \subseteq \mathbb{R}^{d \times d} \times \mathbb{R}^d$.

3. **Simulation of Training Data and Setting Euler-Maruyama Step-size η** Now, that all the hyperparameters are set and the distribution of the target variable Π is fixed, one needs to simulate training data points (π, y_{target}) consisting of realisations of the pair (Π, Y_{target}) . This can be done stepwise: We first draw a sample

$$\pi = (x_0, \pi_\sigma, \pi_\mu, t) \quad (352)$$

of the uniformly distributed random variable

$$\Pi = (X_0, \Pi_\sigma, \Pi_\mu, \mathcal{T}). \quad (353)$$

We then estimate a realisation $s_{(x_0, \pi_\sigma, \pi_\mu, t)}$ of the associated random variable $\tilde{S}(\cdot, (x_0, \pi_\sigma, \pi_\mu, t))$ (see Definition 4.12) by numerically solving the stochastic differential equation

$$S_s = x_0 + \int_0^s \mu(S_r) dr + \int_0^s \sigma(S_r) dB_r, \quad s \in [0, T], \quad (354)$$

at time t , where σ and μ are affine-linear functions determined by the parameters π_σ and π_μ . The classical method to simulate solutions of Equation 354 in a stochastic manner to obtain $s_{(x_0, \pi_\sigma, \pi_\mu, t)}$, is provided by the Euler-Maruyama method [56] with stepsize $\eta > 0$, which represents a straightforward probabilistic generalisation of the Euler method for the solution of ordinary differential equations and which we too employ in our numerical experiments. Finally, one can set $y_{\text{target}} := \varphi(s_{(x_0, \pi_\sigma, \pi_\mu, t)})$. This process can be iterated to produce an arbitrarily large number of training data points $((\pi_i, y_{\text{target}, i}))_{i=1}^\beta$. In order to improve training performance, it is advised to preprocess each training batch by applying a (componentwise) z-score transformation to the input variables $\pi_1, \dots, \pi_\beta$ via

$$\pi_{i,j} \mapsto \frac{\pi_{i,j} - \mathbb{E}[\Pi_j]}{\sqrt{\text{Var}(\Pi_j)}}. \quad (355)$$

4. **Setting the Hyperparameters ρ, l, a, R of the Neural Network Hypothesis Class and Adding Refinements.** The next step consists of choosing a concrete hypothesis space of neural networks to solve the statistical learning problem given by the simulated data points $((\pi_i, y_{\text{target}, i}))_{i=1}^\beta$. While the input dimension

$$d_{\text{input}} := d^3 + 2d^2 + 2d + 1 \quad (356)$$

and output dimension ($= 1$) of the neural network are determined by the shape of our training data, we are free in choosing an arbitrary number of hidden layers l and hidden neurons to fix our neural network architecture

$$a = (d_{\text{input}}, a_1, \dots, a_l, 1). \quad (357)$$

Moreover, we are required to choose a continuous, nonpolynomial activation function $\rho \in \mathcal{C}(\mathbb{R}, \mathbb{R})$.

We are then left with a fixed class of neural networks $\mathcal{N}_{\rho, l, a, R}(D_{\Pi}, \mathbb{R})$.³⁰ In practise, it is not necessary to actually choose a bound $R > 0$ for the size of the neural network parameters θ , since θ is always automatically limited by the specific constraints of the machine one is operating on. If necessary, it is possible to improve the performance of the vanilla neural network architecture by adding refinements 3.3 such as batch normalisation layers and dropout.

5. **Choosing a Training Algorithm and Setting the Training Hyper-parameters γ, β .** It is now possible to train the neural network using batches of simulated data $((\pi_i, y_{\text{target}, i}))_{i=1}^{\beta}$ and gradient-based methods via backpropagation. Advanced techniques such as Adam optimization and gradient clipping 3.3 are available in order to speed up convergence and improve training performance. Usually, gradient based methods, such as Adam optimization, come with a learning rate γ , which is multiplied with the computed gradient to specify the size of the parameter update at each training iteration. Adding special parameter initialization schemes such as Xavier initialization 3.3 can bring additional benefits.

Mathematically speaking, *training* an artificial neural network

$$\Phi_{\rho, l, a}(\theta_R, \cdot) \in \mathcal{N}_{\rho, l, a, R}(D_{\Pi}, \mathbb{R}) \quad (358)$$

corresponds to taking a trajectory of functions

$$\Phi_{\rho, l, a}(\theta_{R, 0}, \cdot), \Phi_{\rho, l, a}(\theta_{R, 1}, \cdot), \Phi_{\rho, l, a}(\theta_{R, 2}, \cdot), \dots \quad (359)$$

through the hypothesis space $\mathcal{N}_{\rho, l, a, R}(D_{\Pi}, \mathbb{R})$ in an attempt to choose θ_R such as to minimize the empirical least squares error

$$\frac{1}{b} \sum_{i=1}^b |\Phi_{\rho, l, a}(\theta_R, \pi_i) - y_{\text{target}, i}|^2. \quad (360)$$

For gradient-based minimization methods, each iterative training step is based on the gradient of Expression 360 with respect to θ_R , with a batch of simulated training points $((\pi_i, y_{\text{target}, i}))_{i=1}^{\tilde{m}}$ of size β . New batches can

³⁰Note, that we tacitly identify the set D_{Π} with a compact subset of $\mathbb{R}^{d_{\text{input}}}$ via some arbitrary homeomorphism to make our construction of a neural network hypothesis class consistent with Definition 3.6. Wherever necessary for formal reasons, we continue to use this identification without further comment.

be simulated on demand at each iteration; when training is stopped after j training steps, one has reached 1 training epoch with a sample size of

$$m := j\beta \quad (361)$$

training points.

6. **Training the Neural Network.** Finally, one can train the neural network $\Phi_{\rho,l,a}(\theta_R, \pi_i)$ with the chosen technique until the loss 360 has decreased to a level where satisfactory accuracy is reached. The time required for this process in particular depends on the dimension d of the equation, the dimension $s(a)$ of the neural network parameter vector θ_R , as well as the desired level of accuracy.

Below, we present two successful numerical experiments, associated with the application of our technique from Remark 5.2 to two concrete classes of Kolmogorov PDEs. Our experiments were run on an

- HP EliteBook 850 G3,
- equipped with Windows 10 Pro (64 Bit),
- an Intel(R) Core(TM) i7-6500U CPU @ 2.50 GHz 2.59 GHz
- and 8192MB RAM.

The source code for our method was written in `Python` using the well-established `Tensorflow` deep learning package and can be found in Appendix A 7.

In our experiments, algorithm 5.2 is applied to solve two types of Kolmogorov equations and the results are compared with independent estimates of the actual solutions of these equations. As an objective error measure for our numerical solutions compared to the actual solutions of the Kolmogorov PDEs in question, we employ a Monte Carlo estimate of the mean squared difference (MSD) between functions.

Definition 5.1 (Mean Squared Difference Between Functions). *Let $\tilde{d} \in \mathbb{N}_{\geq 1}$ be a natural number and $\tilde{D} \subseteq \mathbb{R}^{\tilde{d}}$ be a compact, nonempty set. Furthermore, let*

$$\tilde{u}_1, \tilde{u}_2 : \tilde{D} \rightarrow \mathbb{R} \quad (362)$$

be two $\mathcal{B}(\tilde{D})/\mathcal{B}(\mathbb{R})$ -measurable functions and let

$$X : \Omega \rightarrow \tilde{D} \quad (363)$$

be a uniformly distributed $\mathcal{G}/\mathcal{B}(\tilde{D})$ -measurable random variable. Then the mean squared difference (MSD) between $\tilde{u}_1$ and $\tilde{u}_2$ is defined as

$$MSD(\tilde{u}_1, \tilde{u}_2) := \mathbb{E}[(\tilde{u}_1(X) - \tilde{u}_2(X))^2] = \int_{\tilde{D}} (\tilde{u}_1(x) - \tilde{u}_2(x))^2 \frac{d\lambda(x)}{\lambda(\tilde{D})} \in [0, \infty], \quad (364)$$

where $\lambda : \mathcal{B}(\tilde{D}) \rightarrow [0, \infty]$ is the $\tilde{d}$ -dimensional Lebesgue measure.

Remark 5.3 (Estimating MSD via Monte Carlo Simulation). Assume the setting of Definition 5.1 and let $X_0, X_1, \dots$ be i.i.d. copies of X . Then the law of large numbers implies, that the sequence

$$\left(\sum_{j=0}^{\xi} \frac{(\tilde{u}_1(X_j) - \tilde{u}_2(X_j))^2}{\xi} \right)_{\xi \in \mathbb{N}_{\geq 0}} \quad (365)$$

converges $\mathbb{P}$ -almost surely towards $\text{MSD}(\tilde{u}_1, \tilde{u}_2)$, which can be exploited to numerically estimate $\text{MSD}(\tilde{u}_1, \tilde{u}_2)$ via Monte Carlo simulation.

After training a neural network $\Phi_{\rho, l, a}$ using the algorithm from Remark 5.2, we can simulate an estimation of the type 365 with sample size $\xi \in \mathbb{N}_{\geq 1}$ to approximate the MSD between

$$D_{\Pi} \ni (x_0, \pi_{\sigma}, \pi_{\mu}, t) \mapsto \Phi_{\rho, l, a}(\theta_{R, \text{trained}}, (x_0, \pi_{\sigma}, \pi_{\mu}, t)) \in \mathbb{R} \quad (366)$$

and an independent and valid estimate of the actual solution function

$$D_{\Pi} \ni (x_0, \pi_{\sigma}, \pi_{\mu}, t) \mapsto u^{(\pi_{\sigma}, \pi_{\mu})}(t, x_0) \in \mathbb{R}. \quad (367)$$

of the class of equations

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0, \quad (368)$$

$$u(0, x) = \varphi(x) \quad (369)$$

with variable, affine-linear coefficients σ and μ . Independent estimates of the exact solution map 367 can either be obtained directly via numerical integration (Example 5.2) or via Monte carlo simulation exploiting the Feynman-Kac formula (Example 5.3).

Remark 5.4 (Relative Mean Squared Difference). Assume the setting of Definition 5.1. To obtain a relative (and therefore asymmetric) error measure, one can define the mean squared relative difference (MSRD) via

$$\text{MSRD}(\tilde{u}_1, \tilde{u}_2) := \mathbb{E} \left[\left(\frac{\tilde{u}_1(X) - \tilde{u}_2(X)}{\tilde{u}_1(x)} \right)^2 \right]. \quad (370)$$

The problem with this definition is, that the denominator $\tilde{u}_1(x)$ might vanish, possibly making the expression $\text{MSRD}(\tilde{u}_1, \tilde{u}_2)$ ill-defined. We circumvent this, by instead defining the modified mean squared relative difference (MMSRD_{ϵ})

$$\text{MMSRD}_{\epsilon}(\tilde{u}_1, \tilde{u}_2) := \mathbb{E} \left[\left(\frac{\tilde{u}_1(X) - \tilde{u}_2(X)}{\epsilon + \tilde{u}_1(x)} \right)^2 \right]. \quad (371)$$

for some fixed, small $\epsilon > 0$. The MMSRD_{ϵ} can be numerically estimated in a similar way as the MSD by sampling a number of $\xi \in \mathbb{N}_{\geq 0}$ values and is applied in Example 5.3 in addition to our standard MSD error measure. ³¹

³¹Note, that, strictly speaking, MMSRD_{ϵ} is not a relative error measure, since it is not invariant under rescaling of the functions $(\tilde{u}_1, \tilde{u}_2)$. However, it is still a useful and "sufficiently relative" measure for our empirical purposes.

5.2 The 1-Dimensional Heat Equation

In our first numerical experiment, we apply our algorithm to a simple low-dimensional class of Kolmogorov equations, namely 1-dimensional heat equations with varying heat coefficients $\alpha > 0$:

$$\frac{\partial u}{\partial t}(t, x) = \alpha \frac{\partial^2 u}{\partial x^2}(t, x). \quad (372)$$

As our fixed initial condition, we choose the smooth, bounded, compactly supported bump-function

$$\varphi : \mathbb{R} \rightarrow \mathbb{R}, \quad \varphi(x) = \mathbb{1}_{[-3,3]}(x) e^{\frac{-1}{1-(\frac{x}{3})^2}}. \quad (373)$$

We are interested in training a single neural network $\Phi_{\rho,l,a}$ to describe the family of unique, at most polynomially growing viscosity solutions $(u^{(\alpha)})_{\alpha \in [0,5]}$ of the Equations 372 on the space-time region

$$[0, T] \times [v, w] := [0, 1] \times [-5, 5] \quad (374)$$

for variable coefficient parameters $\alpha \in [0, 5]$.

It follows from well-known results [57], that, for fixed $\alpha > 0$, it is possible to express each solution function $u^{(\alpha)}$ of 372 with initial function φ on the space-time region $(0, 1] \times [-5, 5]$ in the following integral form :

$$u^{(\alpha)}(t, x) = \int_{-\infty}^{\infty} \frac{e^{\frac{-(x-s)^2}{4\alpha t}}}{\sqrt{4\pi\alpha t}} \varphi(s) ds. \quad (375)$$

We will exploit this integral identity in order to approximate the actual solution function $u^{(\cdot)}(\cdot, \cdot)$ of Equation 372 (with variable coefficient parameter $\alpha \in [0, 5]$ and initial condition φ) via numerical integration. This can then in turn be used to estimate the error quantity $\text{MSD}(\Phi_{\rho,l,a}(\theta_{R,\text{trained}}, \cdot), u^{(\cdot)}(\cdot, \cdot))$.

Setting 5.2. To stay in accordance with our algorithmic framework outlined in Remark 5.2, we now choose the following setting to formally capture the above problem.

General Hyperparameters

- Let $d = 1$.
- Let $T = 1$.
- Let $[v, w] = [-5, 5]$
- Let $\Pi : \Omega \rightarrow D_{\Pi}$ be uniformly distributed, as described in Remark 5.2.
- Let $I_{\mathcal{T}} = [0, T] = [0, 1]$.
- Let $I_{X_0} = [v, w] = [-5, 5]$.

- Let $I_{\Pi_\sigma} = [0, 0] = \{0\}$.
- Let $I_{\Pi_\sigma, \text{add}} = [0, \sqrt{10}]$.
- Let $I_{\Pi_\mu} = [0, 0] = \{0\}$.
- Let $I_{\Pi_\mu, \text{add}} = [0, 0] = \{0\}$.
- Let D_{Π_σ} be implicitly defined by I_{Π_σ} and $I_{\Pi_\sigma, \text{add}}$. (see Remark 5.2).
- Let D_{Π_μ} be implicitly defined by I_{Π_μ} and $I_{\Pi_\mu, \text{add}}$. (see Remark 5.2).
- Let $\varepsilon = 1$.
- Let the sample size for the Monte Carlo estimate of the MSD between our trained neural network function $\Phi_{\rho, l, a}(\theta_{R, \text{trained}}, \cdot)$ and our integral approximation of the actual solution function $u^{(\cdot)}(\cdot, \cdot)$ be given by $\xi = 16$.

Neural Network Architecture Hyperparameters

- Let the neural network activation function be $\text{relu}(x) = \max(\{0, x\})$.
- Let the number of hidden layers of our neural network be $l = 3$.
- Let the architecture of our neural network be $a = (6, 10, 10, 10, 1)$ ³².
- Let the bound R of the neural network parameter size be implicitly given by the constraints of our physical machine.
- Let our neural network be refined by 4 batch normalisation layers $b_1, \dots, b_4$, resulting in the final architecture $a_{\text{batchnorm}} = (1, b_1, 10, b_2, 10, b_3, 10, b_4, 1)$.

Neural Network Training Hyperparameters

- Let our training technique be Adam optimization 3.3.
- Let our training stepsize (= learning rate) be $\gamma = 10^{-3}$.
- Let the training batch size be $\beta = 256$.
- Let the stepsize of the Euler-Maruyama method for the simulation of our training data be $\eta = 0.01$.
- Let the biases of all neurons be initialized to 0.
- Let the weights of our neural network be initialized using the Xavier initializer 3.3.

³²Note, that, since $I_{\Pi_\sigma} = I_{\Pi_\mu} = I_{\Pi_\mu, \text{add}} = \{0\}$ we could reduce the input dimension of the network from 6 to 3 to obtain a slight increase in performance; for reasons of formal consistency with our algorithm from Remark 5.2, however, we stick with $d_{\text{input}} = 6$ and, in addition to the relevant variables (t, x, a) , canonically feed 3 constant values of magnitude 0 (corresponding to the constant parts of μ and σ) into our network.

- Let our number of training iterations be $it = 12000$.

Table 1 depicts the mean and standard deviation (StD) of some metrics associated with the first 2000 iterations of the training phase of our neural network, in steps of 100 iterations, computed on the basis of 6 independent trials. The table depicts the amount of time needed (in seconds [s]), the decrease of the empirical error 360 as well as the decrease of the MSD between the neural network prediction and the solution of the Kolmogorov PDE as computed via numerical integration of 375. Furthermore, the decrease of both error measures (empirical error and MSD) as training progresses is depicted in Figure 3. As expected, both the empirical error and the MSD continue to decrease, suggesting convergence of our method towards the real solution of the 1-dimensional heat equation.

An even stronger indication for the empirical success of our approach can be seen in Figure 4, which depicts our neural network predictions after 1 run of Algorithm 5.2 with 12000 training iterations vs. the actual solution function of the 1-dimensional heat equation, calculated via numerical integration of 375, for a selection of fixed pairs (t, α) of time-points t and heat coefficients α .

It	Time [s] (Mean)	Time [s] (StD)	Emp. Error (Mean)	Emp. Error (StD)	MSD (Mean)	MSD (StD)
0	0.0	0.0	1.132	0.4601	0.1533	0.2256
100	27.7837	2.5955	0.0723	0.0373	0.1231	0.12
200	55.3337	4.5527	0.0343	0.0176	0.0841	0.0805
300	81.7915	5.4699	0.0252	0.0082	0.0781	0.0779
400	108.1559	5.9809	0.0207	0.0056	0.0716	0.0445
500	134.7322	6.735	0.0198	0.0036	0.0556	0.0326
600	161.2288	7.0743	0.0173	0.0026	0.0443	0.0292
700	187.5791	7.6556	0.0146	0.0019	0.0427	0.0364
800	214.2606	8.3034	0.0139	0.0032	0.0389	0.026
900	240.9325	8.9616	0.0132	0.0015	0.032	0.0222
1000	267.8036	9.887	0.0139	0.0017	0.04	0.0403
1100	294.2786	10.4828	0.0138	0.0021	0.0224	0.0082
1200	321.9409	11.0399	0.0129	0.0017	0.019	0.0061
1300	348.4994	11.6988	0.0124	0.002	0.0211	0.0044
1400	374.9533	12.3565	0.0119	0.0009	0.0246	0.0158
1500	401.6037	13.3066	0.0126	0.0016	0.0156	0.0058
1600	429.2638	15.2681	0.0116	0.0015	0.0211	0.0075
1700	457.0472	18.0291	0.0123	0.001	0.0156	0.0141
1800	485.2202	21.0554	0.0112	0.0014	0.016	0.0056
1900	511.7238	21.9724	0.0111	0.0016	0.0104	0.0043
2000	538.6509	23.0408	0.0114	0.0017	0.0082	0.003

Table 1: Decrease of Empirical Error and MSD, as well as training time for the first 2000 iterations (It) when applying Algorithm 5.2 to the 1-dimensional heat equation with variable heat coefficient (average over 6 independent trials).

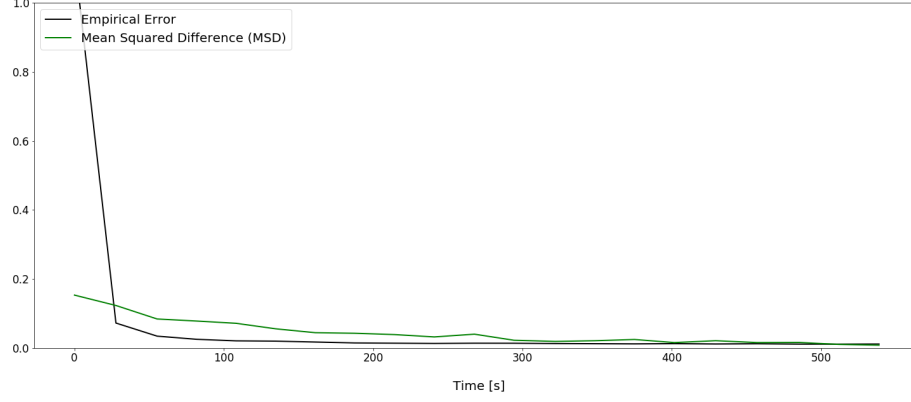


Figure 3: Decrease of Empirical Error and Mean Squared Difference (MSD) in the first 540 seconds of training our neural network to learn the solution of the 1-dimensional heat equations with variable heat coefficient.

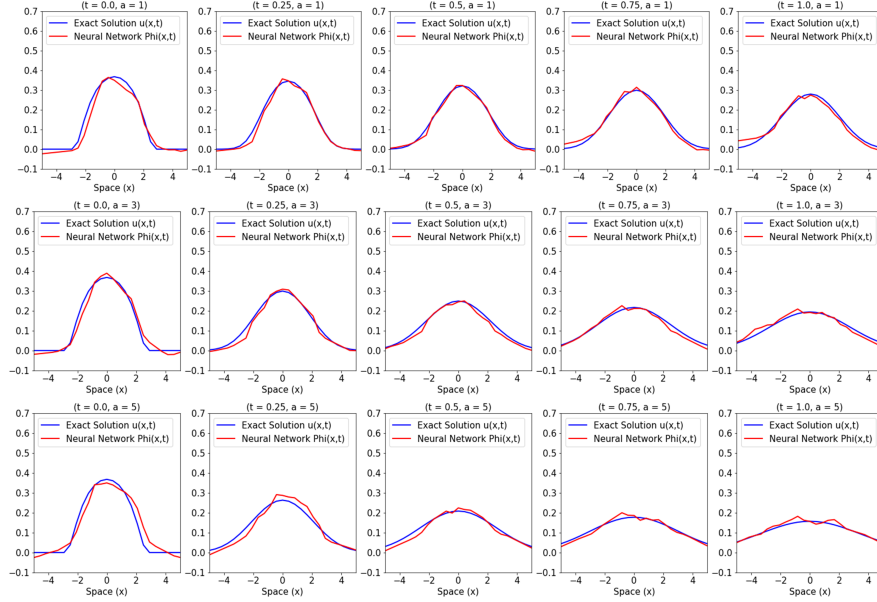


Figure 4: Neural network predictions after 12000 iterations vs. actual solutions of the 1-dimensional heat equation with variable heat coefficient, calculated via numerical integration of 375, depicted for a selection of fixed pairs (t, α) of time-points t and heat coefficients α .

5.3 European Put Option Pricing

In our second experiment, we apply Algorithm 5.2 to numerically solve a general d -dimensional class of Kolmogorov PDE problems, given by

$$\frac{\partial u}{\partial t}(t, x) - \frac{1}{2} \text{Trace}[\sigma(x)[\sigma(x)]^* (\text{Hess}_x u)(x, t)] - \langle \mu(x), (\nabla_x u)(t, x) \rangle = 0, \quad (376)$$

$$u(0, x) = \varphi(x) \quad (377)$$

with the continuous, bounded initial function

$$\varphi : \mathbb{R}^d \rightarrow \mathbb{R}, \quad \varphi(x) = \min(\{\max(\{1 - \sum_{i=1}^d \frac{x_i}{d}, 0\}, 1)\}, 1), \quad (378)$$

and affine-linear coefficients σ, μ , whose defining scalar matrix/vector entries are bounded by some predefined constants.

Remark 5.5 (European Put Option Pricing via Kolmogorov PDEs). Problems of the above type are often encountered in computational finance and can be used to model the evolution of the fair price $u(x, t)$ of an European put option as a function of the time t until the expiration date and the underlying stock prices $x = (x_1, \dots, x_d)$ under idealized conditions. In our case, the strike price of the option is set to 1. Note, that the change of the option price is modelled "backwards in time", i.e. $u(0, x) = \varphi(x)$ corresponds to the fair price of the option at the time of expiration and $u(T, x)$ corresponds to the fair price of the option at the time of aquisition.

Below we present function plots of our results associated with the above equation in the case $d = 1$ as well as detailed numerical tables and error plots for the high-dimensional case $d = 10$. The error measures MSD and MMSRD $_{\epsilon}$ were computed by comparing the estimates produced by our trained neural network $\Phi_{\rho, l, a}(\theta_{R, \text{trained}}, \cdot)$ with an independent Monte Carlo estimate of the actual solution $u^{(\cdot, \cdot)}(\cdot, \cdot)$ with sample size $\tilde{\xi}$, based on the Feynman-Kac 286 representation of the solution

$$u^{(\pi_{\sigma}, \pi_{\mu})}(t, x_0) = \mathbb{E}[\varphi(S_t^{(x_0, \pi_{\sigma}, \pi_{\mu})})]. \quad (379)$$

Setting 5.3. To again formally express the structure of the above problem with our algorithmic framework from Remark 5.2, we choose the following hyperparameters.

General Hyperparameters

- Let $d = 1$ (in the first case) and $d = 10$ (in the second case).
- Let $T = 2$.
- Let $[v, w] = [0, 2]$

- Let $\Pi : \Omega \rightarrow D_\Pi$ be uniformly distributed, as described in Remark 5.2.
- Let $I_{\mathcal{T}} = [0, T] = [0, 2]$.
- Let $I_{X_0} = [v, w] = [0, 2]$.
- Let $I_{\Pi_\sigma} = [-1, 1]$.
- Let $I_{\Pi_\sigma, \text{add}} = [-1, 1]$.
- Let $I_{\Pi_\mu} = [-1, 1]$.
- Let $I_{\Pi_\mu, \text{add}} = [-1, 1]$.
- Let D_{Π_σ} be implicitly defined by I_{Π_σ} and $I_{\Pi_\sigma, \text{add}}$. (see Remark 5.2).
- Let D_{Π_μ} be implicitly defined by I_{Π_μ} and $I_{\Pi_\mu, \text{add}}$. (see Remark 5.2).
- Let the sample size of our Monte Carlo approximation of the actual solution function $u^{(\cdot, \cdot)}(\cdot, \cdot)$ be $\tilde{\xi} = 32$
- Let the sample size for the Monte Carlo estimate of the MSD and MMSRD $_\epsilon$ between our trained neural network function $\Phi_{\rho, l, a}(\theta_{R, \text{trained}}, \cdot)$ and our Monte Carlo approximation of the actual solution function $u^{(\cdot, \cdot)}(\cdot, \cdot)$ be given by $\xi = 32$.
- Let $\epsilon = 1$.

Neural Network Architecture Hyperparameters

- Let the neural network activation function be $\text{relu}(x) = \max(\{0, x\})$.
- Let the number of hidden layers of our neural network be $l = 10$.
- Let $d_{\text{input}} = d^3 + 2d^2 + 2d + 1$.
- Let $h = 5d_{\text{input}}$.
- Let the architecture of our neural network be $a = (d_{\text{input}}, h, h, h, h, h, h, h, h, h, h, 1)$.
- Let the bound R of the neural network parameter size be implicitly given by the constraints of our physical machine.
- Let our neural network be refined by 11 batch normalisation layers $b_1, \dots, b_{11}$, resulting in the final architecture $a_{\text{batchnorm}} = (d_{\text{input}}, b_1, h, \dots, b_{10}, h, b_{11}, 1)$.

Neural Network Training Hyperparameters

- Let our training technique be Adam optimization 3.3.
- Let our training stepsize (= learning rate) be $\gamma = 10^{-3}$.

- Let the training batch size be $\beta = 256$.
- Let the stepsize of the Euler-Maruyama method for the simulation of our training data be $\eta = 0.01$.
- Let the biases of all neurons be initialized to 0.
- Let the weights of our neural network be initialized using the Xavier initializer 3.3.
- Let our number of training iterations be $it = 12000$ in the case of $d = 1$ and $it = 1000$ in the case of $d = 10$.

Figure 5 depicts the predictions of our neural network in the case $d = 1$ for selected time points t and coefficients π_σ, π_μ after 12000 iterations, compared with the Monte Carlo estimate of the actual solution. The function plots indicate successful convergence of our method. The time needed for the first 1000 training iterations was 1883 Seconds; it took 22595 Seconds to complete all 12000 iterations. The approximate amount of time needed to reach a MSD below 0.1 was 150 seconds (380 training iterations). Predictions seemed to be slightly better for small t ; a possible explanation for this is the degradation in the accuracy of our simulated training data for big t , resulting from our use of the Euler-Maruyama method with stepsize $\eta = 0.01$. One could therefore try to counteract this phenomenon by decreasing η , at the cost of a prolonged computation time.

Table 2 shows numerical results associated with average error measures in the case $d = 10$, computed on the basis of 5 independent trials with 1000 training iterations each. As expected, Algorithm 5.2 lead to a decrease in empirical loss, MSE and MMSRD_ϵ , which once again displayed the empirical success of our method. For $d = 10$, the approximate average amount of time needed to reach an average MSD below 0.1 was 10748 seconds (780 training iteration), which is approximately 71.65 times the amount of time which was needed for the same accuracy in the case $d = 1$. Figure 6 and Figure 7 depict the decrease of the empirical error, the MSD and the MMSRD_ϵ as training time progresses in the high-dimensional case $d = 10$. Interestingly, one can see, that the MSE and MMSRD_ϵ kept on decreasing even after the empirical error had reached a plateau, implying a further improvement in the accuracy of our model without a change in empirical error.

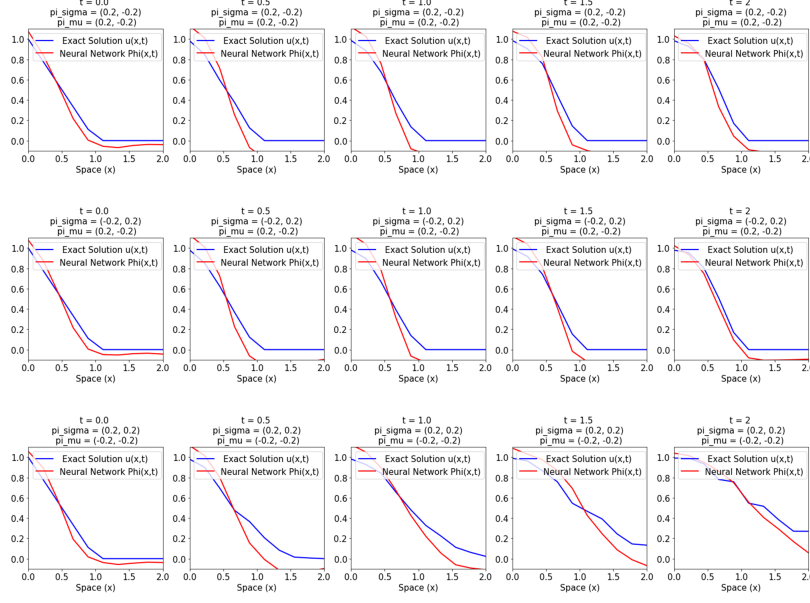


Figure 5: Neural network predictions after 12000 iterations vs. actual solution of the European put option pricing problem with variable initial data, computed via Monte Carlo simulation, for the case $d = 1$ and a selection of fixed input parameters (π_σ, π_μ, t) and variable $x \in [0, 2]$.

It	Time [s] (Mean)	Time [s] (Std)	Emp. Error (Mean)	Emp. Error (Std)	MSD (Mean)	MSD (Std)	MMSRD $_{\epsilon}$ (Mean)	MMSRD $_{\epsilon}$ (Std)
0	0.0	0.0	2.479	0.134	0.252	0.026	0.265	0.0234
20	261.173	8.021	7.689	1.412	42708.438	52106.02	1.079	0.0917
40	526.445	16.697	0.775	0.343	103150.414	29070.973	1.076	0.1004
60	787.928	15.239	0.367	0.043	27609.775	10721.548	0.987	0.0223
80	1048.976	15.685	0.287	0.016	4415.855	2318.712	1.0	0.0525
100	1308.018	14.383	0.263	0.008	1308.071	248.249	1.002	0.0852
120	1567.002	12.584	0.26	0.009	430.014	99.365	16.761	30.6497
140	1824.965	13.093	0.254	0.009	156.524	102.695	1.628	0.8215
160	2081.369	13.004	0.255	0.004	82.817	27.086	3.781	2.8917
180	2339.498	15.069	0.262	0.007	72.45	39.686	368.843	732.0574
200	2596.954	16.173	0.254	0.007	29.444	10.844	65.896	75.9266
220	2856.667	17.144	0.25	0.01	13.892	6.792	90499.54	180896.8392
240	3113.97	16.714	0.253	0.009	9.68	3.974	271.923	235.2003
260	3371.343	17.849	0.248	0.004	4.385	2.336	21051.158	41787.2392
280	3628.507	17.695	0.253	0.006	4.916	2.961	20.061	13.0598
300	3894.96	11.627	0.246	0.005	3.361	1.204	20.019	22.3244
320	4178.118	46.726	0.251	0.003	3.069	1.53	102.955	169.4633
340	4455.808	79.243	0.249	0.01	1.408	0.657	37.288	66.2163
360	4727.634	98.694	0.251	0.004	1.733	1.629	671.86	1303.9598
380	4993.754	113.367	0.248	0.005	0.838	0.426	5.062	8.5974
400	5263.806	134.655	0.249	0.007	0.856	0.435	37.706	70.9991
420	5532.537	134.829	0.25	0.005	0.381	0.161	1.394	2.0752
440	5818.144	138.586	0.247	0.004	0.247	0.074	0.852	0.9443
460	6092.337	153.108	0.249	0.007	0.427	0.209	4.29	5.2642
480	6362.731	163.296	0.256	0.011	0.298	0.192	29.907	41.5759
500	6628.711	170.812	0.249	0.004	0.219	0.125	0.229	0.2916
520	6909.138	200.705	0.253	0.003	0.128	0.062	0.183	0.1857
540	7187.639	224.9	0.246	0.004	0.203	0.125	0.192	0.1247
560	7460.808	245.621	0.255	0.005	0.169	0.139	0.182	0.1532
580	7724.42	246.887	0.251	0.008	0.187	0.081	0.624	0.9109
600	7987.942	247.315	0.247	0.004	0.132	0.075	0.106	0.0835
620	8258.55	265.827	0.253	0.007	0.124	0.041	0.137	0.1814
640	8527.784	283.224	0.253	0.005	0.156	0.129	0.194	0.1739
660	8786.346	283.837	0.248	0.006	0.119	0.104	0.068	0.0614
680	9046.179	284.965	0.253	0.005	0.104	0.042	4.506	8.5961
700	9342.761	348.316	0.254	0.006	0.167	0.128	0.609	1.1076
720	9669.405	467.253	0.251	0.003	0.062	0.03	0.049	0.0405
740	10081.17	727.327	0.252	0.005	0.064	0.056	0.048	0.0534
760	10439.786	924.423	0.256	0.012	0.154	0.205	2.677	5.277
780	10748.059	1008.532	0.254	0.012	0.103	0.104	0.11	0.1763
800	11068.894	1087.953	0.257	0.007	0.071	0.025	0.042	0.0289
820	11409.436	1242.814	0.252	0.008	0.059	0.024	0.026	0.0156
840	11717.364	1287.558	0.25	0.009	0.041	0.025	0.041	0.0465
860	12035.498	1356.541	0.252	0.006	0.05	0.038	0.029	0.0242
880	12350.229	1400.561	0.253	0.007	0.044	0.007	0.034	0.0156
900	12665.498	1452.725	0.251	0.004	0.045	0.024	0.021	0.0139
920	12965.548	1477.467	0.252	0.007	0.052	0.034	0.026	0.0171
940	13239.445	1502.047	0.255	0.005	0.048	0.036	0.032	0.0318
960	13516.633	1535.906	0.251	0.011	0.062	0.038	0.056	0.0445
980	13781.103	1546.602	0.253	0.007	0.037	0.015	0.025	0.0121
1000	14076.828	1619.109	0.252	0.007	0.051	0.025	0.025	0.021

Table 2: Decrease of Empirical Error, MSD, MMSRD $_{\epsilon}$, as well as training time for 1000 training iterations (It) when applying Algorithm 5.2 to the 10-dimensional European put option pricing problem with variable coefficient parameters (average over 5 independent trials)

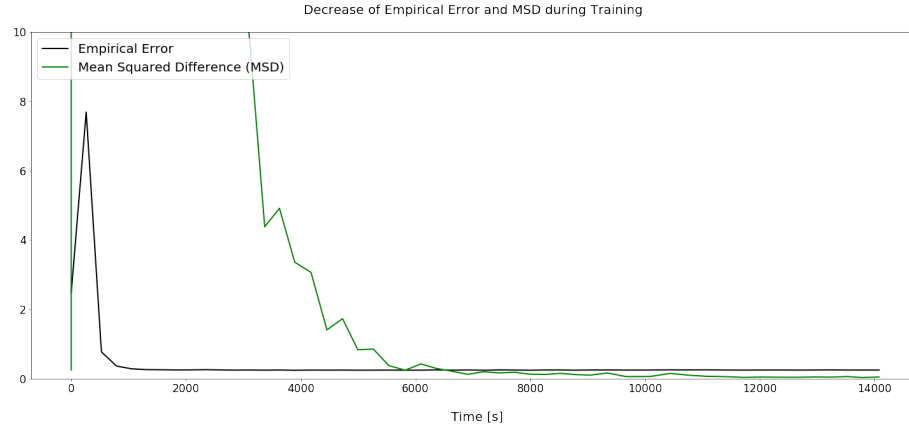


Figure 6: Decrease of Empirical Error and Mean Squared Difference (MSD) while training our neural network to learn the solution of the 10-dimensional European put option pricing problem with variable coefficient parameters.

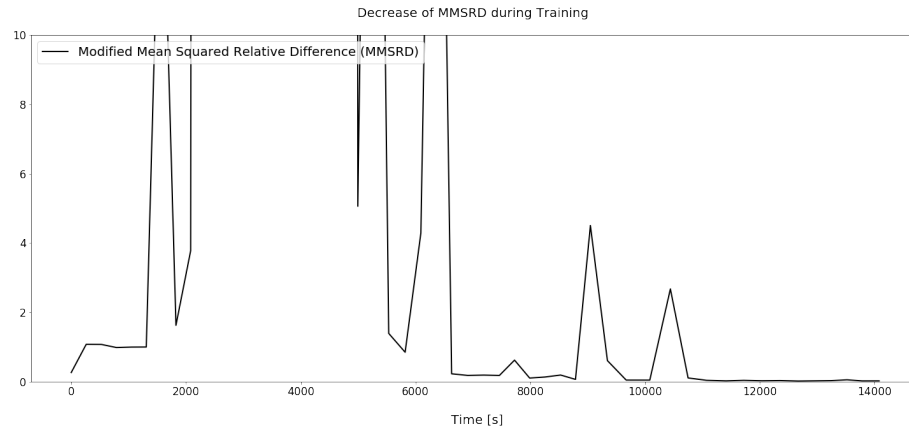


Figure 7: Decrease of Modified Mean Squared Relative Difference (MMSRD_ε) while training our training our neural network to learn the solution of 10-dimensional European put option pricing problem with variable coefficient parameters.

6 Conclusion

In this thesis, we proposed a formal theorem 4.4 and an associated algorithm 5.2, which open up the possibility to numerically solve classes of Kolmogorov partial differential equations with variable, affine linear coefficients using techniques from statistical machine learning. We came close to delivering a technical proof of Theorem 4.4; however, a full justification for the validity of the assumptions (†) made in the beginning of the proof of 4.4 is still outstanding and could form the basis of a future theoretical research project.

From an empirical point of view, it was possible to successfully demonstrate the applicability of Algorithm 5.2 for the numerical solution of the 1-dimensional heat equation 5.2 and a European put option pricing model in both 1 and 10 dimensions 5.3. In particular, our numerical experiments indicate the convergence of Algorithm 5.2 in both a low- and high-dimensional setting against the correct solutions of the underlying classes of Kolmogorov PDEs. The numerical results in this thesis can therefore be seen as very encouraging, even though, rigorously speaking, more research is necessary to theoretically guarantee convergence of our method in all relevant cases. Methods to further boost the computational performance of our technique (both in terms of speed and accuracy), include decreasing the Euler-Maruyama stepsize η to improve the quality of training data simulations for big t , and employing an exhaustive hyperparameter grid search in order to find ideal neural network architectures and training settings.

Future empirical research could focus on studying the growth of computational complexity of our algorithm as the dimension of the Kolmogorov PDE d increases; it seems possible, that the amount of time needed for Algorithm 5.2 to reach a certain desired accuracy δ with high probability only scales polynomially in the parameters d and δ , thus overcoming the curse of dimensionality, as was previously shown for the case of fixed coefficients σ and μ in the work of Berner, Grohs, Jentzen et. al. [10] [11] [9].

Finally, one could further generalize our approach by not only viewing x, σ, μ and t as variable training data, but also the initial condition φ of our Kolmogorov PDE. By choosing a parametrizable family of functions $(\varphi_\pi)_{\pi \in D_\varphi}$ and feeding the functions φ_π into our neural network, one might be able to numerically solve a class of Kolmogorov PDEs with variable initial function parameters π as well as variable coefficient parameters π_σ, π_μ in a single training session.

7 Appendix A

7.1 Source Codes

In this section, we provide the source code for Example 5.2 and Example 5.3, written in the programming language **Python**, using the standard numerical library **Numpy**, along with the well-established deep learning package **Tensorflow**. The concrete parameter settings in the code below correspond to the 10-dimensional case of Example 5.3; minor, straightforward changes in the parameter settings can be made to reproduce Example 5.2 and the 1-dimensional case of Example 5.3.

```
1 # Solving Kolmogorov Partial Differential Equations
2 # with Variable Initial Data by Means of Deep Learning
3
4
5 # import packages
6
7 import numpy as np # numpy numerical matrix package
8 import math # basic mathematics package
9 import tensorflow as tf # tensorflow deep learning package
10 import sdeint # stochastic differential equation solution package
11 from timeit import default_timer as timer # import timer
12
13 # define hyperparameters
14
15
16
17     # general hyperparameters
18
19 d = 1 # dimension of Kolmogorov equation
20
21
22
23     # PDE and training data simulation hyperparameters
24
25 I_x = (0,2) # range of each 1-d random entry of X_0
26 I_sigma = (-1,1) # range of the 1-d random entries of Pi_Sigma[0], ...,
27     Pi_Sigma[d-1]
28 I_sigma_add = (-1, 1) # range of the 1-d random entries of Pi_Sigma[d]
29 I_mu = (-1,1) # range of the 1-d random entries of Pi_mu[0]
30 I_mu_add = (-1,1) # range of the 1-d random entries of Pi_mu[1]
31 I_t = (0,2) # range of the random time  $\tau$ 
32 stepsize_sol_sde = 1e-2 # stepsize for the solution of the SDE
33
34
35     # neural network hyperparameters
36
37 d_input = d + 1 + d*d + d + (d+1)*(d*d) # size of input layer of neural
38     network
39 h = 5*d_input # number of neurons per hidden layer
40 arch_list = [d_input, h, h, h, h, h, h, h, h, h, h, 1] # neural network
41     architecture
42 act_func = tf.nn.relu # neural network activation function
43 batch_norm = True # batch normalisation yes/no
44 dropout = False # apply dropout (true/false)
45 dkr = 0.9 # probability for each neuron to stay activated during dropout
46     (if dropout True)
47 learning_rate = 1e-3 # learning rate for neural network gradient-based
48     optimization
```

```

45 training_method = tf.train.AdamOptimizer(learning_rate) # neural network
    optimization method
46 batchsize = 256 # batchsize during training
47 b_initializer = tf.constant_initializer(value = 0) # neural network bias
    initializer
48 w_initializer = tf.contrib.layers.xavier_initializer() # neural network
    weight initializer
49 loss_function = tf.losses.mean_squared_error # loss function to be
    minimized
50
51
52
53 # error estimation hyperparameters
54
55 kol_pde_sol_mc_sample_size = 32
56 mse_sample_size = 32 # number of samples used to estimate the MSD
57 mmsrd_sample_size = 32 # number of samples used to estimate the MMSRD
58 varepsilon = 1.0 # additive epsilon to avoid division by 0 in the def.
    of the MMSRE
59
60
61
62
63 # define coefficient functions for Kolmogorov PDE
64
65 def sigma(x, pi_sigma):
66     assert np.shape(x)[0] == np.shape(pi_sigma[0])[0]
67
68     matrixlist = []
69
70     for A in pi_sigma[0:-1]:
71         matrixlist.append(list(np.matmul(A,x)))
72
73     result = np.transpose(np.array(matrixlist)) + pi_sigma[-1]
74
75     return np.float32(result)
76
77
78 def mu(x, pi_mu):
79     assert np.shape(x)[0] == np.shape(pi_mu[0])[0]
80     assert np.shape(pi_mu[0])[0] == np.shape(pi_mu[1])[0]
81
82     A = pi_mu[0]
83     b = pi_mu[1]
84
85
86     return np.float32(np.matmul(A,x) + b)
87
88
89
90 # define initial function varphi for Kolmogorov PDEs
91
92
93 # initial function for European put option pricing model
94
95 def varphi_option(x):
96
97     dim_x = len(list(x))
98
99     mathcal_D = 1
100     sum = np.sum(x)/dim_x
101
102     result = min(max(mathcal_D-sum, 0),mathcal_D)
103
104     return result
105
106
107 # initial function for 1-d heat equation

```

```

108
109 def varphi_heat(x):
110     if x > -3 and x < 3:
111         return np.float32(math.exp(-1/(1-(x/3)**2)))
112     else:
113         return np.float32(0)
114
115
116 # define solution functions associated with 1-d heat equation
117
118
119     # fundamental solution of 1-d heat equation
120
121 def fundamental_heat_eq(t, x, a = 1):
122     nenner = math.sqrt(4 * math.pi * a * t)
123     zaehler = math.exp(-x*x/(4*a*t))
124     return zaehler/nenner
125
126
127     # solution of the 1-d heat equation, computed via numerical
128     integration
129
130 def sol_heat_eq(x_0, pi_sigma, pi_mu, t, phi_init = varphi_heat):
131     x_0 = x_0[0]
132     t = t[0]
133
134     if t == 0:
135         result = phi_init(x_0)
136
137     if t > 0:
138         a = ((pi_sigma[1][0][0])**2)/2
139         ff = lambda y: fundamental_heat_eq(t, x_0-y, a=a)*phi_init(y)
140         result = scipy.integrate.quad(ff, -np.inf, np.inf, epsabs = 0)[0]
141     return result
142
143
144
145
146 # define monte carlo / feynman-kac simulation functions
147
148
149     # random variable, which appears inside the expectation symbol
150     # in the Feynman-Kac formula
151     # this requires the stepwise solution of a
152     # stochastic differential equation using the Euler-Maruyama scheme
153
154 def feynman_kac_mc(x_0, pi_sigma, pi_mu, t, phi_init, stepsize_sol_sde =
155     1e-2):
156
157     def s(x, tt):
158         return sigma(x, pi_sigma)
159
160     def m(x, tt):
161         return mu(x, pi_mu)
162
163     points = int(np.round(t/stepsize_sol_sde) + 2)
164     tspan = np.linspace(0, t[0], points)
165
166     result = sdeint.itoEuler(m, s, x_0, tspan)[-1]
167     result = phi_init(result)
168
169     return result
170
171
172
173     # pointwise Monte-Carlo estimate of the

```

```

174     # solution of the Kolmogorov PDE, based on Feynman-Kac formula
175
176 def kol_pde_sol_mc(x_0,
177     pi_sigma, pi_mu, t,
178     phi_init = varphi_option,
179     stepsize_sol_sde = 1e-2,
180     sample_size = kol_pde_sol_mc_sample_size):
181
182     sum = 0
183     for k in range(sample_size):
184         sum += feynman_kac_mc(x_0,
185             pi_sigma, pi_mu, t,
186             phi_init,
187             stepsize_sol_sde = stepsize_sol_sde)
188
189     return sum/sample_size
190
191
192
193
194 # define functions for the simulation of training data
195
196
197 def X_0(I_x, dim):
198     return np.float32(np.random.uniform(I_x[0], I_x[1], dim))
199
200 def Pi_sigma(I_sigma, I_sigma_add, dim):
201
202     matrixlist = []
203
204     for k in range(dim):
205         matrixlist.append(np.float32(np.random.uniform(I_sigma[0],
206             I_sigma[1], (dim, dim))))
207
208     matrixlist.append(np.float32(np.random.uniform(I_sigma_add[0],
209         I_sigma_add[1], (dim, dim))))
210
211     return tuple(matrixlist)
212
213 def Pi_mu(I_mu, I_mu_add, dim):
214     matrixvectorpair = []
215     matrixvectorpair.append(np.float32(np.random.uniform(I_mu[0], I_mu
216         [1], (dim, dim))))
217     matrixvectorpair.append(np.float32(np.random.uniform(I_mu_add[0],
218         I_mu_add[1], (dim,))))
219
220     return tuple(matrixvectorpair)
221
222 def Tau(I_t):
223     return np.float32(np.random.uniform(I_t[0], I_t[1], (1,)))
224
225 def std_unif(l):
226     '''standard deviation of uniform distribution on interval of length
227     l'''
228
229     return math.sqrt((l*1)/12)
230
231 def standardize_unif(A, I):
232     ''' standardize (z-score) numpy array with
233     random entries drawn uniformly from an interval I = (a,b)'''
234
235     A = A-(I[0]/2 + I[1]/2)
236     if I[1]-I[0] == 0:
237         return A
238     else:
239         return A/std_unif(I[1]-I[0])

```

```

237
238 def vectorize_data(x_0,
239                   pi_sigma,
240                   pi_mu,
241                   t,
242                   I_x,
243                   I_sigma,
244                   I_sigma_add,
245                   I_mu,
246                   I_mu_add,
247                   I_t,
248                   standardize = False):
249     '''Vectorize and standardize training data'''
250
251
252     if standardize == False:
253
254         pi_sigma = np.ravel(np.array(pi_sigma))
255         pi_mu_0 = np.ravel(np.array(pi_mu[0]))
256         pi_mu_1 = np.ravel(np.array(pi_mu[1]))
257         result = np.concatenate((x_0, pi_sigma, pi_mu_0, pi_mu_1, t))
258
259
260     if standardize == True:
261
262         x_0 = np.array(x_0)
263         pi_sigma = np.array(pi_sigma)
264         pi_mu = list(pi_mu)
265         t = np.array(t)
266
267         x_0 = standardize_unif(x_0, I_x)
268         pi_sigma[0:-1] = standardize_unif(np.array(pi_sigma[0:-1]),
269         I_sigma)
270         pi_sigma[-1] = standardize_unif(np.array(pi_sigma[-1]),
271         I_sigma_add)
272         pi_mu[0] = standardize_unif(pi_mu[0], I_mu)
273         pi_mu[1] = standardize_unif(pi_mu[1], I_mu_add)
274         t = standardize_unif(t, I_t)
275
276         pi_sigma = np.ravel(np.array(pi_sigma))
277         pi_mu_0 = np.ravel(np.array(pi_mu[0]))
278         pi_mu_1 = np.ravel(np.array(pi_mu[1]))
279         result = np.concatenate((x_0, pi_sigma, pi_mu_0, pi_mu_1, t))
280
281     return np.float32(result)
282
283
284
285 def simulate_training_data(I_x,
286                           I_sigma,
287                           I_sigma_add,
288                           I_mu,
289                           I_mu_add,
290                           I_t,
291                           dim,
292                           phi_init,
293                           batchsize,
294                           stepsize_sol_sde = 1e-3):
295
296     '''Returns simulated training data, where the input variabes
297     are given both raw and as z-scored variables'''
298
299     X_norm = []
300     X = []
301     Y = []
302

```

```

303
304     for k in range(batchsize):
305
306         x_0 = X_0(I_x = I_x, dim = dim)
307         pi_sigma = Pi_sigma(I_sigma = I_sigma, I_sigma_add = I_sigma_add
, dim = dim)
308         pi_mu = Pi_mu(I_mu = I_mu, I_mu_add = I_mu_add, dim = dim)
309         t = Tau(I_t = I_t)
310
311
312         X.append(vectorize_data(x_0,
313                                pi_sigma,
314                                pi_mu,
315                                t,
316                                I_x = I_x,
317                                I_sigma= I_sigma,
318                                I_sigma_add= I_sigma_add,
319                                I_mu= I_mu,
320                                I_mu_add= I_mu_add,
321                                I_t= I_t,
322                                standardize = False))
323
324         X_norm.append(vectorize_data(x_0,
325                                    pi_sigma,
326                                    pi_mu,
327                                    t,
328                                    I_x = I_x,
329                                    I_sigma= I_sigma,
330                                    I_sigma_add= I_sigma_add,
331                                    I_mu= I_mu,
332                                    I_mu_add= I_mu_add,
333                                    I_t= I_t,
334                                    standardize = True))
335
336
337
338         Y.append([feynman-kac-mc(x_0,
339                                pi_sigma,
340                                pi_mu,
341                                t,
342                                phi_init = phi_init,
343                                stepsize_sol_sde = stepsize_sol_sde)])
344
345     X_norm = np.float32(X_norm)
346     X = np.float32(X)
347     Y = np.float32(Y)
348
349     return [X_norm, X, Y]
350
351
352
353
354     # define computational graph instantiation function
355     # which constructs and returns a neural network using tensorflow
356
357
358     def neural_network_constructor(arch_list = [1,10,10,1],
359                                    act_func = tf.nn.relu,
360                                    w_initializer = tf.contrib.layers.
361
362                                    xavier_initializer(),
363                                    b_initializer = tf.zeros_initializer(),
364                                    loss_function = tf.losses.
365
366                                    mean_squared_error,
367                                    training_method = tf.train.AdamOptimizer
368
369                                    (0.5),
370                                    batch_norm = True,
371                                    dropout = False):

```

```

367     '''returns output layer of the network Phi, input variable
368     placeholder X,
369     training data label placeholder Y, the training node train_step,
370     whose execution corresponds to one training step,
371     and two placeholders bn_training and dropout_keep_rate,
372     which control the activation of batch normalisation and the dropout
373     rate of neurons'''
374
375     # define input and output dimension of the network
376     n_input = arch_list[0]
377     n_output = arch_list[-1]
378
379     # define placeholders
380     X = tf.placeholder(dtype = tf.float32, shape = [None, n_input])
381     Y = tf.placeholder(dtype = tf.float32, shape = [None, n_output])
382     bn_training = tf.placeholder(dtype = tf.bool)
383     dropout_keep_rate = tf.placeholder(dtype = tf.float32)
384
385     # define input layer
386     layer = tf.identity(X)
387
388     # define first fully connected layer
389     layer = tf.contrib.layers.fully_connected(
390         inputs = layer,
391         num_outputs = arch_list[1],
392         activation_fn = act_func,
393         weights_initializer = w_initializer,
394         biases_initializer = b_initializer)
395
396     # define first dropout layer
397     if dropout == True:
398         layer = tf.nn.dropout(layer, keep_prob = dropout_keep_rate)
399
400     # define first batch normalisation layer
401     if batch_norm == True:
402         layer = tf.contrib.layers.batch_norm(
403             inputs = layer,
404             is_training=bn_training)
405
406     for N in arch_list[2:-1]:
407
408         # define consecutive fully connected layers
409         layer = tf.contrib.layers.fully_connected(
410             inputs = layer,
411             num_outputs = N,
412             activation_fn = act_func,
413             weights_initializer = w_initializer,
414             biases_initializer = b_initializer)
415
416         # define consecutive dropout layers
417         if dropout == True:
418             layer = tf.nn.dropout(layer, keep_prob = dropout_keep_rate)
419
420         # define consecutive batch normalisation layers
421         if batch_norm == True:
422             layer = tf.contrib.layers.batch_norm(
423                 inputs = layer,
424                 is_training=bn_training)
425
426     # define output layer of the neural network
427     Phi = tf.contrib.layers.fully_connected(
428         inputs = layer,
429         num_outputs = n_output,
430         activation_fn = tf.identity,
431         weights_initializer = w_initializer,
432         biases_initializer = b_initializer)

```

```

433
434
435     # ops. to update mean/variance of batchnorm during training
436     update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
437
438     # loss function and regular training step
439     loss = loss_function(Y, Phi)
440
441     # define training operation
442
443     if batch_norm == True:
444         train_step = [update_ops, training_method.minimize(loss), loss]
445     else:
446         train_step = [training_method.minimize(loss), loss]
447
448
449     return [Phi, X, Y, train_step, bn_training, dropout_keep_rate]
450
451
452
453
454     # instantiation of the computational graph forming the neural network,
455     # using predefined hyperparameters
456
457     tf.reset_default_graph()
458     [Phi, X, Y, train_step, bn_training, dropout_keep_rate_ph] =
459         neural_network_constructor(
460             arch_list = arch_list,
461             act_func = act_func,
462             w_initializer = w_initializer,
463             b_initializer = b_initializer,
464             loss_function = loss_function,
465             training_method = training_method,
466             batch_norm = batch_norm,
467             dropout = dropout)
468
469     # start session and initialize variables
470
471     sess = tf.Session()
472     sess.run(tf.global_variables_initializer())
473
474
475     # train the neural network
476
477     iterations = 10
478     runtime_start = timer()
479
480     for it in range(0, iterations):
481
482
483         [X_train_norm, X_train, Y_train] = simulate_training_data(I_x = I_x,
484             I_sigma = I_sigma,
485             I_sigma_add = I_sigma_add,
486             I_mu = I_mu,
487             I_mu_add = I_mu_add,
488             I_t = I_t,
489             dim = d,
490             phi_init = varphi_option,
491             batchsize = batchsize,
492             stepsize_sol_sde = stepsize_sol_sde)
493
494     loss = sess.run(train_step,
495         feed_dict={X: X_train_norm, Y: Y_train, bn_training: True,
496             dropout_keep_rate_ph : dkr})
497
498

```

```

499     if it%1 == 0:
500         runtime = timer()-runtime_start
501         print("Iterations = ", it ,
502               ", Time = ", runtime ,
503               ", Emp. Error = ", loss[-1])
504
505
506
507 # Wrap a function around our (now trained) neural network
508
509 def neural_net(x_0 ,
510               pi_sigma ,
511               pi_mu ,
512               t ,
513               I_x = I_x ,
514               I_sigma = I_sigma ,
515               I_sigma_add = I_sigma_add ,
516               I_mu = I_mu ,
517               I_mu_add = I_mu_add ,
518               I_t = I_t):
519
520     inp = np.float32([vectorize_data(x_0 ,
521                                     pi_sigma ,
522                                     pi_mu ,
523                                     t ,
524                                     I_x ,
525                                     I_sigma ,
526                                     I_sigma_add ,
527                                     I_mu ,
528                                     I_mu_add ,
529                                     I_t ,
530                                     standardize = True)])
531
532     result = sess.run(Phi, feed_dict = {X : inp, bn_training: False ,
533                                         dropout_keep_rate_ph : 1})[0][0]
534
535     return result

```

References

- [1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [2] Geoffrey E Hinton, Terrence Joseph Sejnowski, and Tomaso A Poggio. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.
- [3] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.
- [4] Vladimir Vapnik. *Statistical learning theory. 1998*, volume 3. Wiley, New York, 1998.
- [5] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.
- [6] Tomaso Poggio and Steve Smale. The mathematics of learning: Dealing with data. *Notices of the AMS*, 50(5):537–544, 2003.
- [7] O Knill. Probability theory and stochastic processes with applications overseas press. *New Delhi*, 2009.
- [8] Rick Durrett. *Probability: theory and examples*. Cambridge university press, 2010.
- [9] Julius Berner, Philipp Grohs, and Arnulf 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. *arXiv preprint arXiv:1809.03062*, 2018.
- [10] Philipp Grohs, Fabian Hornung, Arnulf Jentzen, and Philippe Von Wurstemberger. A proof that artificial neural networks overcome the curse of dimensionality in the numerical approximation of black-scholes partial differential equations. *arXiv preprint arXiv:1809.02362*, 2018.
- [11] Christian Beck, Sebastian Becker, Philipp Grohs, Nor Jaafari, and Arnulf Jentzen. Solving stochastic differential equations and kolmogorov equations by means of deep learning. *arXiv preprint arXiv:1806.00421*, 2018.
- [12] Charalambos D Aliprantis and Kim C Border. Infinite dimensional analysis: a hitchhikers guide, 2006.
- [13] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [14] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

- [15] Gülin Dede and Murat Hüsnü Sazlı. Speech recognition with artificial neural networks. *Digital Signal Processing*, 20(3):763–768, 2010.
- [16] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.
- [17] Dan Cireşan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*, 2012.
- [18] Dan Claudiu Cireşan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [19] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.
- [20] Adam Roberts, Jesse Engel, and Douglas Eck. Hierarchical variational autoencoders for music. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2017.
- [21] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.
- [22] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [24] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [25] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.
- [26] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- [27] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- [28] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 5(4):455–455, 1992.

- [29] Walter Rudin. Principles of mathematical analysis, mcccrawhill companies. *Inc. New York*, 1976.
- [30] E Corominas and F Sunyer Balaguer. Conditions for an infinitely differentiable function to be a polynomial, *rev. mat. hisp. Amer.*(4), 14:26–43, 1954.
- [31] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pages 6231–6239, 2017.
- [32] Philipp Petersen, Mones Raslan, and Felix Voigtlaender. Topological properties of the set of functions generated by neural networks of fixed size. *arXiv preprint arXiv:1806.08459*, 2018.
- [33] Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In *Conference on Learning Theory*, pages 1246–1257, 2016.
- [34] Andrzej P Ruszczyński and Andrzej Ruszczyński. *Nonlinear optimization*, volume 13. Princeton university press, 2006.
- [35] AN Iusem. On the convergence properties of the projected gradient method for convex optimization. *Computational & Applied Mathematics*, 22(1):37–52, 2003.
- [36] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [37] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
- [38] Frank H Clarke. *Optimization and nonsmooth analysis*, volume 5. Siam, 1990.
- [39] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [40] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [42] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*, 2018.

- [43] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.
- [44] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [45] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.
- [46] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 497–504. ACM, 2017.
- [47] Venu G Gudise and Ganesh K Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*, pages 110–117. IEEE, 2003.
- [48] Jing-Ru Zhang, Jun Zhang, Tat-Ming Lok, and Michael R Lyu. A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied mathematics and computation*, 185(2):1026–1037, 2007.
- [49] David J Montana and Lawrence Davis. Training feedforward neural networks using genetic algorithms. In *IJCAI*, volume 89, pages 762–767, 1989.
- [50] HK Lam, SH Ling, Frank HF Leung, and Peter Kwong-Shun Tam. Tuning of the structure and parameters of neural network using an improved genetic algorithm. In *IECON'01. 27th Annual Conference of the IEEE Industrial Electronics Society (Cat. No. 37243)*, volume 1, pages 25–30. IEEE, 2001.
- [51] Jatinder ND Gupta and Randall S Sexton. Comparing backpropagation with a genetic algorithm for neural network training. *Omega*, 27(6):679–684, 1999.
- [52] Michael G Crandall, Hitoshi Ishii, and Pierre-Louis Lions. Users guide to viscosity solutions of second order partial differential equations. *Bulletin of the American mathematical society*, 27(1):1–67, 1992.
- [53] Martin Hairer, Martin Hutzenthaler, Arnulf Jentzen, et al. Loss of regularity for kolmogorov equations. *The Annals of Probability*, 43(2):468–527, 2015.

- [54] Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.
- [55] Jean-François Le Gall. *Brownian motion, martingales, and stochastic calculus*, volume 274. Springer, 2016.
- [56] Peter E Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.
- [57] John Rozier Cannon. *The one-dimensional heat equation*, volume 23. Cambridge University Press, 1984.

List of Figures

1	Artificial neural network with $l = 2$ hidden layers and architecture $a = (4, 5, 7, 3)$ depicted as directed acyclic graph. Source: https://goo.gl/images/qQ1B41 (20.02.2019).	25
2	Neural network with $l = 1$ hidden layer and an architecture of $a = (2, 2, 2)$	32
3	Decrease of Empirical Error and Mean Squared Difference (MSD) in the first 540 seconds of training our neural network to learn the solution of the 1-dimensional heat equations with variable heat coefficient.	77
4	Neural network predictions after 12000 iterations vs. actual solutions of the 1-dimensional heat equation with variable heat coefficient, calculated via numerical integration of 375, depicted for a selection of fixed pairs (t, α) of time-points t and heat coefficients α	77
5	Neural network predictions after 12000 iterations vs. actual solution of the European put option pricing problem with variable initial data, computed via Monte Carlo simulation, for the case $d = 1$ and a selection of fixed input parameters (π_σ, π_μ, t) and variable $x \in [0, 2]$	81
6	Decrease of Empirical Error and Mean Squared Difference (MSD) while training our neural network to learn the solution of the 10-dimensional European put option pricing problem with variable coefficient parameters.	83
7	Decrease of Modified Mean Squared Relative Difference (MMSRD $_\epsilon$) while training our training our neural network to learn the solution of 10-dimensional European put option pricing problem with variable coefficient parameters.	83