A Markovian Model for Learning-to-Optimize

Michael Sucker MICHAEL.SUCKER@MATH.UNI-TUEBINGEN.DE
Department of Mathematics
University of Tübingen
Tübingen, Germany

Peter Ochs OCHS@MATH.UNI-SAARLAND.DE
Department of Mathematics and Computer Science
Saarland University
Saarbrücken, Germany

Abstract

We present a probabilistic model for stochastic iterative algorithms with the use case of optimization algorithms in mind. Based on this model, we present PAC-Bayesian generalization bounds for functions that are defined on the trajectory of the learned algorithm, for example, the expected (non-asymptotic) convergence rate and the expected time to reach the stopping criterion. Thus, not only does this model allow for learning stochastic algorithms based on their empirical performance, it also yields results about their actual convergence rate and their actual convergence time. We stress that, since the model is valid in a more general setting than learning-to-optimize, it is of interest for other fields of application, too. Finally, we conduct five practically relevant experiments, showing the validity of our claims.

Keywords: learning-to-optimize, stochastic processes, pac-bayes, convergence rate, stopping time

1 Introduction

Learning-to-optimize is an important topic of current research, because optimization problems are ubiquitous in science and industry, their solution is often time-consuming and costly, and learned optimization algorithms can outperform classical ones by orders of magnitude. However, more often than not, theoretical guarantees for such learned optimization algorithms are missing, which renders their application at least questionable. Therefore:

In this work, we consider parametric stochastic iterative (optimization) algorithms to minimize parametric (loss) functions, and how to learn such algorithms with theoretical guarantees on their non-asymptotic convergence rate and convergence time.

The purpose of this introduction is a) to motivate the upcoming discussion and b) to decipher the statement above. The starting point of our considerations is a *parameteric loss function* $\ell(s, \theta)$, which we want to minimize in s for every realization of θ. To do this, a *stochastic algorithm* $\mathcal{A}$ is applied *iteratively*, and yields a sequence $\xi = (\xi^{(t)})_{t\in\mathbb{N}_0}$:

$$\xi^{(t+1)} = \mathcal{A}(\alpha, \theta, \xi^{(t)}, \eta^{(t+1)})\,. \tag{1}$$

Here, the *hyperparameters* α allow for adjusting the algorithm, while the *parameters* θ specify the current loss function the algorithm is applied to, and $\eta^{(t+1)}$ models the *(internal)*

randomness in each iteration. Starting from some initialization $\xi^{(0)}$, the overall goal of such an algorithm is to find a state s that satisfies certain properties, typically specified in terms of $\ell(\cdot, \theta)$. If such a state is found, the algorithm is stopped and is considered to be *converged*. Then, the number of iterations it takes to converge is the *convergence time*. Since, generally, one wants to solve optimization problems in the least amount of time, the aim of this work is to learn $\mathcal{A}$ in such a way that we can *guarantee* to reach the stopping criterion in a certain amount of time, that is, such that we can upper bound the convergence time. Here, *learning* $\mathcal{A}$ refers to choosing the hyperparameters based on a data set of parameters. However, there is a catch: Both the convergence rate and stopping time are properties of the *trajectory* of the algorithm, that is, they cannot be pinpoint to single, fixed iterates. Therefore, for stating such guarantees, one has to have access to all the iterates $\xi = \left(\xi^{(t)}\right)_{t \in \mathbb{N}_0}$, which, for every choice of hyperparameters α and parameters θ, form a *discrete-time stochastic process*, which we have to analyze. While stochastic processes might be arbitrarily complex objects, the process induced by Equation (1), which is visualized in Figure 1, is actually driven by a single simple equation, which allows to disentangle the randomness of this particular process. Indeed, the "total randomness" is exactly a superposition of the four separate sources given by the initialization $\xi^{(0)}$, parameters θ, hyperparameters α, and internal randomness $\eta^{(t)}$. Each source has a different kind of influence on the trajectories generated by $\mathcal{A}$. Similarly, as will be shown in Section 4, the distribution of the whole stochastic process ξ emerges from a similar and equally simple equation. Even more so, it is in fact *uniquely defined* by this, that is, this is actually *the only way* to define this distribution which is compatible with the underlying stochasticity. Therefore, after the related work and preliminaries, in Section 4 we will derive the corresponding probabilistic model by building it from ground up, basically starting from Equation (1). Then, in Section 5, we use this model to derive, in a rather standard way, generalization bounds for learning such an algorithm based on data. Especially, our results include generalization bounds for its non-asymptotic convergence rate and its convergence time of the following, informal form (compare to Corollaries 35 and 38 together with Remark 43):

Theorem 1 (Informal) *Under mild boundedness assumptions on the algorithm, the ρ-average convergence time $\bar{\tau}$ and the ρ-average convergence rate $\bar{r}$ can be bounded, respectively, by the ρ-average empirical convergence time $\hat{\tau}$ plus some remainder term $R_{t,N}$, and the the ρ-average empirical convergence rate $\hat{r}$ plus some remainder term $R_{r,N}$, where both $R_{t,N}$ and $R_{r,N}$ vanish with the size N of the data set $\mathscr{P}_{[N]}$, that is, for all $\varepsilon > 0$ and $\lambda > 0$:*

$$\mathbb{P}_{\mathscr{P}_{[N]}} \left\{ \forall \rho \in \mathcal{P} \ : \ \rho[\bar{\tau}] \leq \rho[\hat{\tau}] + R_{t,N} \text{ and } \rho[\bar{r}] \leq \rho[\hat{r}] + R_{r,N} \right\} \geq 1 - \varepsilon \,.$$

Finally, in Section 6, we conduct several practically relevant experiments to underline the validity of our theoretical claims. Here, we design new algorithms, which we train and evaluate with the proposed approach to showcase their superior performance compared to a "standard" algorithm. To our knowledge, approaching learning-to-optimize from the perspective of stochastic processes, modelling the corresponding distribution of the trajectory, and using it for learning a *stochastic* optimization algorithm with guarantees on its convergence rate and convergence time has not been done before, and might pave the way to other approaches.

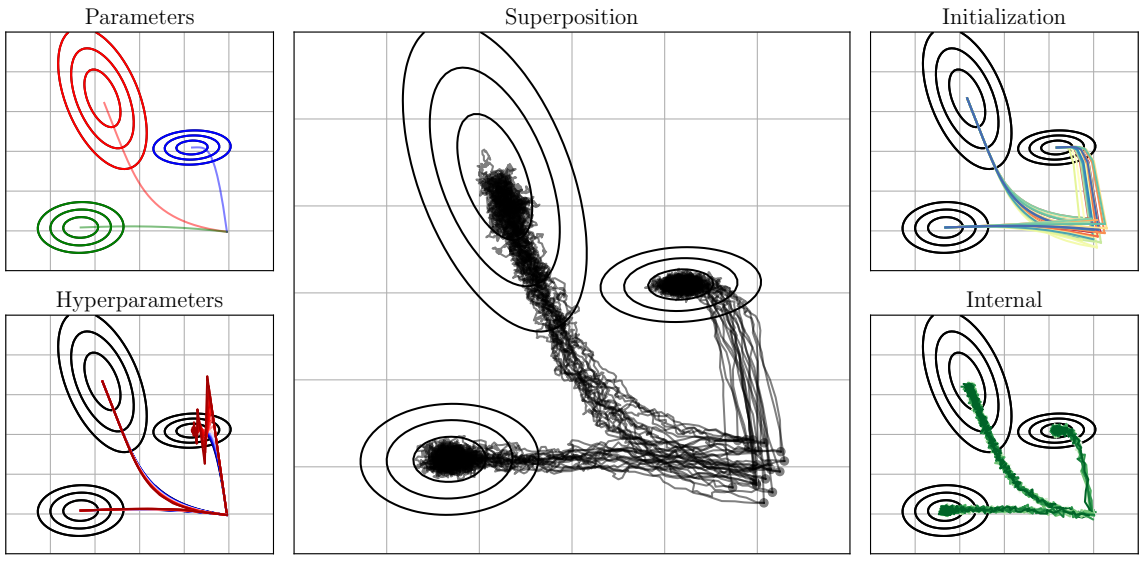

Figure 1: Superposition of different sources of randomness: The algorithm can be applied to several problem instances coming from a common distribution (upper left). Since this is not under the control of the user, we refer to it as *external randomness*. Further, the algorithm might be started from different, randomly chosen initializations (upper right). Furthermore, there might be randomness (or *uncertainty*) in the choice of the hyperparameters of the algorithm (lower left). Finally, the algorithmic update might be inherently stochastic (lower right), which, as it is inherent to the algorithm, we refer to as *internal randomness*. Combining these four sources of randomness yields the superposition depicted in the middle.

2 Related Work

The literature on both learning-to-optimize and the PAC-Bayesian learning approach is vast. Hence, the discussion of learning-to-optimize will mainly focus on learning approaches that provide some theoretical guarantees. Especially, this excludes many approaches that "only" evaluate their model empirically. Chen et al. (2021) provide a good overview about the variety of approaches in learning-to-optimize, and good introductory references for the PAC-Bayesian approach are given by Guedj (2019) and Alquier (2021).

2.0.1 Broader Context of Learning-to-Optimize

Solving optimization problems is an integral part of machine learning. Thus, learning-to-optimize has significant overlap with the areas of meta-learning (or "learning-to-learn") and AutoML. The first one is a subset of learning-to-optimize, because learning-to-optimize applies to general optimization problems while meta-learning is mostly concerned with determining parameters of machine learning models (Vilalta and Drissi, 2002; Hospedales et al., 2021). AutoML, however, more broadly refers to automating all steps necessary to create a machine learning application, which therefore also involves the choice of an

optimization algorithm and its hyperparameters (Yao et al., 2018; Hutter et al., 2019; He et al., 2021).

2.0.2 Learning-to-Optimize with Guarantees

Chen et al. (2021) point out that learned optimization methods may lack theoretical guarantees for the sake of convergence speed. That being said, there are applications where a convergence guarantee is of highest priority. To underline this, Moeller et al. (2019) provide an example where a purely learning-based approach fails to reconstruct the crucial details in a medical image. Also, they prove convergence of their method by restricting the output to descent directions, for which mathematical guarantees exist. The basic idea is to trace the learned object back to, or constrain it to, a mathematical object with convergence guarantees. Similarly, Sreehari et al. (2016) provide sufficient conditions under which the learned mapping is a proximal mapping. Related schemes, under different assumptions and guarantees, are given by Chan et al. (2016), Teodoro et al. (2017), Tirer and Giryes (2018), Buzzard et al. (2018), Ryu et al. (2019), Sun et al. (2019), Terris et al. (2021) and Cohen et al. (2021). A major advantage of these methods is the fact that the number of iterations is not restricted a priori. However, a major drawback is their restriction to specific algorithms and problems. Another approach, which limits the number of iterations, yet in principle can be applied to every iterative optimization algorithm, is unrolling, pioneered by Gregor and LeCun (2010). Xin et al. (2016) study the IHT algorithm and show that it is, under some assumptions, able to achieve a linear convergence rate. Likewise, Chen et al. (2018) establish a linear convergence rate for the unrolled ISTA. However, a difficulty in the theoretical analysis of unrolled algorithms is actually the notion of convergence itself, such that one rather has to consider the generalization performance. Only few works have addressed this: Either directly by means of Rademacher complexity (Chen et al., 2020), or indirectly in form of a stability analysis (Kobler et al., 2020), as algorithmic stability is linked to generalization (Bousquet and Elisseeff, 2000, 2002; Shalev-Shwartz et al., 2010).
Another line of work studies the design of learned optimization algorithms, pathologies and pitfalls during training, and how it affects the possible guarantees (Wichrowska et al., 2017; Metz et al., 2019, 2022). In this context, Liu et al. (2023) advocated for more mathematical structure in learning-to-optimize and proposed to enforce these convergence properties by design. Similarly, Castera and Ochs (2024) analyze hand-crafted optimization algorithms that can be applied in a wide range of problems, extract common geometric properties from them, and, based on that, provide design-principles for learned optimization algorithms.
A recent work somewhat related to ours is the the preprint of Xie et al. (2024), in which the authors also tackle the problem of learning optimization algorithms with convergence rates. In doing so, they also introduce a notion of "stopping time". However, while trying to provide an answer to the same question, the approaches taken are complementary: In their work, the authors combine the ODE-approach to optimization algorithms with learning-to-optimize to leverage the convergence of a continuous-time trajectory together with the stability of a (forward Euler) discretization scheme to deduce, *analytically*, the convergence of their algorithm. Through this, if the ODE is discretized in a stable way, they can provide *asymptotic, worst-case* convergence rates, which yield an upper bound on the performance of the algorithm. On the other hand, we approach the problem *statisti-*

cally, and provide generalization guarantees for the *non-asymptotic, average* convergence rate, that is, a guarantee that the rate which gets observed during training will generalize to new problems from the same distribution. By design, this closely resembles the average performance of the learned algorithm. Another important difference to our work is the fact that, since the discretization scheme is fixed, the choice of the ODE completely determines (up to hyperparameters) the resulting algorithm. In our approach, however, we investigate an abstract algorithm, that is, the design of the update-step is not fixed a-priori. *Thus, we use the iterative approach of learning the update step of an abstract algorithm. This has three advantages: a) We can use an arbitrary number of iterations, b) it does not limit the design of the algorithm, and c) it allows for deriving the distribution of the trajectory. Then, we provide generalization guarantees for (bounded) functions defined on the space of trajectories, that is, functions that resemble some statistic about the algorithm. Especially, this includes non-asymptotic convergence rates, finite stopping times, and function values at any finite iteration.*

2.0.3 PAC-Bayesian Bounds and Bounded Loss Functions

The PAC-Bayesian framework allows for giving high probability bounds on the risk, either as an oracle or as an empirical bound. The key ingredient is a change-of-measure inequality, which strongly influences the corresponding bound. The one used most often is based on a variational representation of the Kullback–Leibler divergence due to Donsker and Varadhan (1975), employed, for example, by Catoni (2004, 2007). Yet, not all bounds are based on a variational representation, that is, holding uniformly over all posterior distributions (Rivasplata et al., 2020). While most bounds involve the Kullback–Leibler divergence as measure of proximity (McAllester, 2003a,b; Seeger, 2002; Langford and Shawe-Taylor, 2002; Germain et al., 2009), more recently other divergences have been used (Honorio and Jaakkola, 2014; London, 2017; Bégin et al., 2016; Alquier and Guedj, 2018; Ohnishi and Honorio, 2021; Amit et al., 2022; Haddouche and Guedj, 2023). Here, the assumptions that can be made about the function in question decisively influence the choice of divergence (or distance). A typical assumption is boundedness, which is used to apply some exponential moment inequality like the Hoeffding- or Bernstein-inequality (Rivasplata et al., 2020; Alquier, 2021). In many applications this is very restrictive, and several ways have been developed to circumvent it (Germain et al., 2009; Alquier et al., 2016; Catoni, 2004; Haddouche and Guedj, 2022; Rodríguez-Gálvez et al., 2024). However, the loss-functions occurring in this work can naturally be bounded by properties of the optimization algorithm. *Thus, we use a standard PAC-Bayesian argument involving the Donsker-Varadhan variational formulation and Hoeffding's inequality to get the generalization bounds.*

2.0.4 Minimization of the PAC-Bound and Choice of the Prior

The PAC-bound relates the true risk to other terms such as the empirical risk. Yet, it does not directly say anything about the absolute numbers. Thus, in learning procedures based on the PAC-Bayesian approach one typically aims to minimize the provided upper bound: Langford and Caruana (2001) compute non-vacuous numerical generalization bounds through a combination of PAC-bounds with a sensitivity analysis. Dziugaite and Roy (2017) extend this by minimizing the PAC-bound directly. Pérez-Ortiz et al. (2021)

also consider learning as minimization of the PAC-Bayesian upper bound and provide tight generalization bounds. Thiemann et al. (2017) provide sufficient conditions under which their resulting minimization problem is quasi-convex, which they solve by alternating minimization. Nevertheless, a common difficulty in learning with PAC-Bayesian bounds is the choice of the prior distribution, as it heavily influences the performance of the learned models and the generalization bound (Catoni, 2004; Dziugaite et al., 2021; Pérez-Ortiz et al., 2021). In part, and especially for the Kullback-Leibler divergence, this is due to the fact that the divergence term can dominate the bound, keeping the posterior close to the prior. This lead to the idea of choosing a data- or distribution-dependent prior (Seeger, 2002; Parrado-Hernández et al., 2012; Lever et al., 2013; Dziugaite and Roy, 2018; Pérez-Ortiz et al., 2021), which, by using an independent subset of the data set, gets optimized to yield a good performance. *We follow this approach and consider learning as minimization of the PAC-Bayesian upper-bound, however, applied to the context of learning-to-optimize. Also for us, the choice of the prior distribution is crucial for the performance of our learned algorithms, such that we use a data-dependent prior, which we construct similarly as in our prior work (Sucker et al., 2024).*

2.0.5 More Generalization Bounds

There are many areas of machine learning research that study generalization bounds and have not been discussed here. Importantly, the vast field of "stochastic optimization" (SO) provides generalization bounds for *specific* algorithms. In most of the cases, the concrete algorithms studied in SO generate a single point estimate by either minimizing the (regularized) empirical risk functional over a possibly large data set, or by repeatedly updating the point estimate based on a newly drawn (small) batch of samples. Then, one studies the properties of this point in terms of the stationarity measure of the true risk functional (Bottou et al., 2018; Davis and Drusvyatskiy, 2022; Bianchi et al., 2022). Further, as the setting in SO is more explicit, more assumptions have to be made. Typical assumptions are (weak) convexity (Shalev-Shwartz et al., 2009; Davis and Drusvyatskiy, 2019), bounded gradients (Défossez et al., 2022), bounded noise (Davis and Drusvyatskiy, 2022), or smoothness (Kavis et al., 2022). *All of these assumptions cannot be made without severely limiting the applicability of our results, because we consider an abstract loss function, an abstract optimization algorithm, and the problem of finding a distribution over its hyperparameters. Further, through learning, we go explicitly beyond analytically tractable quantities.*

3 Preliminaries and Notation

We will endow every topological space X with its Borel-σ-algebra $\mathcal{B}(X)$, and we assume it to be a Polish space, that is, it is separable and admits a complete metrization. Given two spaces X and Y, we write their product as $X \times Y$, and, for products with a generic number of terms, we use $\prod_{n=1}^{N} X_n$. Similarly, the product-σ-algebra of $\mathcal{B}(X)$ and $\mathcal{B}(Y)$ on $X \times Y$ is denoted by $\mathcal{B}(X) \otimes \mathcal{B}(Y)$. Further, for a generic number of terms, we use $\bigotimes_{n=1}^{N} \mathcal{B}(X_n)$ to denote the product-σ-algebra, and, if $X_n \equiv X$ for all $n = 1, ..., N$, also X^N and $\mathcal{B}(X)^{\otimes N}$.

Remark 2 *Countable products of Polish spaces are again Polish, and we have the equality* $\mathcal{B}\left(\prod_{n\in\mathbb{N}} X_n\right) = \bigotimes_{n\in\mathbb{N}} \mathcal{B}(X_n)$ *(Kallenberg, 2021, Lemma 1.2, p.11). Therefore, as we*

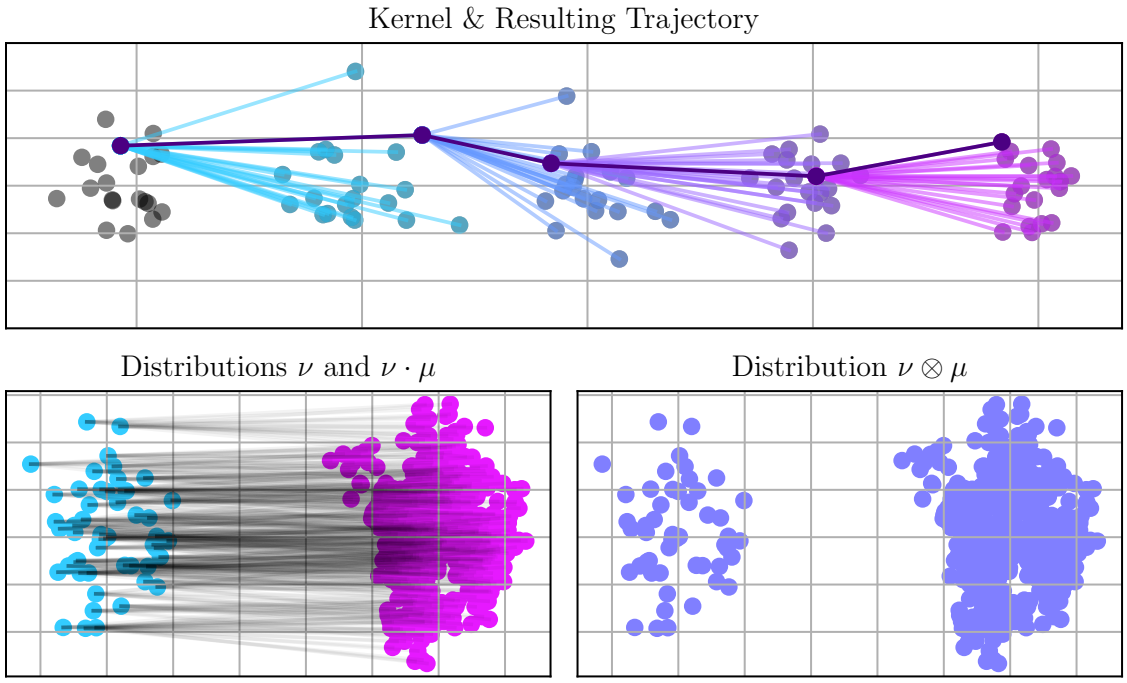

Figure 2: Visualization of kernels and their corresponding operations: The **top figure** visualizes the distributions $\mu(x_i, \cdot)$, $i = 0, ..., 3$, (colored dots) for four selected points $x_0, ..., x_3$. Here, each color represents one distribution $\mu(x_i, \cdot)$, $i = 0, ..., 3$, and the colored lines connecting x_i with each point from the next cluster should represent all the possible outcomes of $\mu(x_i, \cdot)$. The blackish line connecting the points $x_0, ..., x_3$ (and x_4) shows that, by selecting one sample from each $\mu(x_i, \cdot)$, a trajectory emerges from this process. The **lower left figure** visualizes how a distribution ν (blue) is transformed by the kernel μ into the distribution $\nu \cdot \mu$ (purple): At each point x we have a distribution $\mu(x, \cdot)$ (represented by several pink dots connected to one blue dot), and by integrating these points w.r.t. ν, the distribution $\nu \cdot \mu$ emerges. The **lower right figure** shows the distribution $\nu \otimes \mu$. The creation is the same as for $\nu \cdot \mu$, which is the marginal of $\nu \otimes \mu$. However, $\mu \otimes \nu$ is a measure on S^2, while $\nu \cdot \mu$ is a measure on S.

consider a discrete-time algorithm, all resulting product spaces in this work will be Polish spaces endowed with the Borel-σ-algebra, which coincides with the product-σ-algebra.

We use the same notation for measures: Given two measures ν_1 and ν_2 on X and Y, the corresponding product measure on $X \times Y$ is denoted by $\nu_1 \otimes \nu_2$. If the product involves a generic number of measures, this is denoted by $\bigotimes_{n=1}^{N} \nu_n$, and, if $\nu_n \equiv \nu$ for all $n = 1, ..., N$, we also use $\nu^{\otimes N}$. Further, if necessary, we will write the integral of a function $f : X \to \mathbb{R}$ w.r.t. ν in the operator notation, that is, $\nu[f] := \int_X \nu(dx)\ f(x) := \int_X f(x)\ \nu(dx)$. Especially, this applies when having multiple iterated integrals at once, in which case this has to be read from right to left instead from inside to outside. For example, if we integrate

$f : X \times Y \times Z \to \mathbb{R}$ w.r.t. $\nu \otimes \mu \otimes \lambda$, by Fubini's theorem we have:

$$\begin{aligned}\int_{X\times Y\times Z} f(x,y,z)\ (\nu\otimes\mu\otimes\lambda)(dx,dy,dz) &= \int_X \int_Y \int_Z f(x,y,z)\ \lambda(dz)\ \mu(dy)\ \nu(dx)\\ &= \int_X \nu(dx)\ \int_Y \mu(dy)\ \int_Z \lambda(dz)\ f(x,y,z)\,.\end{aligned}$$

In doing so, the integrand is closer to its "next" integrator and one can avoid many brackets. Especially, this applies to kernels, which are of fundamental importance for this work:

Definition 3 *Let $(X,\mathcal{X})$ and $(Y,\mathcal{Y})$ be measurable spaces. A* kernel *μ from X to Y is a mapping*

$$\mu : X \times \mathcal{Y} \to [0,\infty]\,,$$

such that $x \mapsto \mu(x,\mathsf{A})$ is measurable for every fixed $\mathsf{A} \in \mathcal{Y}$, and $\mathsf{A} \mapsto \mu(x,\mathsf{A})$ is a measure for every $x \in X$. μ is called a probability kernel*, if $\mu(x,Y) = 1$ for every $x \in X$.*

Notation 4 *We adopt the standard (abuse of) notation for kernels, that is, $\mu : X \times \mathcal{Y} \to [0,\infty]$ is abbreviated as $\mu : X \to Y$.*

For two (probability) kernels $\mu : X \to Y$ and $\nu : X \times Y \to Z$, the product of μ and ν is defined as the kernel $\mu \otimes \nu : X \to Y \times Z$ given by:

$$(\mu\otimes\nu)(x)[f] = \int_Y \mu(x,dy)\ \int_Z \nu((x,y),dz)\ f(y,z)\,,$$

where $f : Y \times Z \to \mathbb{R}$ is a measurable function. That is, for every $x \in X$, $(\mu\otimes\nu)(x,\cdot)$ is a measure on $Y \times Z$, such that for a set $\mathsf{A}\times\mathsf{B} \in \mathcal{B}(Y\times Z)$ it holds:

$$(\mu\otimes\nu)(x,\mathsf{A}\times\mathsf{B}) = \int_{\mathsf{A}} \nu((x,y),\mathsf{B})\ \mu(x,dy)\,.$$

Similarly, the composition of μ and ν is given by the kernel $\mu\cdot\nu : X \to Z$ defined through:

$$(\mu\cdot\nu)(x)[g] = \int_Y \mu(x,dy)\ \int_Z \nu((x,y),dz)\ g(z)\,,$$

where $g : Z \to \mathbb{R}$ is a measurable function. That is, for a measurable set $\mathsf{B} \subset Z$ it holds:

$$(\mu\cdot\nu)(x,\mathsf{B}) = \int_Y \nu((x,y),\mathsf{B})\ \mu(x,dy)\,.$$

Hence, it holds that $(\mu\cdot\nu)(x)[g] = (\mu\otimes\nu)(x)[\mathbb{1}_Y \otimes g]$, that is, $(\mu\cdot\nu)(x,\cdot)$ is the marginal of $(\mu\otimes\nu)(x,\cdot)$ on Z. Here, $\mathbb{1}_{\mathsf{A}}$ denotes the indicator function of a set A, which is equal to one for $x \in \mathsf{A}$ and zero else, that is, we have $\mathbb{1}_{\mathsf{A}}(x) = \delta_x(\mathsf{A})$, where δ_x is the Dirac-measure.

Example 5 *Having two random variables on the same probability space taking values in Polish spaces X and Y, say $\mathscr{X} : (\Omega,\mathcal{A},\mathbb{P}) \to X$ and $\mathscr{Y} : (\Omega,\mathcal{A},\mathbb{P}) \to Y$, there exists a regular version of the conditional distribution $\mathbb{P}_{\mathscr{Y}|\mathscr{X}}$ of $\mathscr{Y}$, given $\mathscr{X}$, such that their joint distribution can be factorized into the marginal and the conditional distribution:*

$$\mathbb{P}_{(\mathscr{X},\mathscr{Y})} = \mathbb{P}_{\mathscr{X}} \otimes \mathbb{P}_{\mathscr{Y}|\mathscr{X}}\,,$$

where $(x, \mathsf{A}) \mapsto \mathbb{P}_{\mathscr{Y}|\mathscr{X}=x}\{\mathsf{A}\}$ *is a probability kernel from* X *to* Y*. Similarly, the marginal of* $\mathscr{Y}$ *is given by:*

$$\begin{aligned}\mathbb{P}_{\mathscr{Y}}\{\mathsf{B}\} &= \mathbb{P}\{\mathscr{Y} \in \mathsf{B}\} = \mathbb{P}\{\mathscr{X} \in X,\ \mathscr{Y} \in \mathsf{B}\} = \mathbb{P}_{(\mathscr{X},\mathscr{Y})}\{X \times \mathsf{B}\} \\ &= \left(\mathbb{P}_{\mathscr{X}} \otimes \mathbb{P}_{\mathscr{Y}|\mathscr{X}}\right)\{X \times \mathsf{B}\} = \int_X \mathbb{P}_{\mathscr{Y}|\mathscr{X}=x}\{\mathsf{B}\}\ \mathbb{P}_{\mathscr{X}}(dx) = \left(\mathbb{P}_{\mathscr{X}} \cdot \mathbb{P}_{\mathscr{Y}|\mathscr{X}}\right)\{\mathsf{B}\}\ .\end{aligned}$$

For notational simplicity, when we have N elements $x_1, ..., x_N$ in some space X, and we refer to them all at once, that is, to the vector $(x_1, ..., x_N) \in X^N$, we indicate this by $x_{[N]} = (x_1, ..., x_N)$. For example, when integrating w.r.t. a product measure $\mathbb{P}_{\mathscr{X}}^{\otimes N}$ this allows for the more compact notation:

$$\mathbb{P}_{\mathscr{X}}^{\otimes N}[f] = \int_{X^N} \mathbb{P}_{\mathscr{X}}^{\otimes N}(dx_{[N]})\ f(x_{[N]}) := \int_{X^N} f(x_1, ..., x_N)\ \mathbb{P}_{\mathscr{X}}^{\otimes N}(dx_1, ..., dx_N)\,.$$

Similarly, given such a vector $x_{[N]}$, we refer to its components as $x_1, ..., x_N$, and the correspondence will be clear from the context. Furthermore, as this will turn up in the generalization bounds, the space of measures on an underlying space X is denoted by $\mathcal{M}(X)$, and all probability measures that are absolutely continuous w.r.t. a reference measure $\mu \in \mathcal{M}(X)$ are denote by $\mathcal{P}(\mu) := \{\nu \in \mathcal{M}(X)\ :\ \nu \ll \mu \text{ and } \nu[X] = 1\}$. In this context, the Kullback-Leiber divergence between two measures μ and ν is defined as:

$$D_{\mathrm{KL}}(\nu \parallel \mu) = \begin{cases} \nu[\log(f)] = \int_X \log(f(x))\ \nu(dx), & \nu \ll \mu \text{ with density } f\,, \\ +\infty, & \text{otherwise}\,. \end{cases}$$

Here, we have the following variational formulation due to Donsker and Varadhan (1975):

Lemma 6 (Variational Formulation by Donsker and Varadhan) *For any measurable and bounded function* $h : X \to \mathbb{R}$*, it holds that:*

$$\log\left(\int_X \exp(h(x))\ \mu(dx)\right) = \sup_{\nu \in \mathcal{P}(\mu)} \left\{\int_X h(x)\ \nu(dx) - D_{\mathrm{KL}}(\nu \parallel \mu)\right\}\,.$$

Finally, the following definitions are used in the so-called *monotone-class argument*, which is needed to derive the distribution of the trajectory of the algorithm:

Definition 7 *Let* $(X, \mathcal{X})$ *be a measurable space. A class* $\mathcal{C} \subset \mathcal{X}$ *is called a* π-system*, if it is closed under finite intersection, that is,* $\mathsf{A}, \mathsf{B} \in \mathcal{C}$ *implies* $\mathsf{A} \cap \mathsf{B} \in \mathcal{C}$*. Furthermore, a class* $\mathcal{D} \subset \mathcal{X}$ *is called a* λ-system*, if it contains* X *and it is closed under proper differences and increasing limits. That is, we require* $X \in \mathcal{D}$*, that* $\mathsf{A}, \mathsf{B} \in \mathcal{D}$ *with* $\mathsf{A} \supset \mathsf{B}$ *implies* $\mathsf{A} \setminus \mathsf{B} \in \mathcal{D}$*, and that* $\mathsf{A}_1, \mathsf{A}_2, ... \in \mathcal{D}$ *with* $\mathsf{A}_n \uparrow \mathsf{A}$ *implies* $\mathsf{A} \in \mathcal{D}$*.*

The following theorem can be found in Kallenberg (2021, Thm. 1.1, p.10).

Theorem 8 (Monotone Classes) *For any* π*-system* $\mathcal{C}$ *and* λ*-system* $\mathcal{D}$ *in a measurable space* X*, it holds that:*

$$\mathcal{C} \subset \mathcal{D} \implies \sigma(\mathcal{C}) \subset \mathcal{D},$$

where $\sigma(\mathcal{C})$ *is the* σ*-algebra generated by* $\mathcal{C}$*.*

4 The Probabilistic Model

In this section we derive the distribution of the trajectories of the algorithm. Starting from the ad-hoc motivation in the introduction, especially Equation (1), we aim for a probability space that describes the trajectories generated by the algorithm depending on the hyperparameters and parameters. For this, we first introduce a probability space $(\Omega_{\text{pre}}, \mathcal{A}_{\text{pre}}, \mathbb{P}_{\text{pre}})$ which describes the underlying randomness outlined in the introduction. Then, in Definition 13, we define the *transition kernel* of $\mathcal{A}$, which is the needed measure-theoretic equivalent to Equation (1). Afterwards, in Theorem 19, we show that the transition kernel (together with the initialization) yields a unique probability kernel onto the space of trajectories, which describes the distribution of $\xi = (\xi^{(t)})_{t\in\mathbb{N}_0}$ depending on the hyperparameters α and parameters θ. This, in turn, allows us to define the probability space $(\Omega, \mathcal{A}, \mathbb{P})$, which describes the joint distribution of the hyperparameters, parameters, and the resulting trajectory ξ in the correct way, and which will be used to derive the generalization results. Since this is quite technical, we want to stress once again that this builds the fundament of our principled treatment of learning-to-optimize with theoretical guarantees, and it is absolutely necessary for giving generalization bounds for the convergence rate and stopping times. For this, our model relies on the following two mild assumptions:

Assumption 9 *We are given four Polish probability spaces: the state space $(S, \mathcal{B}(S), \mathbb{P}_{\mathscr{S}})$, the parameter space $(P, \mathcal{B}(P), \mathbb{P}_{\mathscr{P}})$, the hyperparameter space $(H, \mathcal{B}(H), \mathbb{P}_{\mathscr{H}})$, and the randomization space $(R, \mathcal{B}(R), \mathbb{P}_{\mathscr{R}})$.*

Remark 10 *It is assumed implicitly that the state space* encompasses *the space of the optimization variable as a subspace, and we denote the corresponding projection from S onto this subspace by Π_S. This is done, for example, to be able to model algorithms that depend on a finite number of other variables, such as previous iterates.*

Assumption 11 *We are given a measurable function $\ell : S \times P \to [0, \infty]$, the* loss function*, and a measurable map $\mathcal{A} : H \times P \times S \times R \to S$, the* algorithmic update*.*

As stated in the introduction, starting from some initialization $\xi^{(0)} \in S$, the algorithm generates a sequence of iterates $\xi = (\xi^{(t)})_{t\in\mathbb{N}_0}$ as follows:

$$\xi^{(t+1)} = \mathcal{A}(\alpha, \theta, \xi^{(t)}, \eta^{(t+1)}), \quad t \geq 1\,,$$

where $\alpha \in H$, $\theta \in P$, and $\eta^{(t+1)} \in R$. Thus, ξ is a discrete time stochastic process taking values in the space S, and the goal is to learn the hyperparameters $\alpha \in H$ on a dataset of parameters $\theta_{[N]} := (\theta_1, ..., \theta_N) \in P^N$.

Example 12 *(i) Consider stochastic gradient descent to minimize the parametric empirical risk $\ell(x, \theta) := \frac{1}{m}\sum_{i=1}^m f_i(x, \theta)$. In each iteration, the algorithm samples an index j uniformly in $\{1, ..., m\}$ and performs the update:*

$$\xi^{(t+1)} = \xi^{(t)} - \alpha \nabla f_j(\xi^{(t)}, \theta)\,,$$

where $\alpha > 0$ is a step-size. This can be summarized into a single mapping $\mathcal{A}$ as:

$$\mathcal{A}(\alpha, \theta, \xi^{(t)}, \eta^{(t+1)}) := \xi^{(t)} - \alpha \sum_{i=1}^m \mathbb{1}_{\{i\}}(\eta^{(t+1)}) \nabla f_i(\xi^{(t)}, \theta)\,,$$

where $\eta^{(t+1)} \sim \mathcal{U}\{1,...,m\}$. *Thus, it holds* $S = \mathbb{R}^n$, $H = [0,\infty)$, *and* $R = \{1,...,m\}$.

(ii) Consider an update of the form:

$$\xi^{(t+1)} := \begin{pmatrix} h^{(t+1)} \\ x^{(t+1)} \end{pmatrix} := \begin{pmatrix} \mathcal{N}_1(\alpha_1, \theta, h^{(t)}, \eta^{(t+1)}) \\ x^{(t)} - \mathcal{N}_2(\alpha_2, \theta, \xi^{(t)}, \eta^{(t+1)}) \end{pmatrix},$$

where, additionally to updating the iterates $x^{(t)} \in \mathbb{R}^n$ *with a neural network* $\mathcal{N}_2$, *one updates a hidden state* $h^{(t)} \in \mathbb{R}^m$ *with another neural network* $\mathcal{N}_1$. *In this case, the state would consist of* $\xi^{(t)} = (h^{(t)}, x^{(t)}) \in \mathbb{R}^{m+n}$, Π_S *would be the projection from* $\mathbb{R}^{m+n}$ *onto* $\mathbb{R}^n$ *with* $\Pi_S(\xi^{(t)}) = x^{(t)}$, *and the hyperparameters* α *would be given by the parameters of these two networks, that is, the tuple* $\alpha = (\alpha_1, \alpha_2)$.

4.1 The Distribution of the Trajectory on $S^{\mathbb{N}_0}$

We model the underlying stochasticity through the following probability space, which combines the four independent sources of randomness in the canonical way. Since we want to learn on a dataset of size N, we have to use the N-fold product of several of these spaces. Thus, define the measurable space $(\Omega_{\text{pre}}, \mathcal{A}_{\text{pre}})$ through:

$$\Omega_{\text{pre}} := H \times P^N \times S^N \times \left(R^{\mathbb{N}}\right)^N, \quad \mathcal{A}_{\text{pre}} := \mathcal{B}(\Omega_{\text{pre}}),$$

and endow it with the probability measure

$$\mathbb{P}_{\text{pre}} := \mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}}^{\otimes N} \otimes \mathbb{P}_{\mathscr{I}}^{\otimes N} \otimes \bigotimes_{n=1}^{N} \mathbb{P}_{\mathscr{R}}^{\otimes \mathbb{N}}.$$

We denote the canonical process on Ω_{pre}, that is, the coordinate projections, by

$$\mathscr{X}_{\text{pre}} := \left(\mathscr{H}, \mathscr{P}_{[N]}, \mathscr{I}_{[N]}, (\mathscr{R}^{(t)})_{t\in\mathbb{N},[N]}\right).$$

Thus, it holds that $\mathscr{H} \sim \mathbb{P}_{\mathscr{H}}$, $\mathscr{P}_1, ..., \mathscr{P}_N \overset{iid}{\sim} \mathbb{P}_{\mathscr{P}}$, $\mathscr{I}_1, ..., \mathscr{I}_N \overset{iid}{\sim} \mathbb{P}_{\mathscr{I}}$, and all $\mathscr{R}_n^{(t)} \sim \mathbb{P}_{\mathscr{R}}$, $t \in \mathbb{N}$, $n = 1, ..., N$, are i.i.d. Then, by definition of $(\Omega_{\text{pre}}, \mathcal{A}_{\text{pre}}, \mathbb{P}_{\text{pre}})$ and Fubini's theorem, for any cylinder set $\mathsf{B}_1 \times ... \times \mathsf{B}_N \in \mathcal{B}(S)^{\otimes N}$ we have the following factorization:

$$\begin{aligned}
&\mathbb{P}_{\text{pre}}\left\{\left(\mathcal{A}(\mathscr{H}, \mathscr{P}_1, \mathscr{I}_1, \mathscr{R}_1^{(1)}), ..., \mathcal{A}(\mathscr{H}, \mathscr{P}_N, \mathscr{I}_N, \mathscr{R}_N^{(1)})\right) \in \mathsf{B}_1 \times ... \times \mathsf{B}_N\right\} \\
&= \int_{H\times P^N \times S^N} \prod_{n=1}^{N} \mathbb{P}_{\mathscr{R}}\left\{\mathcal{A}(\alpha, \theta_n, x_n, \cdot) \in \mathsf{B}_n\right\} \left(\mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}}^{\otimes N} \otimes \mathbb{P}_{\mathscr{I}}^{\otimes N}\right)(d\alpha, d\theta_{[N]}, dx_{[N]}),
\end{aligned}$$

which motivates the following definition:

Definition 13 *The* transition kernel *of* $\mathcal{A}$ *is given through*

$$\gamma : H \times P \times S \to S, \ ((\alpha, \theta, x), \mathsf{B}) \mapsto \mathbb{P}_{\mathscr{R}}\left\{\mathcal{A}(\alpha, \theta, x, \cdot) \in \mathsf{B}\right\}.$$

The joint transition kernel *of* $\mathcal{A}$ *is given through*

$$\Gamma : H \times P^N \times S^N \to S^N, \ \left((\alpha, \theta_{[N]}, x_{[N]}), \mathsf{B}_1 \times ... \times \mathsf{B}_N\right) \mapsto \prod_{n=1}^{N} \mathbb{P}_{\mathscr{R}}\left\{\mathcal{A}(\alpha, \theta_n, x_n, \cdot) \in \mathsf{B}_n\right\},$$

that is, $\Gamma(\alpha, \theta_{[N]}, x_{[N]}) = \bigotimes_{n=1}^{N} \gamma(\alpha, \theta_n, x_n)$.

Remark 14 *As will be shown below, for every $(\alpha, \theta) \in H \times P$, the distribution of the process $\xi = \left(\xi^{(t)}\right)_{t \in \mathbb{N}_0}$ generated by $\mathcal{A}(\alpha, \theta, \cdot, \cdot)$ is uniquely defined by the transition kernel $\gamma(\alpha, \theta, \cdot) : S \to S$ and the initial distribution $\mathbb{P}_{\mathscr{I}}$. This is the probabilistic* generalization *of the fact that the trajectory of a deterministic algorithm is uniquely defined by the initialization and its update-step.*

Example 15 *(i) The transition kernel of stochastic gradient descent from Example 12 is given by:*

$$\mathbb{P}_{\mathscr{R}}\left\{\mathcal{A}(\alpha, \theta, x, \cdot) \in \mathsf{B}\right\} := \mathcal{U}_{\{1,\dots,m\}}\left\{x - \alpha \sum_{i=1}^{m} \mathbb{1}_{\{i\}}(\cdot) \nabla f_i(x, \theta) \in \mathsf{B}\right\}.$$

Basically, this is the transition kernel used by Bianchi et al. (2022), and our definition is a direct generalization of it.

(ii) The transition kernel is a direct generalization of the usual algorithmic update: Consider a deterministic algorithm. Then, it holds $\gamma(\alpha, \theta, x) = \delta_{\mathcal{A}(\alpha,\theta,x)}$, and we get:

$$\mathbb{P}_{\mathscr{R}}\left\{\mathcal{A}(\alpha, \theta, x) \in \mathsf{B}\right\} = \delta_{\mathcal{A}(\alpha,\theta,x)}[\mathsf{B}] = \mathbb{1}_{\mathsf{B}}(\mathcal{A}(\alpha, \theta, x)).$$

Thus, integrating w.r.t. γ just yields the new iterate. Taking this approach, we recover the average-case setting of Pedregosa and Scieur (2020); Scieur and Pedregosa (2020).

Lemma 16 *Suppose that $\mathcal{A}$ satisfies Assumption 11. Then the transition kernel γ is a probability kernel from $H \times P \times S$ to S, and the joint transition kernel Γ is a probability kernel from $H \times P^N \times S^N$ to S^N.*

Proof By definition, it holds that:

$$\gamma((\alpha, \theta, x), \mathsf{B}) = \left(\mathbb{P}_{\mathscr{R}} \circ \mathcal{A}(\alpha, \theta, x, \cdot)^{-1}\right)\{\mathsf{B}\}.$$

Thus, since $\mathcal{A} : H \times P \times S \times R \to S$ is measurable, and $\mathbb{P}_{\mathscr{R}}$ can be seen as the constant kernel from $H \times P \times S \to R$, we get from Kallenberg (2021, Lemma 3.2 (ii), p.56) that γ is a probability kernel from $H \times P \times S$ to S. Hence, by definition of Γ, if $(\alpha, \theta_{[N]}, x_{[N]})$ is given, we also get that $\Gamma(\alpha, \theta_{[N]}, x_{[N]})$ is a measure on S^N. Therefore, it remains to show measurability of $\Gamma(\cdot, \mathsf{A})$ for fixed $\mathsf{A} \in \mathcal{B}(S)^{\otimes N}$. We do this by a monotone-class argument. For this, define the classes of sets:

$$\mathcal{D} := \{\mathsf{A} \in \mathcal{B}(S)^{\otimes N} \ : \ (\alpha, \theta_{[N]}, x_{[N]}) \mapsto \Gamma(\alpha, \theta_{[N]}, x_{[N]}, \mathsf{A}) \text{ is measurable}\}$$
$$\mathcal{C} := \{\mathsf{A}^0 \times ... \times \mathsf{A}^N \ : \ \mathsf{A}^0, ..., \mathsf{A}^K \in \mathcal{B}(S)\}.$$

$\mathcal{C}$ is the class of cylinder sets, which, by definition, is a $\cap$-stable generator of the product-σ-field. Thus, $\mathcal{C}$ is a π-system with $\sigma(\mathcal{C}) = \mathcal{B}(S)^{\otimes N}$. Furthermore, for any $\mathsf{B}_1 \times ... \times \mathsf{B}_N \in \mathcal{C}$, it holds that:

$$\Gamma\left((\alpha, \theta_{[N]}, x_{[N]}), \mathsf{B}_1 \times ... \times \mathsf{B}_N\right) = \prod_{n=1}^{N} \gamma(\alpha, \theta_n, x_n, \mathsf{B}_n).$$

Since γ is a probability kernel, that is, measurable for fixed (α, θ_n, x_n), it follows that $\mathcal{C} \subset \mathcal{D}$. Thus, it remains to show that $\mathcal{D}$ is a λ-system. Clearly, it holds that $S^N \in \mathcal{C} \subset \mathcal{D}$. Thus, take $\mathsf{A}, \mathsf{B} \in \mathcal{D}$ with $\mathsf{A} \supset \mathsf{B}$. Then, since $\Gamma\left(\alpha, \theta_{[N]}, x_{[N]}\right)$ was already shown to be a probability measure for each fixed $(\alpha, \theta_{[N]}, x_{[N]})$, we have the following pointwise equality:

$$\Gamma\left((\alpha, \theta_{[N]}, x_{[N]}), \mathsf{A} \setminus \mathsf{B}\right) = \Gamma\left((\alpha, \theta_{[N]}, x_{[N]}), \mathsf{A}\right) - \Gamma\left((\alpha, \theta_{[N]}, x_{[N]}), \mathsf{B}\right) .$$

Therefore, since $\mathsf{A}, \mathsf{B} \in \mathcal{D}$, we have that the right-hand side is measurable, which in turn implies $\mathsf{A} \setminus \mathsf{B} \in \mathcal{D}$. Finally, for $\mathsf{A}_1, \mathsf{A}_2, ... \in \mathcal{D}$ with $\mathsf{A}_n \uparrow \mathsf{A}$, by continuity of a measure, we have the pointwise equality:

$$\Gamma\left((\alpha, \theta_{[N]}, x_{[N]}), \mathsf{A}\right) = \lim_{n\to\infty} \Gamma\left((\alpha, \theta_{[N]}, x_{[N]}), \mathsf{A}_n\right) .$$

Since limits of measurable functions are measurable, it follows that $\mathsf{A} \in \mathcal{D}$. Therefore, $\mathcal{D}$ is a λ-system, and Theorem 8 yields $\mathcal{B}(S)^{\otimes N} \subset \mathcal{D}$. Thus, Γ is a probability kernel from $H \times P^N \times S^N$ to S^N. ■

Given a starting point $\xi^{(0)}$, we can compute the t-th iterate $\xi^{(t)}$ by applying the algorithm $\mathcal{A}$ t-times recursively to $\xi^{(0)}$. Similarly, given an initial distribution $\mathbb{P}_{\mathscr{I}}$, we get the distribution of $\xi^{(t)}$ by applying the transition kernel γ t-times recursively to $\mathbb{P}_{\mathscr{I}}$:

Definition 17 *The* transition semi-group $(\gamma^t)_{t\in\mathbb{N}_0}$ *is defined recursively by:*

$$\gamma^0(\alpha, \theta, x) := \delta_x, \quad \gamma^t(\alpha, \theta, x) := \gamma^{t-1}(\alpha, \theta, x) \cdot \gamma(\alpha, \theta, \cdot), \quad t \in \mathbb{N} .$$

Similarly, the joint transition semi-group *is defined by:*

$$\Gamma^0(\alpha, \theta_{[N]}, x_{[N]}) := \delta_{x_{[N]}}, \quad \Gamma^t(\alpha, \theta_{[N]}, x_{[N]}) := \Gamma^{t-1}(\alpha, \theta_{[N]}, x_{[N]}) \cdot \Gamma(\alpha, \theta_{[N]}, \cdot), \quad t \in \mathbb{N} .$$

Remark 18 *(i) γ^t models the t-fold application of the algorithm: applying γ^t to the distribution of $\xi^{(t_0)}$ yields the distribution of $\xi^{(t_0+t)}$. Thus, the resulting process is* time-homogeneous*. Nevertheless, one can still model algorithms that have a parameter $T^{(t)}$, which itself models the progression of time, as, for example, in Nesterov's accelerated gradient descent (Nesterov, 1983). The prerequisite for this is that the update of this time-parameter can be written in closed-form, that is, $T^{(t+1)} = f(T^{(t)})$ for some measurable function f, such that $T^{(t)}$ can be included into the state $\xi^{(t)}$.*

(ii) For every $(\alpha, \theta) \in H \times P$, the process ξ generated by $\mathcal{A}(\alpha, \theta, \cdot, \cdot)$ is a time-homogeneous Markov process (w.r.t. the natural filtration generated by the iterates) with initial distribution $\mathbb{P}_{\mathscr{I}}$ and transition semi-group $\left(\gamma^t(\alpha, \theta, \cdot)\right)_{t\in\mathbb{N}_0}$. This can be seen by noting that, if α and θ are fixed*, the equation*

$$\xi^{(t+1)} = \mathcal{A}(\alpha, \theta, \xi^{(t)}, \eta^{(t+1)})$$

corresponds to the so-called functional representation *of a Markov process.*

Through this recursive definition, and as is shown in Lemma 45 in the appendix, we have that the factorization of Γ extends to the joint transition semi-group $\left(\Gamma^t\right)_{t\in\mathbb{N}_0}$, which will ultimately result in a corresponding factorization of the joint distribution of the processes $\xi_1, ..., \xi_N$ corresponding to the parameters $\theta_1, ..., \theta_N$. By Klenke (2013, Corollary 14.44, p.299), for every $\alpha \in H$ and $\theta \in P$ there exist a unique probability measure $\psi_{\alpha,\theta}$ on $S^{\mathbb{N}_0}$, such that for any natural numbers $0 = t_0 < t_1 < ... < t_K$, it holds that:

$$\psi_{\alpha,\theta} \circ \mathscr{X}_J^{-1} = \mathbb{P}_{\mathscr{I}} \otimes \bigotimes_{k=0}^{K-1} \gamma^{t_{k+1}-t_k}(\alpha, \theta, \cdot)\,, \tag{2}$$

where $J := \{t_0, ..., t_K\}$ and $\mathscr{X}_J : S^{\mathbb{N}_0} \to S^{K+1}$ denotes the corresponding coordinate projection. This means that the finite-dimensional distribution of $\psi_{\alpha,\theta}$ corresponding to the time-points $t_0, ..., t_K$ is given by the distribution of the iterates $\left(\xi^{(t_0)}, ..., \xi^{(t_K)}\right)$ generated by $\mathcal{A}(\alpha, \theta, \cdot, \cdot)$ (with initial distribution $\mathbb{P}_{\mathscr{I}}$). Since the distribution of a stochastic process is uniquely determined by its finite-dimensional distributions (Kallenberg, 2021, Prop. 4.2, p.84), this implies that, for every $(\alpha, \theta) \in H \times P$, there is exactly one distribution on the space of trajectories $S^{\mathbb{N}_0}$, namely $\psi_{\alpha,\theta}$, that describes the iterates corresponding to $\mathcal{A}(\alpha, \theta, \cdot, \cdot)$. Similarly, there exists a unique probability measure $\Psi_{\alpha,\theta_{[N]}}$ on $\left(S^N\right)^{\mathbb{N}_0}$, such that:

$$\Psi_{\alpha,\theta_{[N]}} \circ \mathscr{X}_{J,[N]}^{-1} = \mathbb{P}_{\mathscr{I}}^{\otimes N} \otimes \bigotimes_{k=0}^{K-1} \Gamma^{t_{k+1}-t_k}(\alpha, \theta_{[N]}, \cdot)\,, \tag{3}$$

where $\mathscr{X}_{J,[N]} : \left(S^N\right)^{\mathbb{N}_0} \to \left(S^N\right)^{K+1}$ denotes the corresponding coordinate projections on $\left(S^N\right)^{\mathbb{N}_0}$. As before, $\Psi_{\alpha,\theta_{[N]}}$ is the unique measure on $\left(S^N\right)^{\mathbb{N}_0}$ that describes the distribution of the trajectory of $\xi_{[N]} = (\xi_1, ..., \xi_N)$ in S^N which is generated by the collection $(\mathcal{A}(\alpha, \theta_1, \cdot, \cdot), ..., \mathcal{A}(\alpha, \theta_N, \cdot, \cdot))$. Intuitively, and as will be shown later on, this is (up to reordering) the same as considering N individual processes $\xi_1, ..., \xi_N$ on S generated by $\mathcal{A}(\alpha, \cdot, \cdot, \cdot)$ on the problem instances $\theta_1, ..., \theta_N$. First, however, we show that both $\psi_{\alpha,\theta}$ and $\Psi_{\alpha,\theta_{[N]}}$ can be "summarized" into a unique kernel:

Theorem 19 *Suppose that Assumptions 9 and 11 hold. Then, the map*

$$\psi : (H \times P) \times \mathcal{B}(S)^{\otimes\mathbb{N}_0} \to [0,1],\ ((\alpha,\theta), \mathsf{B}) \mapsto \psi_{\alpha,\theta}(\mathsf{B})$$

is the unique probability kernel from $H \times P$ to $S^{\mathbb{N}_0}$, such that (2) *holds. Similarly, the map*

$$\Psi : (H \times P^N) \times \mathcal{B}\left(S^N\right)^{\otimes\mathbb{N}_0} \to [0,1],\ \left((\alpha, \theta_{[N]}), \mathsf{B}\right) \mapsto \Psi_{\alpha,\theta_{[N]}}(\mathsf{B})$$

is the unique probability kernel from $H \times P^N$ to $\left(S^N\right)^{\mathbb{N}_0}$, such that (3) *holds.*

Proof By construction, we have that $\psi(\alpha, \theta, \cdot)$ is a probability measure on $S^{\mathbb{N}_0}$ for each $(\alpha, \theta) \in H \times P$. Thus, we only have to show the measurability. Again, this follows by a monotone-class argument. A detailed proof is given in Appendix B. ■

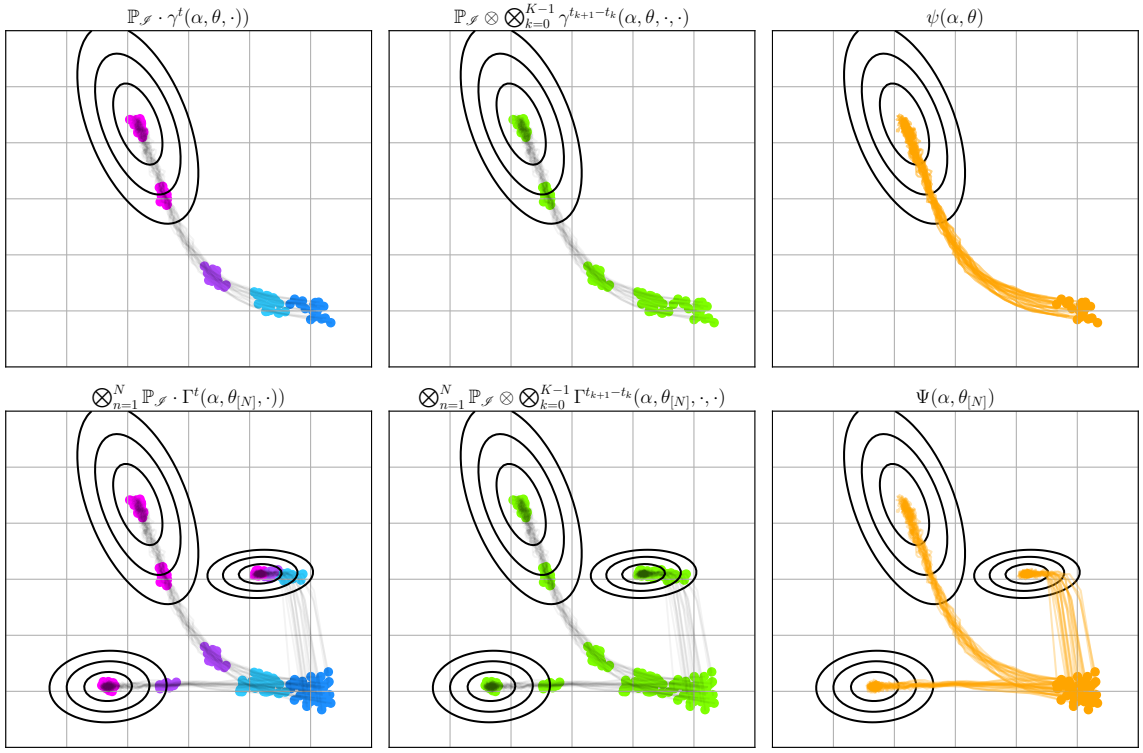

Figure 3: Visualization of the (joint) transition kernel: The upper row shows how the kernel $\gamma(\alpha, \theta, \cdot)$ acts on the initial distribution: The iterative concatenation (upper left) transforms the initial distribution of $\xi^{(0)}$ on S (dark blue) into the distributions of $\xi^{(t_1)}$ (light blue), $\xi^{(t_2)}$ (purple), $\xi^{(t_3)}$ (pink), and $\xi^{(t_4)}$ (light pink). Similarly, the iterative product (upper middle) transforms the initial distribution on S into a distribution on S^5, namely the joint distribution of $(\xi^{(0)}, \xi^{(t_1)}, ..., \xi^{(t_4)})$. Then, this yields the unique distribution $\psi(\alpha, \theta, \cdot)$ on $S^{\mathbb{N}_0}$ (upper right) for the whole trajectory (orange lines). The lower row shows the same thing on the space S^N, just that the initial distribution now is given by $\bigotimes_{n=1}^{N} \mathbb{P}_{\mathscr{I}}$ and the corresponding kernel is $\Gamma(\alpha, \theta_{[N]}, \cdot)$, which acts on all problem instances $\theta_1, ..., \theta_N$ at once.

Remark 20 *Assumption 11 is needed for measurability, while Assumptions 9 is needed for the existence of $\psi_{\alpha,\theta}$ and $\Psi_{\alpha,\theta_{[N]}}$.*

Theorem 19 states that the distribution of the trajectory on $S^{\mathbb{N}_0}$ depends *measurably* on the parameters of the problem and the hyperparameters of the algorithm. This allows to define a new probability space, which describes these three directly. Before doing this, the next lemma states that the factorization of Γ actually extends to a factorization of Ψ, that is, the resulting processes $(\xi_1, ..., \xi_N)$ are conditionally independent. Here, for a lack of a better notation, we will also write cylinder sets $\mathsf{A} \in \mathcal{B}\left(S^N\right)^{\otimes \mathbb{N}_0}$ with their corresponding coordinates $\mathsf{A} = \mathsf{A}_1 \times ... \times \mathsf{A}_N$, where each $\mathsf{A}_i \in \mathcal{B}\left(S^{\mathbb{N}_0}\right)$. This is justified

by the fact that $\left(S^N\right)^{\mathbb{N}_0} \cong \left(S^{\mathbb{N}_0}\right)^N$ and $\mathcal{B}\left(S^N\right)^{\otimes\mathbb{N}_0} \cong \mathcal{B}\left(S^{\mathbb{N}_0}\right)^{\otimes N}$. Further, $\mathcal{B}\left(S^N\right)^{\otimes\mathbb{N}_0}$ is generated by cylinder sets of the form $\left(\mathsf{B}_1^0 \times ... \times \mathsf{B}_N^0\right) \times ... \times \left(\mathsf{B}_1^K \times ... \times \mathsf{B}_N^K\right) \times \prod_{k>K} S^N$, while $\mathcal{B}\left(S^{\mathbb{N}_0}\right)^{\otimes N}$ is generated by cylinder sets of the form $\left(\mathsf{B}_1^0 \times ... \times \mathsf{B}_1^K \times \prod_{k>K} S\right) \times ... \times \left(\mathsf{B}_N^0 \times ... \times \mathsf{B}_N^K \times \prod_{k>K} S\right)$, which are just a reordering of each other.

Lemma 21 *Suppose that Assumptions 9 and 11 hold. Then, for any $(\alpha, \theta_{[N]}) \in H \times P^N$, and any set $\mathsf{A}_1 \times ... \times \mathsf{A}_N \in \mathcal{B}\left(S^N\right)^{\otimes\mathbb{N}_0}$, we have the following factorization:*

$$\Psi(\alpha, \theta_{[N]}, \mathsf{A}_1 \times ... \times \mathsf{A}_N) = \prod_{i=1}^{N} \psi(\alpha, \theta_n, \mathsf{A}_n)\,.$$

Thus, it holds that:

$$\Psi(\alpha, \theta_{[N]}) \cong \bigotimes_{n=1}^{N} \psi(\alpha, \theta_n)\,.$$

Proof $\Psi(\alpha, \theta_{[N]})$ is uniquely defined by its values on a $\cap$-stable generator of $\mathcal{B}\left(S^N\right)^{\otimes\mathbb{N}_0}$. As stated above, such a generator is given by cylinder sets of the form:

$$\underbrace{\left(\mathsf{B}_1^0 \times ... \times \mathsf{B}_N^0\right)}_{=:\mathsf{C}^0} \times ... \times \underbrace{\left(\mathsf{B}_1^K \times ... \times \mathsf{B}_N^K\right)}_{=:\mathsf{C}^K} \times \prod_{k>K} S^N\,.$$

Thus, denoting $J := \{0, ..., K\}$, we get:

$$\begin{aligned}
&\Psi(\alpha, \theta_{[N]}, \mathsf{A}_1 \times ... \times \mathsf{A}_N) = \Psi_{\alpha,\theta_{[N]}}\left\{\mathsf{A}_1 \times ... \times \mathsf{A}_N\right\} \\
&= \left(\Psi_{\alpha,\theta_{[N]}} \circ \mathscr{X}_{J,[N]}^{-1}\right)\left\{\mathsf{C}^0 \times ... \times \mathsf{C}^K\right\} = \left(\mathbb{P}_{\mathscr{I}}^{\otimes N} \otimes \bigotimes_{k=0}^{K-1} \Gamma(\alpha, \theta_{[N]}, \cdot)\right)\left\{\mathsf{C}^0 \times ... \times \mathsf{C}^K\right\} \\
&= \int_{S^N} \mathbb{P}_{\mathscr{I}}^{\otimes N}(dx_{[N]}) \left(\delta_{x_{[N]}} \otimes \bigotimes_{k=0}^{K-1} \Gamma(\alpha, \theta_{[N]}, \cdot)\right)\left\{\mathsf{C}^0 \times ... \times \mathsf{C}^K\right\}\,.
\end{aligned}$$

By Lemma 45, this is the same as:

$$= \int_{S^N} \mathbb{P}_{\mathscr{I}}^{\otimes N}(dx_{[N]}) \prod_{i=1}^{N} \left(\delta_{x_n} \otimes \bigotimes_{k=0}^{K-1} \gamma(\alpha, \theta_n, \cdot)\right)\left\{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K\right\}\,.$$

By Fubini's theorem, this is the same as:

$$\begin{aligned}
&= \prod_{i=1}^{N} \int_S \mathbb{P}_{\mathscr{I}}(dx_n) \left(\delta_{x_n} \otimes \bigotimes_{k=0}^{K-1} \gamma(\alpha, \theta_n, \cdot)\right)\left\{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K\right\} \\
&= \prod_{i=1}^{N} \left(\mathbb{P}_{\mathscr{I}} \otimes \bigotimes_{k=0}^{K-1} \gamma(\alpha, \theta_n, \cdot)\right)\left\{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K\right\} \\
&= \prod_{i=1}^{N} \left(\psi_{\alpha,\theta_n} \circ \mathscr{X}_J^{-1}\right)\left\{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K\right\} = \prod_{i=1}^{N} \psi_{\alpha,\theta_n}\left\{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K \times \prod_{k>K} S\right\}\,.
\end{aligned}$$

Since $\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K \times \prod_{k>K} S$ is the n-th "coordinate" of the set $\mathsf{C}^0 \times .... \times \mathsf{C}^K \times \prod_{k>K} S^N$, this shows that all finite-dimensional marginals of $\Psi(\alpha, \theta_{[N]})$ coincide with the corresponding ones of $\otimes_{n=1}^N \psi(\alpha, \theta_n)$. Thus, by the Kolmogorov Extension Theorem (Klenke, 2013, Thm. 14.36, p.295) (uniqueness of the projective limit), we get that

$$\Psi(\alpha, \theta_{[N]}) \cong \bigotimes_{n=1}^{N} \psi(\alpha, \theta_n)\,,$$

that is, up to reordering, $\Psi(\alpha, \theta_{[N]})$ is just the product of $\psi(\alpha, \theta_n)$, $n = 1, ..., N$. ■

Figure 3 visualizes these two constructions: The left column shows how the initial distributions $\mathbb{P}_{\mathscr{I}}$ (upper row) and $\mathbb{P}_{\mathscr{I}}^{\otimes N}$ (lower row) are transformed by the transition semi-group $\left(\gamma^t\right)_{t\in\mathbb{N}_0}$ and $\left(\Gamma^t\right)_{t\in\mathbb{N}_0}$, respectively. This gives the distribution of the iterates $\xi^{(t_0)}, ..., \xi^{(t_K)} \in S$ and $\xi_{[N]}^{(t_0)}, ..., \xi_{[N]}^{(t_K)} \in S^N$, respectively. Then, the middle column shows the corresponding joint distributions of $(\xi^{(t_0)}, ..., \xi^{(t_K)}) \in S^{K+1}$ and $(\xi_{[N]}^{(t_0)}, ..., \xi_{[N]}^{(t_K)}) \in \left(S^N\right)^{K+1}$. Finally, the right column visualizes ψ and Ψ on $S^{\mathbb{N}_0}$ and $\left(S^N\right)^{\mathbb{N}_0}$, respectively.

Remark 22 *Based on Lemma 21, to simplify the argument, we can (and often will) identify the measure $\Psi(\alpha, \theta_{[N]})$ on $\left(S^N\right)^{\mathbb{N}_0}$ with the measure $\bigotimes_{n=1}^N \psi(\alpha, \theta_n)$ on $\left(S^{\mathbb{N}_0}\right)^N$, and correspondingly $\left(\left(S^N\right)^{\mathbb{N}_0}, \mathcal{B}\left(S^N\right)^{\otimes\mathbb{N}_0}\right)$ with $\left(\left(S^{\mathbb{N}_0}\right)^N, \mathcal{B}\left(S^{\mathbb{N}_0}\right)^{\otimes N}\right)$.*

4.2 Definition of the Probability Space

In the following, we will only be interested in the distribution of the trajectory on $S^{\mathbb{N}_0}$ or $\left(S^N\right)^{\mathbb{N}_0} \cong \left(S^{\mathbb{N}_0}\right)^N$ respectively, and how they evolve with α and θ. Hence, we define the measurable space $(\Omega, \mathcal{A})$ as:

$$\Omega := H \times P^N \times \left(S^N\right)^{\mathbb{N}_0}, \quad \mathcal{A} := \mathcal{B}(\Omega)\,,$$

and endow it with the probability measure $\mathbb{P}$, given by:

$$\mathbb{P} := \left(\mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}}^{\otimes N}\right) \otimes \Psi\,.$$

Further, as before, we denote the coordinate projections by $\mathscr{X} := (\mathscr{H}, \mathscr{P}_{[N]}, \xi_{[N]})$, that is, $\mathscr{H} \sim \mathbb{P}_{\mathscr{H}}$, $\mathscr{P}_1, ..., \mathscr{P}_N \overset{iid}{\sim} \mathbb{P}_{\mathscr{P}}$, and $\xi_{[N]} := (\xi_1, ..., \xi_N) \sim (\mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}}^{\otimes N}) \cdot \Psi$.

Remark 23 *Since we define $\mathbb{P}$ through the probability kernel Ψ, it is assumed implicitly that Assumptions 9 and 11 do hold all the time, as they were needed for its construction.*

The following lemma summarizes results about regular versions of the conditional distributions arising from the construction above. Additionally, it is also meant to fix the notation.

Lemma 24 *It holds that:*

(i) Ψ is a regular version of the conditional distribution of $\xi_{[N]}$, given $\mathscr{H}$ and $\mathscr{P}_{[N]}$, that is:

$$\mathbb{P}_{\xi_{[N]}|\mathscr{H},\mathscr{P}_{[N]}}\{\mathsf{A}\} = \Psi(\mathscr{H}, \mathscr{P}_{[N]}, \mathsf{A}), \qquad \mathbb{P}_{(\mathscr{H},\mathscr{P}_{[N]})}\text{−a.s.}$$

(ii) $\mathbb{P}_{\mathscr{P}_{[N]}} \otimes \Psi$ *is a regular version of the conditional distribution of* $(\mathscr{P}_{[N]}, \xi_{[N]})$*, given* $\mathscr{H}$*, that is:*

$$\mathbb{P}_{(\mathscr{P}_{[N]},\xi_{[N]})|\mathscr{H}}\{\mathsf{A}\} = \left(\mathbb{P}_{\mathscr{P}_{[N]}} \otimes \Psi(\mathscr{H}, \cdot)\right)\{\mathsf{A}\}, \qquad \mathbb{P}_{\mathscr{H}}\text{−a.s.}$$

(iii) ψ *is a regular version of the conditional distribution of* ξ_n*, given* $\mathscr{H}$ *and* $\mathscr{P}_n$*, that is:*

$$\mathbb{P}_{\xi_n|\mathscr{H},\mathscr{P}_n}\{\mathsf{A}\} = \psi(\mathscr{H}, \mathscr{P}_n, \mathsf{A}), \qquad \mathbb{P}_{(\mathscr{H},\mathscr{P}_n)}\text{−a.s.}$$

(iv) $\mathbb{P}_{\mathscr{P}} \otimes \psi$ *is a regular version of the conditional distribution of* $(\mathscr{P}_n, \xi_n)$*, given* $\mathscr{H}$*, that is,* $\mathbb{P}_{\mathscr{H}}$−a.s.*:*

$$\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H}}\{\mathsf{A}\} = \left(\mathbb{P}_{\mathscr{P}_n} \otimes \psi(\mathscr{H}, \cdot)\right)\{\mathsf{A}\} = \left(\mathbb{P}_{\mathscr{P}} \otimes \psi(\mathscr{H}, \cdot)\right)\{\mathsf{A}\} =: \mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\{\mathsf{A}\}\,.$$

Proof Basically, these statements are a direct consequence of the definition of the probability space and the properties of ψ and Ψ. For more details, see Appendix C. ■

Remark 25 *(i) In the following, we will use solely these regular versions of the conditional distributions and omit the "*a.s.*".*

(ii) Other regular conditional distributions, for example, a regular version of the conditional distribution of $(\mathscr{P}_n, \xi_n)$*, given* $\mathscr{H}$ *and* $\mathscr{P}_n$*, that is,* $\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}$*, follow from the ones specified in Lemma 24 through sections:*

$$\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H}=\alpha,\mathscr{P}_n=\theta}\{\mathsf{A}\} := \mathbb{P}_{\xi_n|\mathscr{H}=\alpha,\mathscr{P}_n=\theta}\{\mathsf{A}_\theta\}\,,$$

where $\mathsf{A}_\theta := \{z \in S^{\mathbb{N}_0} \ : \ (\theta, z) \in \mathsf{A}\}$*.*

(iii) Since $\mathscr{X} = (\mathscr{H}, \mathscr{P}_{[N]}, \xi_{[N]})$ *is defined as the coordinate projections on* Ω*, we will use the expressions for the regular version of the conditional probability and conditional distribution interchangeably depending on which seems to be more easy to read:*

$$\begin{aligned}\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H}=\alpha,\mathscr{P}_n=\theta}\{\mathsf{A}\} &= \mathbb{P}\{(\mathscr{P}_n, \xi_n) \in \mathsf{A} \mid \mathscr{H} = \alpha, \mathscr{P}_n = \theta\} \\ &= \mathbb{P}\{(\mathscr{P}_n, \xi_n) \in \mathsf{A} \mid \mathscr{H}, \mathscr{P}_n\}(\omega)\,.\end{aligned}$$

Similarly for expected values.

Corollary 26 *Given any* $(\alpha, \theta_{[N]}) \in H \times P^N$*, the processes* $\xi_{[N]} = (\xi_1, ..., \xi_N)$ *are independent, and* ξ_i *is independent of* θ_j *for* $i \neq j$*. That is, for any set* $\mathsf{A}_1 \times ... \times \mathsf{A}_N \in \left(S^{\mathbb{N}_0}\right)^N$ *it holds that:*

$$\mathbb{P}\left\{\xi_{[N]} \in \mathsf{A}_1 \times ... \times \mathsf{A}_N \mid \mathscr{H} = \alpha, \mathscr{P}_{[N]} = \theta_{[N]}\right\} = \prod_{i=1}^{N} \mathbb{P}\left\{\xi_n \in \mathsf{A}_n \mid \mathscr{H} = \alpha, \mathscr{P}_n = \theta_n\right\}\,,$$

that is, $\mathbb{P}_{\xi_{[N]}|\mathscr{H},\mathscr{P}_{[N]}} = \bigotimes_{n=1}^{N} \mathbb{P}_{\xi_n|\mathscr{H},\mathscr{P}_n}$*.*

Proof This follows directly from Lemma 21 and Lemma 24. ■

Corollary 27 *Let $f_1, ..., f_N : S^{\mathbb{N}_0} \to \mathbb{R}_{\geq 0}$ be measurable functions. Then it holds that:*

$$\mathbb{E}\left\{\sum_{n=1}^{N} f_n(\xi_n) \mid \mathscr{H}, \mathscr{P}_{[N]}\right\} = \sum_{n=1}^{N} \mathbb{E}\left\{f_n(\xi_n) \mid \mathscr{H}, \mathscr{P}_n\right\} .$$

Proof By Lemma 24, it holds that:

$$\begin{aligned}\mathbb{E}\left\{\sum_{n=1}^{N} f_n(\xi_n) \mid \mathscr{H}, \mathscr{P}_{[N]}\right\} &= \int_{(S^{\mathbb{N}_0})^N} \sum_{n=1}^{N} f_n(z_n)\ \Psi(\mathscr{H}, \mathscr{P}_{[N]}, dz_{[N]}) \\ &= \sum_{n=1}^{N} \int_{(S^{\mathbb{N}_0})^N} f_n(z_n)\ \Psi(\mathscr{H}, \mathscr{P}_{[N]}, dz_{[N]}) .\end{aligned}$$

By Lemma 21 and Fubini's theorem, this is the same as:

$$\begin{aligned}&= \sum_{n=1}^{N} \int_{S^{\mathbb{N}_0}} f_n(z_n)\ \psi(\mathscr{H}, \mathscr{P}_n, dz_n) \cdot \prod_{i \neq n}^{N} \underbrace{\int_{S^{\mathbb{N}_0}} \psi(\mathscr{H}, \mathscr{P}_i, dz_i)}_{=1} \\ &= \sum_{n=1}^{N} \int_{S^{\mathbb{N}_0}} f_n(z_n)\ \psi(\mathscr{H}, \mathscr{P}_n, dz_n) = \sum_{n=1}^{N} \mathbb{E}\left\{f_n(\xi_n) \mid \mathscr{H}, \mathscr{P}_n\right\} .\end{aligned}$$

where the last step follows from applying Lemma 24 again. ■

4.3 Stopping the Algorithm

Ultimately, the algorithm is stopped at some point. Typically, this is the case as soon as some convergence criterion is met. Here, the set of points $s \in S$ where $\mathcal{A}$ satisfies the convergence criterion can be represented as a subset $\tilde{\mathsf{C}} \subset S$. However, since we are considering parametric loss functions, we also have to use a parametric set $\mathsf{C} \subset P \times S$:

Definition 28 *The* convergence set $\mathsf{C} \subset P \times S$ *is defined as:*

$$\mathsf{C} := \{(\theta, s) \in P \times S \ : \ s \text{ satisfies the convergence criterion for } \ell(\cdot, \theta).\}$$

Since the convergence set is defined in terms of a not further specified convergence criterion, we need the following assumption:

Assumption 29 *The convergence set C is measurable.*

Example 30 *(i) One could use $\mathsf{C} := \{(\theta, s) \in P \times S \ : \ \ell(s, \theta) \leq \varepsilon\}$ for convergence in terms of the loss function, which is measurable, since ℓ is measurable.*

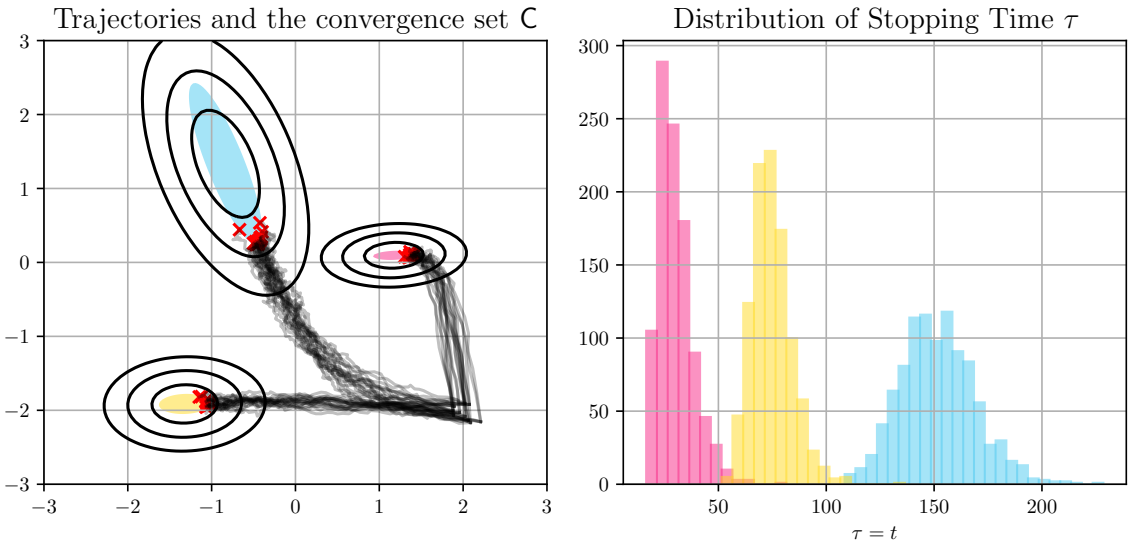

Figure 4: Visualization of the stopping time τ: In the left plot, the convergence set for each problem is shown as shaded region. As soon as a trajectory enters this region (red crosses), the algorithm is stopped. This yields the distribution of τ depending on the parameter θ, as shown in the right plot.

(ii) *If $\ell(s,\theta)$ has a unique minimizer s^*_θ, convergence in terms of the iterates could be written as $\mathsf{C} := \{(\theta, s) \in P \times S \ : \ d(s, s^*_\theta) \leq \varepsilon\}$. If $\theta \mapsto s^*_\theta$ is measurable, C is measurable, because the distance d is continuous.*

(iii) *One could use $\mathsf{C} := \{(\theta, s) \in P \times S \ : \ \|\nabla_s \ell(s,\theta)\| \leq \varepsilon\}$ for convergence to a stationary point. For example, if $\nabla_s \ell$ is (jointly) continuous, C is measurable.*

Usually, the algorithm $\mathcal{A}$ is stopped either if it is converged, that is, $(\mathscr{P}, \xi^{(t)}) \in \mathsf{C}$ for some $t \in \mathbb{N}_0$, or, if the maximal computational budget is reached, that is, for example, a certain number of iterations has been performed. Hence, we introduce the following random times $\tau_{\max} := t_{\max} \in \mathbb{N}$ and $\tau_{\text{conv,n}} := \inf\{t \in \mathbb{N}_0 \ : \ (\mathscr{P}_n, \xi_n^{(t)}) \in \mathsf{C}\}$, $n = 1, ..., N$. Then, we combine them into the random times τ_n:

$$\tau_n := \tau_{\max} \wedge \tau_{\text{conv,n}} := \min\{\tau_{\max}, \tau_{\text{conv,n}}\}, \quad n = 1, ..., N\,.$$

The following lemma shows that the random times τ_n are indeed *stopping times* (sometimes also called *optional times*) in the sense of probability theory, that is, $\{\tau_n \leq t\} \in \mathcal{F}_n^{(t)}$ for each $t \in \mathbb{N}_0$, $n = 1, ..., N$. Intuitively, this tells us that, at time t, the information collected in $\mathcal{F}_n^{(t)}$ is indeed enough to decide whether the algorithm did reach the convergence set:

Proposition 31 *Suppose that Assumption 29 holds. Then, for each $n \in \{1, ..., N\}$, τ_n is a stopping time w.r.t. the filtration $\mathcal{F}_n = (\mathcal{F}_n^{(t)})_{t \in \mathbb{N}_0}$, where $\mathcal{F}_n^{(t)} := \sigma(\mathscr{P}_n, \xi_n^{(0)}, ..., \xi_n^{(t)})$.*

Proof If $\tau_{\max}$ and $\tau_{\text{conv,n}}$ are $\mathcal{F}_n$-optional, so is τ_n by Kallenberg (2021, Lemma 9.1 (i), p.186). Since $\tau_{\max}$ is constant, it is $\mathcal{F}_n$-optional by Kallenberg (2021, Lemma 9.1 (v), p.186). Hence, we only have to show that $\tau_{\text{conv,n}}$ is $\mathcal{F}_n$-optional. By definition, it holds that $\tau_{\text{conv,n}}$ is a so-called *hitting time* for the process $\mathscr{Y}_n = (\mathscr{Y}_n^{(t)})_{t\in\mathbb{N}_0}$ given by $\mathscr{Y}_n^{(t)} := (\mathscr{P}_n, \xi_n^{(t)})$, that is, it is of the form

$$\tau_{\text{conv,n}} = \inf\left\{ t \in \mathbb{N}_0 \ : \ \mathscr{Y}_n^{(t)} \in \mathsf{C} \right\} .$$

Then, by Kallenberg (2021, Lemma 9.6 (i)), $\tau_{\text{conv,n}}$ is weakly $\tilde{\mathcal{F}}$-optional for every filtration $\tilde{\mathcal{F}}$ such that $\mathscr{Y}_n$ is adapted to $\tilde{\mathcal{F}}$. In particular, this holds for $\mathcal{F}_n$, because:

$$\mathcal{F}_n^{(t)} = \sigma(\mathscr{P}_n, \xi_n^{(0)}, ..., \xi_n^{(t)}) = \sigma((\mathscr{P}_n, \xi_n^{(0)}), ..., (\mathscr{P}_n, \xi_n^{(t)})) = \sigma(\mathscr{Y}_n^{(0)}, ..., \mathscr{Y}_n^{(t)}) .$$

Since in discrete time there is no distinction between weakly optional times and optional times, the conclusion follows. ■

5 Generalization Results

In this section, we use the previously constructed probability space to derive the generalization results. To this end, in Theorem 33 we give a PAC-Bayesian generalization result for the case of a bounded parametric function defined on the space of trajectories $S^{\mathbb{N}_0}$. Then, in Corollary 35 we specialize it to the case of the convergence time, and in Corollary 38 to the case of the convergence rate. Both results are immediate consequences of Theorem 33.

Lemma 32 *Let $f : P \times S^{\mathbb{N}_0} \to [0,\infty)$ be a measurable function that is bounded from above by $f_{\max} \in \mathbb{R}$. Then, for every $\lambda \in \mathbb{R}$ it holds that:*

$$\mathbb{E}\left\{ \exp\left(\lambda \left(\frac{1}{N}\sum_{n=1}^{N} \mathbb{E}\left\{ f(\mathscr{P}_n, \xi_n) \mid \mathscr{H}, \mathscr{P}_n \right\} - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{ f \right\} \right) - \frac{\lambda^2}{2N} f_{\max}^2 \right) \right\} \leq 1 .$$

Proof This follows from standard arguments like Jensen's inequality, Fubini's theorem, Hoeffding's inequality, and the i.i.d. assumption. A detailed proof is given in Appendix D. ■

For the following result, please recall that $\alpha \mapsto \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}=\alpha}\{f\} = (\mathbb{P}_{\mathscr{P}} \otimes \psi(\alpha,\cdot))[f]$ is a measurable function on H, that is, it can be integrated w.r.t. an arbitrary (probability) measure $\rho \in \mathcal{M}(H)$. Furthermore, note that, by projecting from $S^{\mathbb{N}_0}$ onto the corresponding coordinate, this result can be applied to single, fixed iterates, too.

Theorem 33 *Let $f : P \times S^{\mathbb{N}_0} \to [0,\infty)$ be a measurable function that is bounded from above by $f_{\max} \in \mathbb{R}$. Then, for every $\lambda \in (0,\infty)$ and $\varepsilon > 0$ it holds that:*

$$\mathbb{P}_{\mathscr{P}_{[N]}}\Big\{ \forall \rho \in \mathcal{P}(\mathbb{P}_{\mathscr{H}}) \ : \ \rho\left[\mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{f\right\}\right] \leq \frac{1}{N}\sum_{n=1}^{N} \rho\left[\mathbb{E}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}\left\{f\right\}\right] + \frac{D_{\text{KL}}(\rho \parallel \mathbb{P}_{\mathscr{H}}) + \frac{\lambda^2}{2N} f_{\max}^2 - \log(\varepsilon)}{\lambda} \Big\} \geq 1 - \varepsilon .$$

Proof Abbreviate $\bar{f} := \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}$ and $\hat{f} := \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}\{f\}$. Then, since Lemma 32 holds for any $\lambda \in \mathbb{R}$, we get that:

$$\mathbb{E}_{(\mathscr{P}_{[N]},\mathscr{H})}\left\{\exp\left(\lambda\left(\bar{f}-\hat{f}\right)-\frac{\lambda^2}{2N}f_{\max}^2\right)\right\} \leq 1\,.$$

By Fubini's theorem applied to $\mathbb{P}_{(\mathscr{P}_{[N]},\mathscr{H})} = \mathbb{P}_{\mathscr{P}_{[N]}} \otimes \mathbb{P}_{\mathscr{H}}$, the variational formulation of Donsker-Varadhan, and the linearity of the integral, this is the same as:

$$\mathbb{E}_{\mathscr{P}_{[N]}}\left\{\exp\left(\sup_{\rho\in\mathcal{P}(\mathbb{P}_{\mathscr{H}})}\lambda\left(\rho[\bar{f}]-\rho[\hat{f}]\right)-D_{\mathrm{KL}}(\rho\parallel\mathbb{P}_{\mathscr{H}})-\frac{\lambda^2}{2N}f_{\max}^2\right)\right\} \leq 1\,.$$

Then, for any $s \in \mathbb{R}$ we get from Markov's inequality:

$$\mathbb{P}_{\mathscr{P}_{[N]}}\left\{\sup_{\rho\in\mathcal{P}(\mathbb{P}_{\mathscr{H}})}\lambda\left(\rho[\bar{f}]-\rho[\hat{f}]\right)-D_{\mathrm{KL}}(\rho\parallel\mathbb{P}_{\mathscr{H}})-\frac{\lambda^2}{2N}f_{\max}^2 \geq s\right\} \leq \exp(-s)\,.$$

Using $s = \log(1/\varepsilon)$ yields:

$$\mathbb{P}_{\mathscr{P}_{[N]}}\left\{\sup_{\rho\in\mathcal{P}(\mathbb{P}_{\mathscr{H}})}\lambda\left(\rho[\bar{f}]-\rho[\hat{f}]\right)-D_{\mathrm{KL}}(\rho\parallel\mathbb{P}_{\mathscr{H}})-\frac{\lambda^2}{2N}f_{\max}^2 \geq \log\left(\frac{1}{\varepsilon}\right)\right\} \leq \varepsilon\,.$$

Restricting to $\lambda > 0$, reformulating, and taking the complementary event yields the result. ■

Remark 34 *Under some mild assumptions, by a covering argument, the provided bound could also be made uniform in λ.*

5.1 Guarantees for the Convergence Time

Please recall the definition of $\tau_n = \tau_{\mathrm{conv,n}} \wedge \tau_{\max}$. Then, note that $\tau_{\mathrm{conv,n}}$ can be written as $\tilde{\tau} \circ (\mathscr{P}_n, \xi_n)$, where $\tilde{\tau} : P \times S^{\mathbb{N}_0} \to \mathbb{N}_0 \cup \{+\infty\}$ is given by:

$$(\theta, (z^{(t)})_{t\in\mathbb{N}_0}) \mapsto \inf\{k \in \mathbb{N}_0 \;:\; (\theta, z^{(k)}) \in \mathsf{C}\} = \inf_{k\in\mathbb{N}_0} k\cdot(1+\iota_{\mathsf{C}}(\theta, z^{(k)}))\,,$$

where $\iota_{\mathsf{C}}(\theta, z^{(k)}) = 0$, if $(\theta, z^{(k)}) \in \mathsf{C}$, and $+\infty$ otherwise. Since C is measurable by Assumption 29, and ι_{C} only takes the values $\{0, +\infty\}$, the map $(\theta, (z^{(t)})_{t\in\mathbb{N}}) \mapsto k\cdot(1+\iota_{\mathsf{C}}(\theta, z^{(k)}))$ is measurable. Thus, $\tilde{\tau}$ is measurable as infimum of countably many measurable functions. Therefore, the map $T : P \times S^{\mathbb{N}_0} \to \mathbb{N}_0$, defined through

$$T(\theta, (z^{(t)})_{t\in\mathbb{N}_0}) := t_{\max} \wedge \tilde{\tau}(\theta, (z^{(t)})_{t\in\mathbb{N}_0})\,,$$

is measurable and bounded by $t_{\max}$, and we can write τ_n as $T \circ (\mathscr{P}_n, \xi_n)$. Hence, defining the average expected convergence time as $\bar{\tau} := \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{T\}$, we get the following result:

Corollary 35 *Suppose that Assumption 29 holds. Then, for every $\lambda \in (0,\infty)$ and $\varepsilon > 0$ it holds that:*

$$\mathbb{P}_{\mathscr{P}_{[N]}}\Big\{\forall\rho\in\mathcal{P}\left(\mathbb{P}_{\mathscr{H}}\right)\;:\;\rho\left[\bar{\tau}\right] \leq \frac{1}{N}\sum_{n=1}^{N}\rho\left[\mathbb{E}\left\{\tau_n \mid \mathscr{H}, \mathscr{P}_n\right\}\right] + \frac{D_{\mathrm{KL}}(\rho\parallel\mathbb{P}_{\mathscr{H}}) + \frac{\lambda^2}{2N}t_{\max}^2 - \log(\varepsilon)}{\lambda}\Big\} \geq 1-\varepsilon\,.$$

Proof Since T is bounded by $t_{\max}$, one can apply Theorem 33 with $f = T$ and $f_{\max} = t_{\max}$, which, by noting that $\mathbb{E}\{\tau_n \mid \mathscr{H}, \mathscr{P}_n\} = \mathbb{E}_{(\mathscr{P}_n, \xi_n)|\mathscr{H}, \mathscr{P}_n}\{T\}$, directly yields the result. ■

5.2 Guarantees for the Convergence Rate

The convergence rate of an algorithm is determined by how it contracts a certain criterion along the iterates, for example, the loss function (convergence rate in terms of function values) or the distance to the set of minimizers (convergence in terms of the iterates). For this, we need a corresponding function on the space of sequences:

Definition 36 *The* contraction function *is defined as:*

$$c : P \times S \times S \to \mathbb{R}_{\geq 0},\ (\theta, x, y) \mapsto \frac{\ell(x, \theta)}{\ell(y, \theta)} \cdot \mathbb{1}\{\ell(y, \theta) > 0\}\,.$$

Abbreviate $T := T(\theta, (z^{(t)})_{t \in \mathbb{N}})$*. Then, the* rate function *is defined as:*

$$r : P \times S^{\mathbb{N}_0} \to \mathbb{R}_{\geq 0},\ (\theta, (z^{(t)})_{t \in \mathbb{N}_0}) \mapsto (c(\theta, z^{(T)}, z^{(0)}))^{\frac{1}{T}} \cdot \mathbb{1}\{T \geq 1\}\,,$$

and the expected rate function *is defined as* $\bar{r} := \mathbb{E}_{(\mathscr{P}, \xi)|\mathscr{H}}\{r\}$*. Similarly, for some* $r_{\max} \in \mathbb{R}_{\geq 0}$*, we define the* bounded rate function *and* expected bounded rate function *as* $r_b := r \cdot \mathbb{1}\{r \leq r_{\max}\}$*, and* $\bar{r}_b = \mathbb{E}_{(\mathscr{P}, \xi)|\mathscr{H}}\{r_b\}$*.*

Remark 37 *Note that, in contrast to defining the rate function in terms of the maximum over the iterations, this definition is applicable in the stochastic case, too.*

Since ℓ is measurable, we have that c is measurable. Further, since T is measurable (if C is) and only takes countably many values, we get that $z^{(T)}$ is measurable. Therefore, also r is measurable. Hence, r_b is measurable and bounded, and we get the following result:

Corollary 38 *Suppose that Assumption 29 holds. Then, for every* $\lambda \in (0, \infty)$ *and* $\varepsilon > 0$ *it holds that:*

$$\mathbb{P}_{\mathscr{P}_{[N]}}\Big\{\forall \rho \in \mathcal{P}(\mathbb{P}_{\mathscr{H}})\ :\ \rho[\bar{r}_b] \leq \frac{1}{N}\sum_{n=1}^{N} \rho\left[\mathbb{E}_{(\mathscr{P}_n, \xi_n)|\mathscr{H}, \mathscr{P}_n}\{r_b\}\right] + \frac{D_{\mathrm{KL}}(\rho \parallel \mathbb{P}_{\mathscr{H}}) + \frac{\lambda^2}{2N} r_{\max}^2 - \log(\varepsilon)}{\lambda}\Big\} \geq 1 - \varepsilon\,.$$

Proof Again, this follows directly from Theorem 33 with $f = r_b$ and $f_{\max} = r_{\max}$. ■

5.3 Properties of the Trajectory

The last theoretical result concerns the probability to observe a trajectory that obeys a certain property, for example, to converge with at least a rate of $r \leq r_{\max}$. Such properties can be encoded in a measurable set $\mathsf{A} \subset P \times S^{\mathbb{N}_0}$, and we have to consider the function $f := \mathbb{1}_{\mathsf{A}}$. Since $\mathbb{1}_{\mathsf{A}}$ is measurable and bounded, one could directly apply the results from above. Yet, in this case, one can compute the integral in closed-form and get tighter results:

Lemma 39 *Let $\mathsf{A} \subset P \times S^{\mathbb{N}_0}$ be measurable. Then, for any $\lambda \in \mathbb{R}$, it holds that:*

$$\mathbb{E}\left\{\exp\left(-\frac{\lambda}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\right)\right\} = \mathbb{E}\left\{\left(1-\left[1-\exp\left(-\frac{\lambda}{N}\right)\right]\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right)^N\right\}.$$

Proof By the same arguments as before, we get from Lemma 24:

$$\mathbb{E}\left\{\exp\left(-\frac{\lambda}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\right)\right\} = \mathbb{E}\left\{\left(\mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\exp\left(-\frac{\lambda}{N}\mathbb{1}_{\mathsf{A}}\right)\right\}\right)^N\right\}.$$

Then, for any $\alpha \in H$, the inner integral is given by:

$$\begin{aligned}\mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}=\alpha}\left\{\exp\left(-\frac{\lambda}{N}\mathbb{1}_{\mathsf{A}}\right)\right\} &= \mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}=\alpha}\left\{\mathsf{A}^c\right\} + \exp\left(-\frac{\lambda}{N}\right)\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}=\alpha}\left\{\mathsf{A}\right\}\\ &= 1-\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}=\alpha}\left\{\mathsf{A}\right\} + \exp\left(-\frac{\lambda}{N}\right)\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}=\alpha}\left\{\mathsf{A}\right\}\\ &= 1-\left[1-\exp\left(-\frac{\lambda}{N}\right)\right]\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}=\alpha}\left\{\mathsf{A}\right\},\end{aligned}$$

which concludes the proof. ■

The term inside the power is the Laplace transform of a Bernoulli random variable with parameter $\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\{\mathsf{A}\}$. Generalization bounds for Bernoulli-random variables were already presented, for example, by Catoni (2007), whose approach will be applied in the following.

Lemma 40 (Catoni (2007)) *Define the function*

$$\Phi_a(p) := -\frac{1}{a}\log\left(1-\left[1-\exp(-a)\right]p\right).$$

Then it holds that:

(i) *Φ_a is an increasing one-to-one mapping of the unit-interval onto itself,*

(ii) *Φ_a is strictly convex for $a > 0$ and strictly concave for $a < 0$,*

(iii) *Φ_a^{-1} is given by: $\Phi_a^{-1}(q) := \frac{1-\exp(-aq)}{1-\exp(-a)}$.*

Proof This follows by standard arguments. The details are provided in Appendix E. ■

This yields the following corollary, which is needed to apply the PAC-Bayesian argument:

Corollary 41 *Let $\mathsf{A} \subset P \times S^{\mathbb{N}_0}$ be measurable. Then, for any $\lambda \in \mathbb{R}$ it holds that:*

$$\mathbb{E}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\right]\right)\right\} = 1.$$

In particular, it holds that:

$$\mathbb{E}_{\mathscr{P}_{[N]}}\left\{\mathbb{E}_{\mathscr{H}}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}\left\{\mathsf{A}\right\}\right]\right)\right\}\right\} \leq 1.$$

Proof The first statement follows directly by combining Lemma 39 and Lemma 40. The second statement then follows from Fubini's theorem and Jensen's inequality. More details are given in Appendix F. ■

As before, this yields the following PAC-Bayesian generalization bound. For completeness, we provide a proof in Appendix G.

Theorem 42 (Catoni (2007)) *Let $\mathsf{A} \subset P \times S^{\mathbb{N}_0}$ be measurable. Then, for $\lambda \in (0, \infty)$, it holds that:*

$$\mathbb{P}_{\mathscr{P}_{[N]}}\Big\{\forall \rho \in \mathcal{P}(\mathbb{P}_{\mathscr{H}}) \ : \ \rho[\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\{\mathsf{A}\}] \leq \Phi^{-1}_{\frac{\lambda}{N}}\left(\frac{1}{N}\sum_{n=1}^{N}\rho\left[\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}\{\mathsf{A}\}\right] + \frac{D_{KL}(\rho \parallel \mathbb{P}_{\mathscr{H}}) + \log\left(\frac{1}{\varepsilon}\right)}{\lambda}\right)\Big\} \geq 1-\varepsilon\,.$$

Remark 43 *By a union bound with confidence level $1-\frac{\varepsilon}{3}$, Corollary 35, Corollary 38 and Theorem 42 can be applied at once, because:*

$$\begin{aligned}\mathbb{P}_{\mathscr{P}_{[N]}}\{\mathsf{A}\cap\mathsf{B}\cap\mathsf{C}\} &= 1-\mathbb{P}_{\mathscr{P}_{[N]}}\{\mathsf{A}^c\cup\mathsf{B}^c\cup\mathsf{C}^c\}\\ &\geq 1-\left(\mathbb{P}_{\mathscr{P}_{[N]}}\{\mathsf{A}^c\}+\mathbb{P}_{\mathscr{P}_{[N]}}\{\mathsf{B}^c\}+\mathbb{P}_{\mathscr{P}_{[N]}}\{\mathsf{C}^c\}\right) \geq 1-\varepsilon\,.\end{aligned}$$

This tells us that, with confidence level $1-\varepsilon$, we can estimate the probability to converge with a rate of at least r_+, an upper bound on the actual convergence rate, and the expected convergence time all at once. *Generally, however, without adjusting the confidence level ε, this is false, and one cannot expect that all three bounds do hold at once.*

6 Numerical Results

We consider the following five experiments: a strongly convex and smooth quadratic problem, a convex and smooth image processing problem, the convex and non-smooth LASSO problem, the non-convex and non-smooth problem of training a neural network, and a non-convex and non-smooth stochastic empirical risk minimization problem. The training procedure is mostly the same as the one by Sucker et al. (2024). The main difference here is that we replace $\ell_{\text{train}}(\alpha,\theta,\xi^{(0)},s) = \sum_{i=1}^{s}\mathbb{1}\{\ell(\xi^{(i-1)},\theta)>0\}\frac{\ell(\xi^{(i)},\theta)}{\ell(\xi^{(i-1)},\theta)}$ by

$$\ell_{\text{train}}(\alpha,\theta,\xi^{(t)}) = \mathbb{1}\{\ell(\xi^{(t)},\theta)>0\}\frac{\ell(\xi^{(t+1)},\theta)}{\ell(\xi^{(t)},\theta)}\cdot\mathbb{1}_{\mathsf{C}^c}(\xi^{(t)},\theta)\,,$$

where $\mathsf{C}\subset P\times S$ is the convergence set. Thus, we fix $s=1$ and the algorithm "observes" a loss only as long as it did not reach the convergence set. This effectively solves the problem mentioned by Sucker et al. (2024) that the algorithm might observe a "full loss" in the case of convergence.

For completeness, we briefly summarize the training procedure: In the outer loop, we sample a loss-function randomly from the training set. Then, in the inner loop, we train the algorithm on this loss-function with ℓ_{train}, that is, in each iteration the algorithm computes

a new point and observes the loss ℓ_{train}, which is used to update its hyperparameters. This finally yields some hyperparameters α_0. Then, starting from α_0, we construct the *discrete* prior distribution $\mathbb{P}_{\mathscr{H}}$ over points $\alpha_1, ..., \alpha_{n_{\text{sample}}} \in H$, by a sampling procedure. Finally, we perform the (closed-form) PAC-Bayesian optimization step, which yields the posterior $\rho^* \in \mathcal{P}(\mathbb{P}_{\mathscr{H}})$. In the end, for simplicity, we set the hyperparameters to

$$\alpha^* = \underset{i=1,...,n_{\text{sample}}}{\arg\max} \ \rho^*\{\alpha_i\}\,.$$

In the description below, we use $x^{(t)}$, $t \in \mathbb{N}_0$, to denote the iterates of the algorithm in the optimization space, that is, $x^{(t)} = \Pi_S(\xi^{(t)})$, and, typically, we have $x^{(t)} \in \mathbb{R}^d$, $d \in \mathbb{N}$.

6.1 Quadratics

In this subsection, we consider strongly convex quadratic functions with varying strong convexity, varying smoothness and varying right-hand side, that is, each optimization problem is of the form:

$$\min_{x \in \mathbb{R}^d} \frac{1}{2}\|Ax - b\|^2\,, \qquad A \in \mathbb{R}^{d\times d},\ b \in \mathbb{R}^d\,.$$

Thus, the parameters are given by $\theta = (A, b) \in \mathbb{R}^{d^2+d} =: P$, while the optimization variable is $x \in \mathbb{R}^d$, and we use $d = 200$. We control the strong-convexity and smoothness of ℓ by sampling them randomly in the intervals $[m_-, m_+], [L_-, L_+] \subset (0, +\infty)$, and define the matrix A_j, $j = 1, ..., N$, as a *diagonal matrix* with entries $a^j_{ii} = \sqrt{m_j} + i \cdot \frac{\sqrt{L_j} - \sqrt{m_j}}{d}$, $i = 1, ..., d$. While, in principle, this is restrictive, we do not use this knowledge explicitly, and, later on, achieve a similar performance in the image-processing and LASSO problems, which both include a non-diagonal quadratic term. Finally, we define the convergence set as

$$\mathsf{C}_{\text{quad}} := \{(\theta, s) \in P \times S \ :\ \ell(s, \theta) < 10^{-8} \text{ or } \|\nabla \ell(s, \theta)\| < 10^{-6}\}\,.$$

Since the given class of functions is L_+-smooth and m_--strongly convex, we use *heavy-ball with friction* (HBF) (Polyak, 1964) as baseline. Its update is given by $x^{(t+1)} = x^{(t)} - \beta_1 \nabla f(x^{(t)}) + \beta_2 \left(x^{(t)} - x^{(t-1)}\right)$, where the optimal worst-case convergence rate is attained for $\beta_1 = \left(\frac{2}{\sqrt{L_+} + \sqrt{\mu_-}}\right)^2, \beta_2 = \left(\frac{\sqrt{L_+} - \sqrt{\mu_-}}{\sqrt{L_+} + \sqrt{\mu_-}}\right)^2$ (Nesterov, 2018). On the other hand, the learned algorithm $\mathcal{A}$ performs an update of the form $x^{(t+1)} = x^{(t)} + \beta^{(t)} \cdot d^{(t)}$, where $\beta^{(t)}$ and $d^{(t)}$ are predicted by separate blocks of a neural network. For more details on the architecture we refer to the Appendix H. The upper plot of Figure 5 shows that the learned algorithm outperforms HBF by orders of magnitude. The median is shown as dotted line, while the mean is shown in dashed line. The shaded region indicates the area up to the quantile $q = 0.95$, that is, 95% of the test data. We can observe that the mean is not representative for the typical performance of the algorithm, and is strongly influenced by a few problem instances for which the learned algorithm does not work as good. In the lower right plot, the convergence time is shown. We can see that most of the problems reach the stopping criterion before $t_{\max} = 500$ iterations, and, on average, the learned algorithms needs less than 300 iterations to solve the problem. Further, the provided PAC-bound yields a reasonable estimate of the true (average) convergence time. Similarly, the lower left plot

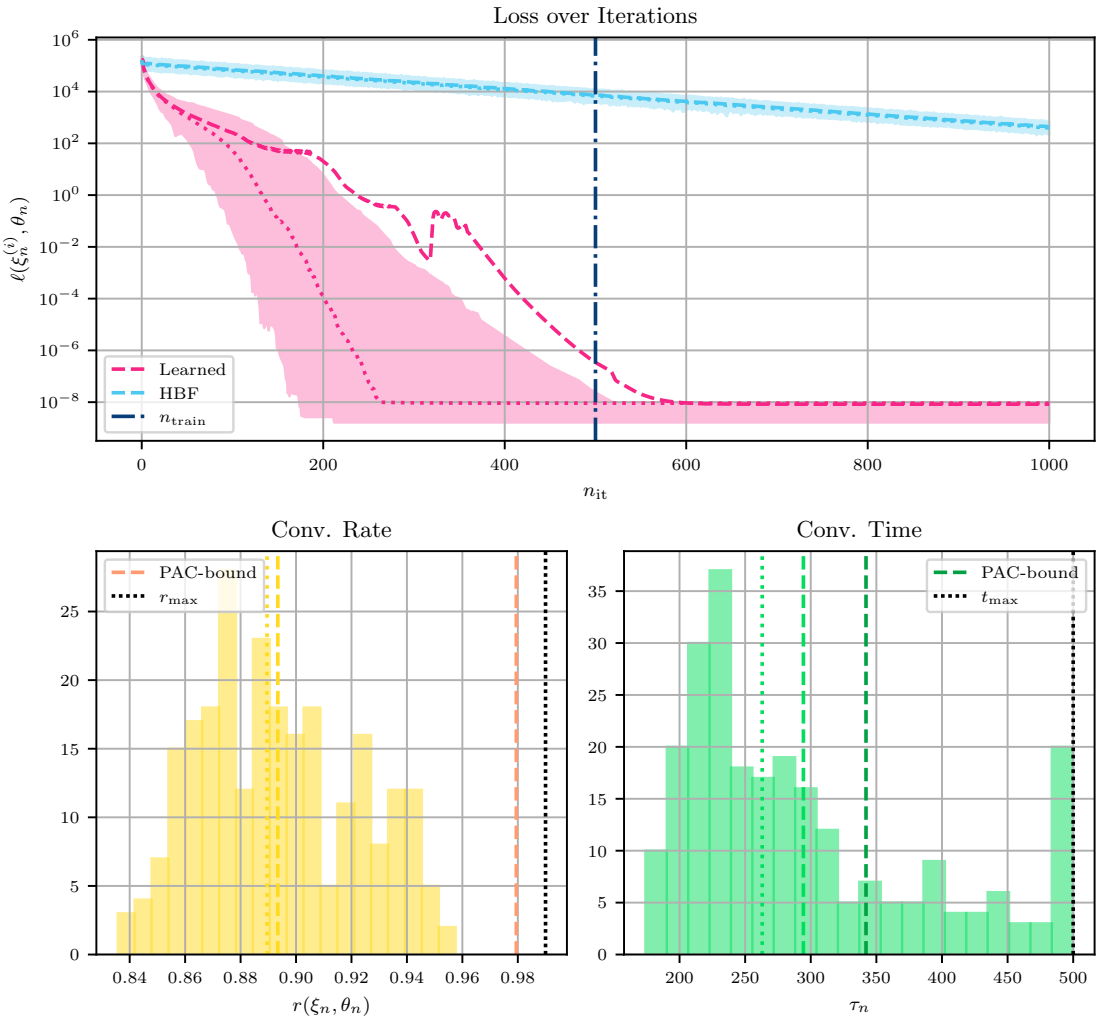

Figure 5: Quadratic: The **top figure** shows the loss over the iterations, where HBF is shown in blue and the learned algorithm in pink. The mean and median are shown as dashed and dotted lines, respectively, while the shaded region represents the test data up to the quantile $q = 0.95$, that is, 95% of the test data. We can see from the figure that the learned algorithm reaches the convergence criterion way faster than HBF. The **lower left plot** shows the convergence rate of the learned algorithm. Here, the dashed line represent the empirical mean and the PAC-bound, respectively, and we can see that the bound is not vacuous, but also not really tight. Similarly, the **lower right figure** shows the convergence time of the learned algorithm. Again, the dashed lines represent the empirical mean and the corresponding PAC-bound, which, in this case, is reasonably tight.

shows the estimated convergence rate: On average, the learned algorithm contracts the loss by a factor of less than 0.9 per iteration. However, while the given PAC-bound is not vacuous, it is also not tight.

6.2 Image Processing

In this subsection, we consider a (gray-scale) *image denoising/deblurring* problem with a smooth approximation to the L_1-norm of the image derivative as regularizer, that is, problems of the form:

$$\min_{x\in\mathbb{R}^d} \frac{1}{2}\|Ax-b\|^2 + \lambda \sum_{i,j=1}^{d} \sqrt{(D_h x)_{i,j}^2 + (D_w x)_{i,j}^2 + \varepsilon^2} \qquad \lambda \in \mathbb{R},\ A, D_h, D_w \in \mathbb{R}^{d\times d},\ b \in \mathbb{R}^d\,.$$

The matrix A describes the "blurring" of the image, while D_h and D_w are the discrete image derivatives in h- and w-direction, respectively, which are used to penalize local changes in the image. We use images of height $N_h = 200$ and width $N_w = \lfloor 0.75 \cdot N_h \rfloor = 150$. Thus, the dimension d of the optimization space is given by $d = 30000$. Further, as parameters θ we use the observed image and the regularization parameter, that is, $\theta = (b, \lambda) \in \mathbb{R}^{d+1} =: P$. Throughout, we use $\varepsilon = 0.01$. For computational efficiency, the matrices A, D_h, D_w are implemented through the convolution of the image x with a corresponding kernel (with reflective boundary conditions). Additionally, after blurring an image with A, we add centered Gaussian noise $\varepsilon_{i,j}$ with standard deviation $\sigma = \frac{25}{256}$ to each pixel. The regularization parameters $\lambda_i \in \mathbb{R}$, $i = 1, ..., N$, are sampled uniformly from the interval $[0.05, 0.5]$. Finally, we define the convergence set as

$$\mathsf{C}_{\text{img}} = \{(\theta, s) \in P \times S \ :\ \|\nabla \ell(s, \theta)\| < 10^{-4}\}\,.$$

Since the problem is smooth and convex, yet not strongly convex, the baseline algorithm is given by the *accelerated gradient descent* algorithm due to Nesterov (1983). Its update is given by first computing $y^{(t+1)} = x^{(t)} + \frac{\beta_1^{(t)}-1}{\beta_1^{(t+1)}}(x^{(t)} - x^{(t-1)})$ followed by setting $x^{(t+1)} = y^{(t)} - \beta_2 \nabla f(y^{(t+1)})$. We use the optimal choices $\beta_1^{(t+1)} = \frac{1}{2}\left(1 + \sqrt{1 + 4(\beta_1^{(t)})^2}\right)$ and $\beta_2 = \frac{1}{L}$. Here, the smoothness constant L is given by the largest eigenvalue of $A^TA + \frac{\lambda}{\varepsilon} D^T D$, where $D \in \mathbb{R}^{2d\times d}$ is given by "stacking" D_h and D_w, that is, $D = \begin{pmatrix} D_h & D_w \end{pmatrix}^T$. On the other hand, the learned algorithm $\mathcal{A}$ performs an update of the form $x^{(t+1)} = x^{(t)} + \frac{\|\nabla_x \ell(x^{(t)},\theta)\|}{L} d_1^{(t)} + \|x^{(t)} - x^{(t-1)}\| d_2^{(t)} - \frac{1}{L}\nabla_x \ell(x^{(t)}, \theta)$, where the directions $d_1^{(t)}$ and $d_2^{(t)}$ are predicted by a neural network. For more details on the architecture we refer to Appendix I. The results of this experiment are summarized in Figure 6. We can see that, on a typical problem from this distribution, the algorithm clearly outperforms NAG, and reaches the convergence set in about 500 iterations. However, as the mean (dashed line) strongly deviates from the remaining 95% of the test problems (shaded area), we observe that the learned algorithm does not converge for all of the problems. This is also validated by the lower right plot, which shows the convergence time: For about 10 out of 250 problems (4%), the algorithm does not reach the convergence set before $t_{\max} = 3000$. Furthermore, the lower left plot shows, unfortunately, that the PAC-bound for the convergence rate is vacuous here. This might be due to the fact that the rate function already yields a value close to $r_{\max}$. Thus, the PAC-bound, which has to be greater or equal, will exceed $r_{\max}$ easily. Therefore, we actually expect this behavior to happen whenever the actual rate is close to $r_{\max}$. Nevertheless, with more training data the bound would get more tight, and, additionally, we still have the guarantee for the convergence time, which is actually quite tight here.

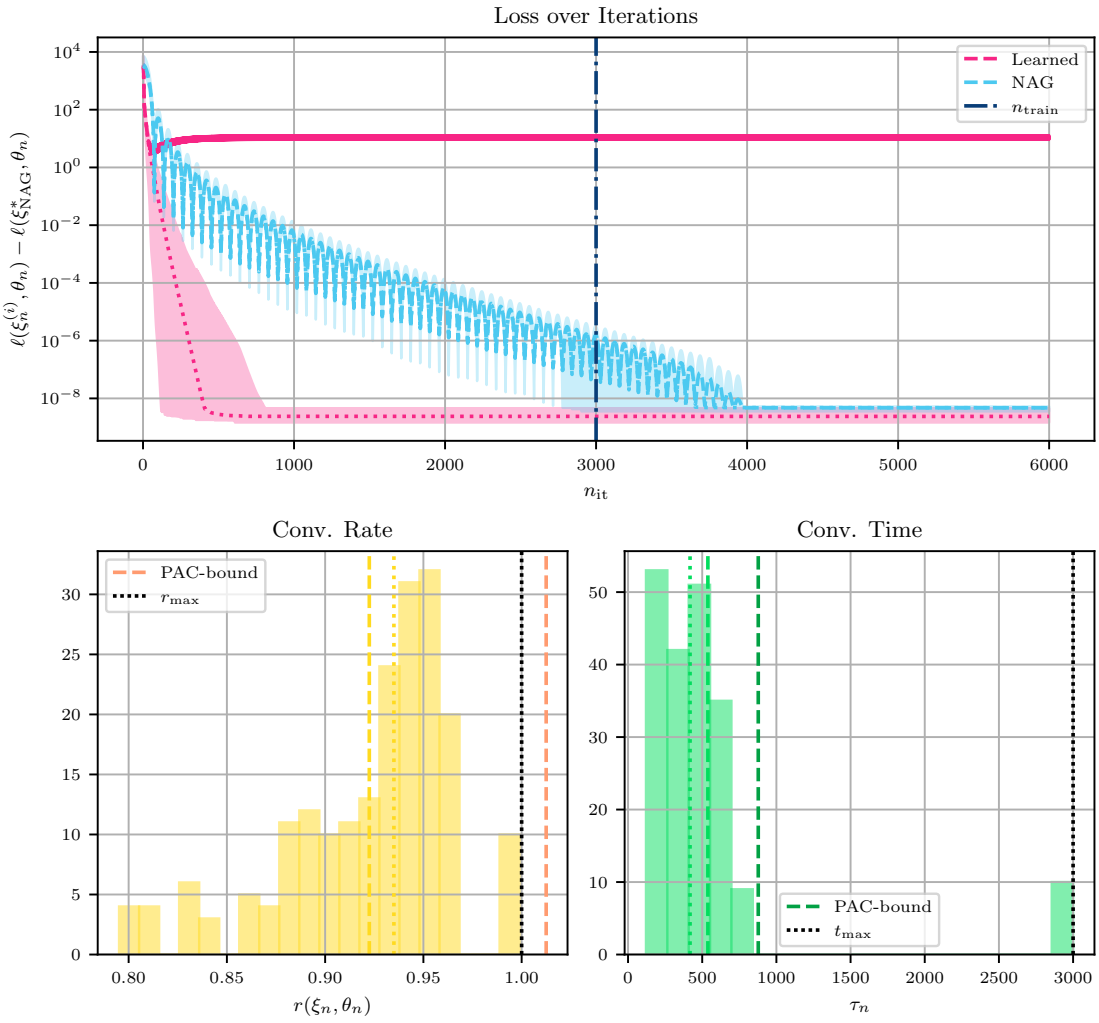

Figure 6: Image processing: The **top figure** shows the loss over the iterations, where NAG is shown in blue and the learned algorithm in pink. The mean and median are shown as dashed and dotted lines, respectively, while the shaded region represents the test data up to the quantile $q = 0.95$, that is, 95% of the test data. The **lower left plot** shows the convergence rate of the learned algorithm. The dashed lines represent the empirical mean and the PAC-bound, and we can see that the bound is vacuous here. Similarly, the **lower right figure** shows the convergence time of the learned algorithm. Again, the dashed lines represent the empirical mean and the corresponding PAC-bound, which, in this case, is quite tight.

6.3 LASSO

In this subsection, we consider the Lasso problem (Tibshirani, 1996), that is, a non-smooth problem of the form:

$$\min_{x\in\mathbb{R}^d} \frac{1}{2}\|Ax-b\|_2^2 + \lambda\|x\|_1 \qquad A\in\mathbb{R}^{p\times d},\ b\in\mathbb{R}^p\,,$$

with $p \leq d$. Hence, the optimization variable is given by $x \in \mathbb{R}^d$, and we use the same matrix $A \in \mathbb{R}^{p\times d}$ with dimensions $d = 70$ and $p = 35$ for all problem instances, where we sample each entry uniformly in $[-0.5, 0.5]$. Thus, the parameters θ are given by the right-hand side and the regularization parameter, that is, $\theta = (b, \lambda) \in \mathbb{R}^{p+1} =: P$. For this, the regularization parameter λ is sampled uniformly from $[5, 10]$, while the right-hand side is sampled from a multivariate normal distribution. Since the problem is convex and non-smooth, we use the FISTA algorithm (Beck and Teboulle, 2009) as baseline, which performs an extrapolation step followed by a proximal gradient step, that is, abbreviating $h(x) := \frac{1}{2}\|Ax-b\|^2$ and $g(x) := \lambda\|x\|_1$, the update is given by first computing $y^{(t)} = x^{(t)} + \beta_1^{(t)}\left(x^{(t)} - x^{(t-1)}\right)$ followed by setting $x^{(t+1)} = \operatorname{prox}_{\beta g}\left(y^{(t)} - \beta\nabla h(y^{(t)})\right)$. Here, the proximal mapping can be computed in closed-form yielding the *soft-thresholding operator* $\hat{x}_i = \mathbb{1}\{|\bar{x}_i| > \beta\lambda\}\cdot\left(\bar{x}_i - \beta\lambda\frac{\bar{x}_i}{|\bar{x}_i|}\right),\ i = 1, ..., d$. We choose $\beta = 1/L$, where L is the largest eigenvalue of A^TA, while $\beta_1^{(t)}$ is set to $\beta_1^{(t)} := (\beta_2^{(t)} - 1)/\beta_2^{(t+1)}$ with $\beta_2^{(t+1)} = (1 + \sqrt{1 + 4(\beta_2^{(t)})^2})/2$. Since the problem is non-smooth, the gradient norm cannot be used as stopping criterion. Thus, we define the convergence set as

$$\mathsf{C}_{\text{lasso}} := \left\{(\theta, s) \in P \times S \ :\ \|\Pi_S(s) - \operatorname{prox}_{\beta g}\left(\Pi_S(s) - \beta\nabla h(\Pi_S(s))\right)\| < 10^{-6}\right\}.$$

On the other hand, the learned algorithm $\mathcal{A}$ performs an update of the form $x^{(t+1)} = \operatorname{prox}_{\beta g}\left(x^{(t)} + \frac{1}{L}\left(d_1^{(t)} - \nabla h(x^{(t)}) + \|x^{(t)} - x^{(t-1)}\|\cdot d_1^{(t)}\right)\right)$, where $d_1^{(t)}$ and $d_2^{(t)}$ are predicted by a neural network. For more details on the architecture we refer to the Appendix J. The results are summarized in Figure 7: The upper plot shows that, on a typical example, the learned algorithm reaches the convergence criterion in about 200 iterations, and outperforms FISTA be many orders of magnitude. However, since the mean (dashed line) strongly deviates from the other 95% of the test problems (shaded region), we can observe that there are single problem instances on which the algorithm does not perform as good. The two lower plots show the convergence rate and time, respectively, together with the predicted PAC-bounds. In both cases, they are reasonable tight.

6.4 Training a Neural Network

In this subsection, we consider the problem of training a neural network on a regression problem, that is, $\mathcal{A}$ is trained to predict the parameters $\beta \in \mathbb{R}^p$ of a neural network $\mathtt{N}(\beta, \cdot)$, which then is used to predict a function $g : \mathbb{R} \to \mathbb{R}$. Hence, the optimization variable is given by $\beta \in \mathbb{R}^p$. We assume that the neural network should learn a function $g : \mathbb{R} \to \mathbb{R}$ from noisy observations $y_j = g(x_j) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$. For this, we construct polynomials g_i, $i = 1, ..., N$, of degree $d = 5$ by sampling points $\{x_{i,j}\}_{j=1}^K$ (here: $K = 50$) uniformly in $[-2, 2]$ and the coefficients $(c_{i,0}, ..., c_{i,5})$ of g_i uniformly in $[-5, 5]$. For every function $g_i : \mathbb{R} \to \mathbb{R}$

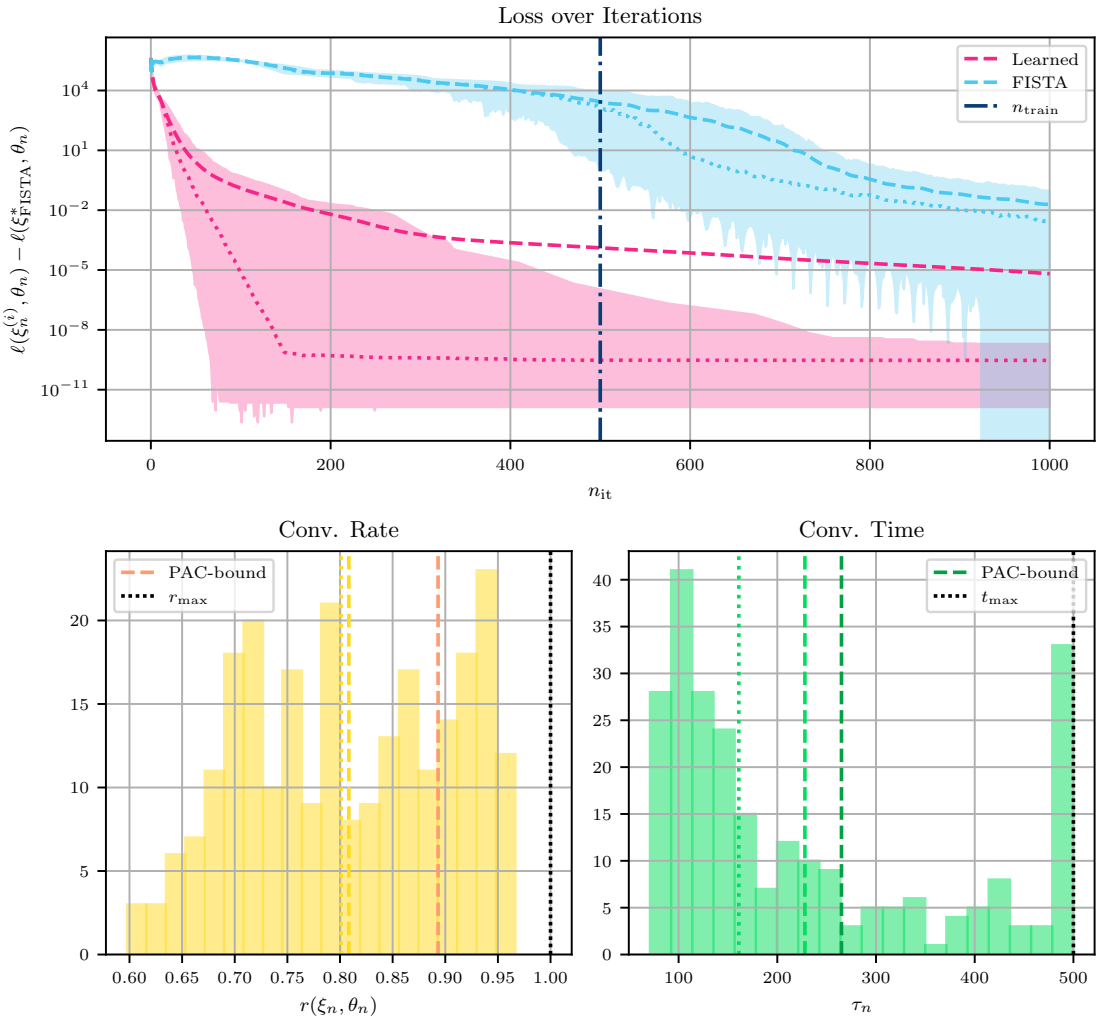

Figure 7: LASSO-Problem: The **top figure** shows the loss over the iterations, where FISTA is shown in blue and the learned algorithm in pink. The mean and median are shown as dashed and dotted lines, while the shaded region represents the test data up to the quantile $q = 0.95$, that is, 95% of the test data. We can see that, on a *typical* problem from this distribution, the learned algorithm reaches the stopping criterion in less than 200 iterations, and outperforms FISTA by several orders of magnitude. However, as the mean indicates, there are problem instances on which also the learned algorithm is rather slow. As the **lower left plot** shows, the learned algorithm strongly contracts the loss, on average, by a factor of 0.8 in each iteration. Similarly, the **lower right figure** shows that, typically, the learned algorithm reaches the convergence set in about 200 iterations. In both cases, the predicted PAC-bound is reasonable tight.

the neural network is trained on the data set $\theta_i := \{X_i, Y_i\}$ with $X_i = (x_{i,1}, ..., x_{i,K}) \in \mathbb{R}^K$ and $Y_i = (y_{i,1}, ..., y_{i,K}) \in \mathbb{R}^K$. Hence, the data set will serve as the parameter θ

of the loss function, such that the parameter space P can be identified as the space of these data sets, that is, $P = \mathbb{R}^{K\times 2}$. Since the mean square error is the standard choice for training models on regression tasks, the loss is given by $\ell(\beta, \theta_i) := c(\mathtt{N}(\beta, X_i), Y_i) := \frac{1}{K}\sum_{j=1}^{K}(\mathtt{N}(\beta, x_{i,j}) - y_{i,j})^2$, and for $\mathtt{N}$ we use a fully-connected two layer neural network with ReLU-activation functions. To have more features in the input layer, the input x is transformed into the vector $(x, x^2, ..., x^5)$. Hence, the parameters $\beta \in \mathbb{R}^p$ are given by the weights $A_1 \in \mathbb{R}^{50\times 5}, A_2 \in \mathbb{R}^{1\times 50}$ and biases $b_1 \in \mathbb{R}^{50}, b_2 \in \mathbb{R}$ of the two fully-connected layers. Therefore, the optimization space is of dimension $p = (5\cdot 50)+(1\cdot 50)+50+1 = 351$. As baseline we use Adam (Kingma and Ba, 2015) as it is implemented in PyTorch (Paszke et al., 2019), which is a widely used optimization algorithm for training neural networks. For tuning, we perform a grid search over 100 step-size parameters in $[10^{-4}, 10^{-2}]$, such that its performance is best for the given $n_{\text{train}} = 200$ iterations, which yields the value $\kappa = 0.008$. Note that, originally, Adam was introduced for stochastic optimization, while we use it in the "full-batch setting" here, that is, without stochasticity. We define the convergence set as

$$\mathsf{C}_{\text{nn}} := \{(\theta, s) \in P \times S \ : \ \|\nabla\ell(s, \theta)\| < 0.75 \text{ and } \ell(s, \theta) < 0.75\}\,.$$

These numbers are based on the fact that, over 10^4 iterations, Adam was not able to decrease the gradient norm below 0.5, and, since we add standard normal noise, the expected loss of the ground truth takes the value 1.0, that is, a loss below 1.0 can be regarded as overfitting. The learned algorithm simply performs the update $x^{(t+1)} = x^{(t)}+d^{(t)}$, where $d^{(t)}$ is predicted by a neural network. For more details on the architecture, we refer to the Appendix K. The upper plot of Figure 8 shows that the learned algorithm clearly outperforms Adam, reaching the ground-truth loss already after about 25 iterations. At the same time, Adam is not able to reach it within 400 iterations (on average), as the mean is still above 1.0. The lower left plot shows that the learned algorithm contracts the loss in each iteration, on average, by a factor of 0.9, and the lower right plot shows that it typically needs about 100 iterations to reach the convergence set. However, one can also observe that there is a significant amount of problems that do not reach the convergence set in 200 iterations, and that do not have a "fast convergence rate". These two findings are strongly correlated, because, if the loss plateaus but does not reach the convergence set, the estimated rate-function deteriorates. In this case, that is, a non-smooth and non-convex problem, it might be due to the fact that our definition of the convergence set is somewhat arbitrary. Nevertheless, in both cases, the provided PAC-bound is reasonably tight.

Remark 44 *Another definition that one could use is given by the procedure of* early stopping*, that is, one stops the algorithm as soon as the validation loss deviates from the training loss by a certain amount. Through this, one could get a guarantee for how long the network has to be trained until it starts to overfit to the training data.*

6.5 Stochastic Empirical Risk Minimization

Lastly, we consider the problem of stochastic empirical risk minimization. For this, we use the same problem and setup as in Subsection 6.4 for training a neural network. However, this time, in each iteration the algorithm only has access to a randomly selected minibatch

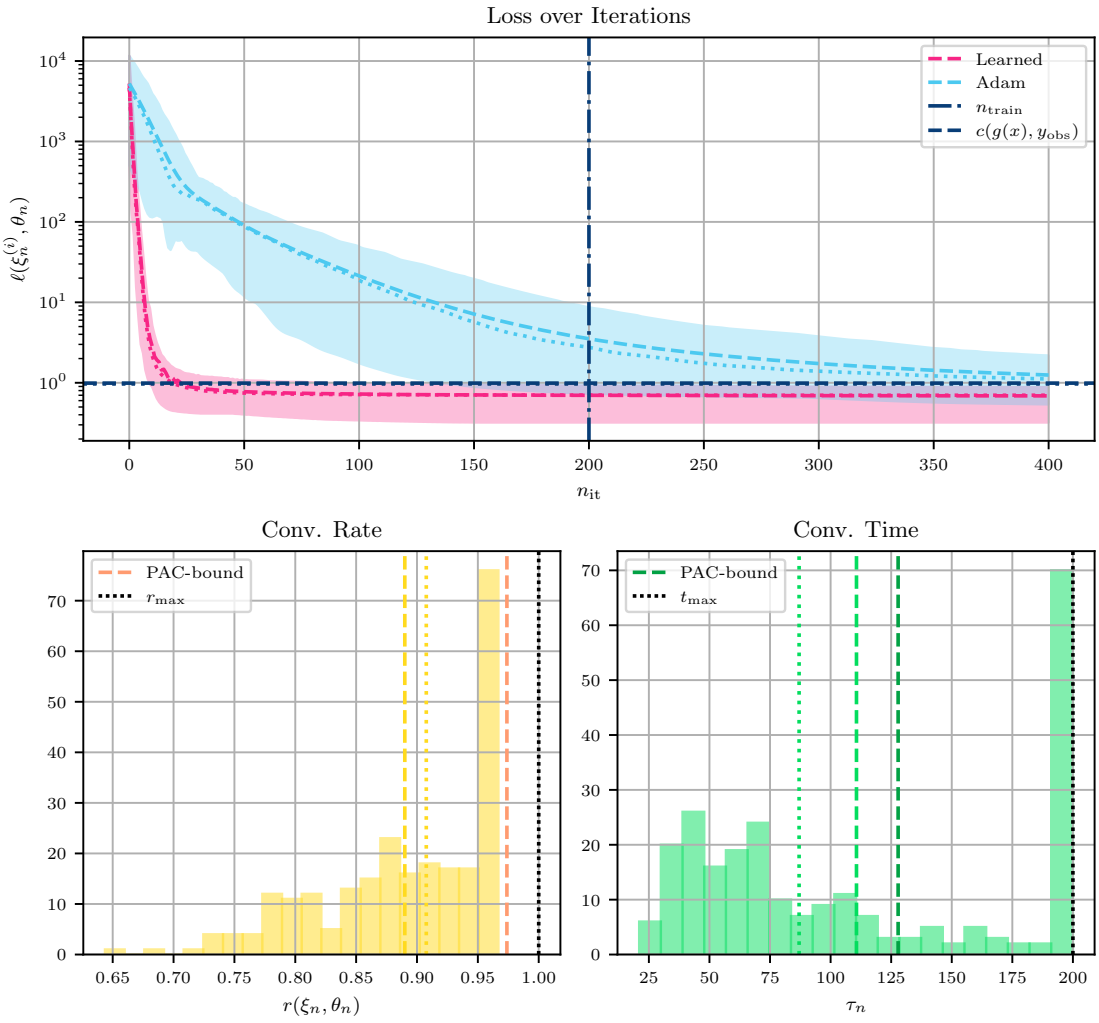

Figure 8: Training the neural network: The **top figure** shows the loss over the iterations, where Adam is shown in blue and the learned algorithm in pink. The mean and median are shown as dashed and dotted lines, respectively, while the shaded region represents the test data up to the quantile $q = 0.95$, that is, 95% of the test data. As can be seen, the learned algorithm reaches the ground-truth loss after about 25 iterations, and clearly outperforms Adam. The **lower left plot** shows the convergence rate of the learned algorithm, and we can see that it varies strongly. This is due to the fact that the loss plateaus quickly, but does not necessarily reach the convergence set, which can also be observed in the **lower right figure**, as there are about 70 problem instances, where the learned algorithm does not reach the convergence criterion in the given 200 iterations. However, on average, the learned algorithm reaches the convergence set in about 100 iterations.

(of size $m = 5$) instead of the full-batch setting considered before. Therefore, we have to deal with a stochastic, non-convex, and non-smooth optimization problem. Since we cannot

access the full gradient anymore, we define the convergence set as

$$\mathsf{C}_{\text{stoch}} := \{(\theta, s) \in P \times S \ : \ \ell(s, \theta) < 0.75\}\,.$$

As before, the value 0.75 is chosen due to our construction of the data set. As baseline, Adam is used again, where we perform a grid-search over 100 step-sizes in the interval $[10^{-6}, 10^{-1}]$ in such a way that its average performance is best after $n_{\text{train}} = 2500$ iterations, which yields the value of $\kappa = 5 \cdot 10^{-3}$. Here, the (empirical) average is taken over 25 problem instances, where we perform 10 runs per problem instance. As learned algorithm, we use a "preconditioned" version of Adam, that is, we add additional, learned parameters to enhance its performance. More details about the architecture are given in the Appendix L. The upper plot of Figure 9 shows the empirical risk over the iterations, where we have performed one run per problem. The plot shows that the learned algorithm still outperforms Adam, yet, not as clearly as in the full-batch case. The lower right plot shows that, on average, the learned algorithm reaches the convergence set in about 1000 iterations, and we can see that the provided PAC-bound is reasonably tight. Here, the median shows that more than 50% of the problems actually need less than 500 iterations, while the median of Adam (in the upper plot) indicates that the median convergence time for Adam is at about 2000 iterations. Thus, on a majority of instances, the learned algorithm needs way less iterations to reach the convergence set. On the other hand, however, the lower left plot shows that the provided bound for the convergence rate is vacuous. This again can be attributed to the fact that the algorithm has to perform many iterations to reach the convergence set, which in turn yields a value (for the convergence rate) close to 1.0, which then gives a vacuous PAC-bound.

7 Conclusion

The goal of this paper was to model optimization algorithms in learning-to-optimize more faithfully, so that it is possible to give generalization guarantees for (nearly) any kind of statistics that one wants to have for such an algorithm. In particular, this involves convergence rates and stopping times of the algorithm, which both depend on the *trajectory* of the algorithm instead of single iterates. Therefore, we introduced a probabilistic model for the distribution of the trajectory of an abstract, iterative optimization algorithm. Based on this model, we then provided generalization bounds for the convergence rate and the convergence time of the learned algorithm. However, both results are *non-asymptotic*: The stopping time has to be finite and the provided rate function is only an approximation to a "true" convergence rate, which, by construction of the algorithm, would be valid for any iterate. While the provided framework does, theoretically, allow for non-asymptotic results as it allows accessing the whole trajectory of the algorithm, they might not be within reach for *practical generalization results*, simply because of the fact that asymptotic events are inherently non-observable.

Acknowledgments and Disclosure of Funding

M. Sucker and P. Ochs acknowledge funding by the German Research Foundation under Germany's Excellence Strategy – EXC number 2064/1 – 390727645.

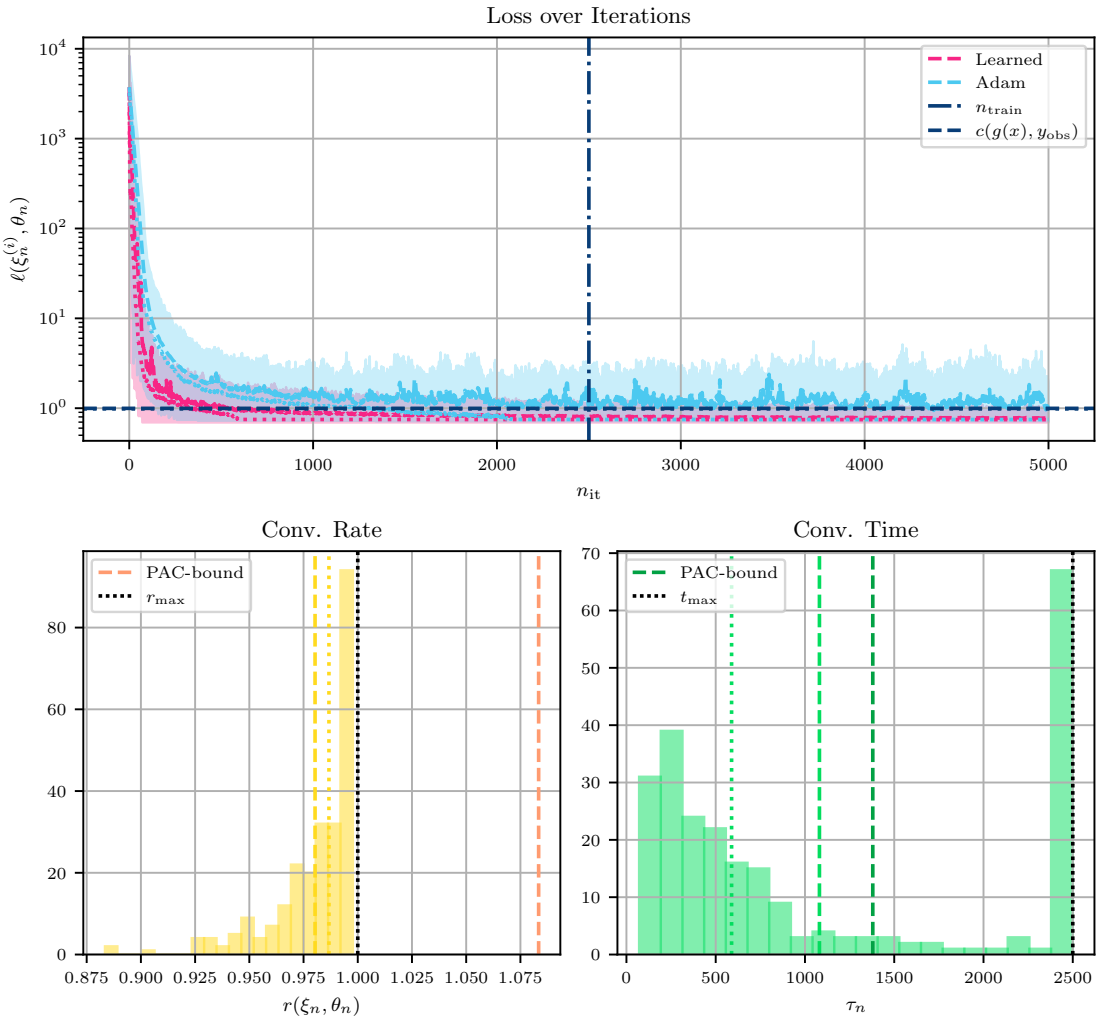

Figure 9: Stochastic empirical risk minimization: The **top figure** shows the loss (emp. risk) over the iterations, where Adam is shown in blue and the learned algorithm in pink. The mean and median are shown as dashed and dotted lines, respectively, while the shaded region represents the test data up to the quantile $q = 0.95$, that is, 95% of the test data. Here, the learned algorithm reaches the ground-truth faster than Adam, an its performance varies way less. The **lower left plot** shows the convergence rate of the learned algorithm and we can see that, unfortunately, the provided PAC-bound is vacuous. This is due to the fact that the loss plateaus quickly, while the algorithm still needs many iterations to reach the convergence set. However, the **lower right figure** shows that, on average, the learned algorithm reaches the convergence set in about 1000 iterations and the PAC-bound provides a good estimate for this.

Appendix A. Extension of Factorization to Joint Semi-Group

Lemma 45 *For every $(\alpha, \theta_{[N]}, x_{[N]}) \in H \times P^N \times S^N$, $K \in \mathbb{N}$, $t_0 < t_1 < ... < t_K \in \mathbb{N}_0$, and any set $\left(\mathsf{B}_1^{t_0} \times ... \times \mathsf{B}_N^{t_0}\right) \times ... \times \left(\mathsf{B}_1^{t_K} \times ... \times \mathsf{B}_N^{t_K}\right)$ with $\mathsf{B}_j^{t_k} \in \mathcal{B}(S)$, it holds that:*

$$\left(\delta_{x_{[N]}} \otimes \bigotimes_{k=0}^{K-1} \Gamma^{t_{k+1}-t_k}(\alpha, \theta_{[N]}, \cdot)\right) \left\{\left(\mathsf{B}_1^{t_0} \times ... \times \mathsf{B}_N^{t_0}\right) \times ... \times \left(\mathsf{B}_1^{t_K} \times ... \times \mathsf{B}_N^{t_K}\right)\right\}$$
$$= \prod_{n=1}^{N} \left(\delta_{x_n} \otimes \bigotimes_{k=0}^{K-1} \gamma^{t_{k+1}-t_k}(\alpha, \theta_n, \cdot)\right) \left\{\mathsf{B}_n^{t_0} \times ... \times \mathsf{B}_n^{t_K}\right\}$$

Proof Inserting $B_j^{t_k} = S$ if necessary, w.l.o.g. we can restrict to the case $t_k = k$, that is, $t_{k+1} - t_k = 1$. Then, we prove the result by induction. Thus, first, consider the case $K = 1$:

$$\left(\delta_{x_{[N]}} \otimes \Gamma(\alpha, \theta_{[N]}, \cdot)\right) \left\{\left(\mathsf{B}_1^0 \times ... \times \mathsf{B}_N^0\right) \times \left(\mathsf{B}_1^1 \times ... \times \mathsf{B}_N^1\right)\right\}$$
$$= \delta_{x_{[N]}} \left[\mathbb{1}_{\mathsf{B}_1^0 \times ... \times \mathsf{B}_N^0} \Gamma(\alpha, \theta_{[N]}, \cdot, \mathsf{B}_1^1 \times ... \times \mathsf{B}_N^1)\right] .$$

By the factorization of Γ and Fubini's theorem this can be written as:

$$= \delta_{x_{[N]}} \left[\prod_{n=1}^{N} \mathbb{1}_{\mathsf{B}_n^0} \gamma(\alpha, \theta_n, \cdot, \mathsf{B}_n^1)\right] = \prod_{n=1}^{N} \delta_{x_n} \left[\mathbb{1}_{\mathsf{B}_n^0} \gamma(\alpha, \theta_n, \cdot, \mathsf{B}_n^1)\right]$$
$$= \prod_{n=1}^{N} \left(\delta_{x_n} \otimes \gamma(\alpha, \theta_n, \cdot) \left\{\mathsf{B}_n^0 \times \mathsf{B}_n^1\right\}\right) .$$

Thus, let the statement be true for $K \in \mathbb{N}$ and consider the case $K + 1$. Since we have the recursive definition $\bigotimes_{k=0}^{K-1} \Gamma(\alpha, \theta_{[N]}, \cdot) = \left(\bigotimes_{k=0}^{K-2} \Gamma(\alpha, \theta_{[N]}, \cdot)\right) \otimes \Gamma(\alpha, \theta_{[N]}, \cdot)$, the statement follows directly from the factorization of Γ, the induction hypothesis, and Fubini's theorem:

$$\left(\delta_{x_{[N]}} \otimes \bigotimes_{k=0}^{K} \Gamma(\alpha, \theta_{[N]}, \cdot)\right) \left\{\left(\mathsf{B}_1^0 \times ... \times \mathsf{B}_N^0\right) \times ... \times \left(\mathsf{B}_1^{K+1} \times ... \times \mathsf{B}_N^{K+1}\right)\right\}$$
$$= \left(\delta_{x_{[N]}} \otimes \bigotimes_{k=0}^{K-1} \Gamma(\alpha, \theta_{[N]}, \cdot)\right) \left[\mathbb{1}_{\mathsf{B}_1^0 \times ... \times \mathsf{B}_N^K} \Gamma(\alpha, \theta_{[N]}, \cdot, \mathsf{B}_1^{K+1} \times ... \times \mathsf{B}_N^{K+1})\right]$$
$$= \left(\delta_{x_{[N]}} \otimes \bigotimes_{k=0}^{K-1} \Gamma(\alpha, \theta_{[N]}, \cdot)\right) \left[\prod_{n=1}^{N} \mathbb{1}_{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K} \gamma(\alpha, \theta_{[N]}, \cdot, \mathsf{B}_n^{K+1})\right]$$
$$= \prod_{n=1}^{N} \left(\delta_{x_n} \otimes \bigotimes_{k=0}^{K-1} \gamma(\alpha, \theta_n, \cdot)\right) \left[\mathbb{1}_{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^K} \gamma(\alpha, \theta_{[N]}, \cdot, \mathsf{B}_n^{K+1})\right]$$
$$= \prod_{n=1}^{N} \left(\delta_{x_n} \otimes \bigotimes_{k=0}^{K} \gamma(\alpha, \theta_n, \cdot)\right) \left\{\mathsf{B}_n^0 \times ... \times \mathsf{B}_n^{K+1}\right\} .$$

■

Appendix B. Proof of Theorem 19

Proof By construction, we have that $\psi(\alpha, \theta, \cdot)$ is a probability measure on $S^{\mathbb{N}_0}$ for each $(\alpha, \theta) \in H \times P$. Therefore, we only have to show measurability. Again, we do this by a monotone-class argument. For this, define the following two classes of sets:

$$\mathcal{D} := \{\mathsf{A} \in \mathcal{B}(S^{\mathbb{N}_0}) \ : \ (\alpha, \theta) \mapsto \psi_{\alpha,\theta}(\mathsf{A}) \text{ is measurable}\}$$
$$\mathcal{C} := \{\mathsf{A}^0 \times ... \times \mathsf{A}^K \times \prod_{k>K} S \ : \ K \in \mathbb{N}_0, \quad \mathsf{A}^0, ..., \mathsf{A}^K \in \mathcal{B}(S)\}\,.$$

The first step of the proof is to show that $\mathcal{C} \subset \mathcal{D}$:

$$\psi\left((\alpha, \theta), \mathsf{A}^0 \times ... \times \mathsf{A}^K \times \prod_{k>K} S\right) = \psi_{\alpha,\theta}\left\{\mathsf{A}^0 \times ... \times \mathsf{A}^K \times \prod_{k>K} S\right\}$$
$$= \left(\psi_{\alpha,\theta} \circ \mathscr{X}^{-1}_{\{0,...,K\}}\right)\{\mathsf{A}^0 \times ... \times \mathsf{A}^K\} = \left(\mathbb{P}_{\mathscr{I}} \otimes \bigotimes_{k=0}^{K-1} \gamma^1(\alpha, \theta, \cdot)\right)\{\mathsf{A}^0 \times ... \times \mathsf{A}^K\}$$
$$= \left(\mathbb{P}_{\mathscr{I}} \otimes \bigotimes_{k=0}^{K-1} \gamma(\alpha, \theta, \cdot)\right)\{\mathsf{A}^0 \times ... \times \mathsf{A}^K\}\,.$$

Since γ is a probability kernel from $H \times P \times S$ to S, by Kallenberg (2021, Lemma 3.3 (i), p.58) it holds that $\mathbb{P}_{\mathscr{I}} \otimes \bigotimes_{k=0}^{K-1} \gamma(\alpha, \theta, \cdot)$ is a probability kernel from $H \times P$ to S^{K+1}. Thus, the map $(\alpha, \theta) \mapsto \mathbb{P}_{\mathscr{I}} \otimes \bigotimes_{k=0}^{K-1} \gamma(\alpha, \theta, \cdot)$ is measurable. Therefore, the map $(\alpha, \theta) \mapsto \psi\left((\alpha, \theta), \mathsf{A}^0 \times ... \times \mathsf{A}^K \times \prod_{k>K} S\right)$ is measurable. Hence, we get that

$$\mathcal{C} \subset \mathcal{D}\,.$$

Further, since $\mathcal{C}$ is the class of cylinder sets, it is clearly $\cap$-stable, that is, it is a π-system. The second step of the proof is to show that $\mathcal{D}$ is a λ-system. Since $S^{\mathbb{N}_0} \in \mathcal{C} \subset \mathcal{D}$, we have $S^{\mathbb{N}_0} \in \mathcal{D}$. Hence, take $\mathsf{A}, \mathsf{B} \in \mathcal{D}$ with $\mathsf{A} \supset \mathsf{B}$. By definition of $\mathcal{D}$, we have that both $(\alpha, \theta) \mapsto \psi_{\alpha,\theta}(\mathsf{A})$ and $(\alpha, \theta) \mapsto \psi_{\alpha,\theta}(\mathsf{B})$ are measurable. However, this implies that the map $(\alpha, \theta) \mapsto \psi_{\alpha,\theta}(\mathsf{A} \setminus \mathsf{B}) = \psi_{\alpha,\theta}(\mathsf{A}) - \psi_{\alpha,\theta}(\mathsf{B})$ is measurable, where the (point-wise) equality follows from $\psi_{\alpha,\theta}$ being a probability measure for each $(\alpha, \theta) \in H \times P$. Therefore, we have that $\mathsf{A} \setminus \mathsf{B} \in \mathcal{D}$, and $\mathcal{D}$ is closed under proper differences. Finally, take $\mathsf{A}_1, \mathsf{A}_2, ... \in \mathcal{D}$ with $\mathsf{A}_n \uparrow \mathsf{A}$. Then, since $\psi_{\alpha,\theta}$ is a measure for each $(\alpha, \theta) \in H \times P$, we have the equality:

$$\psi(\alpha, \theta, \mathsf{A}) = \psi_{\alpha,\theta}(\mathsf{A}) = \lim_{n\to\infty} \psi_{\alpha,\theta}(\mathsf{A}_n) = \lim_{n\to\infty} \psi(\alpha, \theta, \mathsf{A}_n)\,.$$

By definition of $\mathcal{D}$, we have that $(\alpha, \theta) \mapsto \psi(\alpha, \theta, \mathsf{A}_n)$ is measurable for each $n \in \mathbb{N}$. Thus, the map $(\alpha, \theta) \mapsto \psi(\alpha, \theta, \mathsf{A})$ is the pointwise limit of measurable functions, and therefore measurable itself. Hence, we have $\mathsf{A} \in \mathcal{D}$, such that $\mathcal{D}$ is a λ-system. Then, by Theorem 8 we get that

$$\sigma(\mathcal{C}) \subset \mathcal{D}\,.$$

However, since $\mathcal{C}$ is the class of cylinder sets, which, by definition, is a $\cap$-stable generator of the product σ-algebra on $S^{\mathbb{N}_0}$, we have that $\sigma(\mathcal{C}) = \mathcal{B}(S^{\mathbb{N}_0})$. Thus, the map $(\alpha, \theta) \mapsto \psi_{\alpha,\theta}(\mathsf{A})$

is measurable for each $\mathsf{A} \in \mathcal{B}(S^{\mathbb{N}_0})$. It follows that ψ is a probability kernel from $H \times P$ to $S^{\mathbb{N}_0}$. Since $\psi_{\alpha,\theta}$ is unique, also $\psi : H \times P \to S^{\mathbb{N}_0}$ is unique. This concludes the proof for ψ, and the statement for Ψ follows by the same argument. ■

Appendix C. Proof of Lemma 24

Proof

(i) We have to show that for any $\mathsf{B} \in \mathcal{B}\left(S^N\right)^{\otimes \mathbb{N}_0}$ and $\mathsf{A} \in \mathcal{B}\left(H \times P^N\right)$ it holds that:

$$\mathbb{E}_{(\mathscr{H},\mathscr{P}_{[N]})}\left\{\mathbb{1}_{\mathsf{A}}\Psi(\cdot,\cdot,\mathsf{B})\right\} = \mathbb{P}_{(\mathscr{H},\mathscr{P}_{[N]},\xi_{[N]})}\left\{\mathsf{A}\times\mathsf{B}\right\} .$$

Since $\mathbb{P}_{(\mathscr{H},\mathscr{P}_{[N]},\xi_{[N]})} = \mathbb{P}$ (coordinate projections), this holds by construction of $\mathbb{P}$.

(ii) Since $\mathbb{P}_{(\mathscr{H},\mathscr{P}_{[N]})} = \mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}_{[N]}} = \mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}}^{\otimes N}$, this follows directly from (i).

(iii) We have to show that for any $\mathsf{B} \in \mathcal{B}\left(S\right)^{\otimes \mathbb{N}_0}$ and $\mathsf{A} \in \mathcal{B}\left(H \times P\right)$ it holds that:

$$\mathbb{E}_{(\mathscr{H},\mathscr{P}_{n})}\left\{\mathbb{1}_{\mathsf{A}}\psi(\cdot,\cdot,\mathsf{B})\right\} = \mathbb{P}_{(\mathscr{H},\mathscr{P}_{n},\xi_{n})}\left\{\mathsf{A}\times\mathsf{B}\right\} .$$

Since $\mathcal{B}(H \times P)$ is generated by the cylinder sets, it suffices to consider A of the form $\mathsf{A} = \mathsf{C} \times \mathsf{D}$. Here, one gets:

$$\mathbb{E}_{(\mathscr{H},\mathscr{P}_{n})}\left\{\mathbb{1}_{\mathsf{C}\times\mathsf{D}}\psi(\cdot,\cdot,\mathsf{B})\right\} = \int_H \mathbb{P}_{\mathscr{H}}(d\alpha)\ \mathbb{1}_{\mathsf{C}}(\alpha) \int \mathbb{P}_{\mathscr{P}_n}(d\theta_n)\ \mathbb{1}_{\mathsf{D}}(\theta_n)\psi(\alpha,\theta_n,\mathsf{B}) .$$

By inserting $1 = \prod_{i\neq n}^{N} \mathbb{1}_P(\theta_i)\psi\left(\alpha,\theta_i,S^{\mathbb{N}_0}\right)$ and using Lemma 21, this is the same as:

$$\begin{aligned}&\mathbb{P}\left\{\mathscr{H}\in\mathsf{C},\ \mathscr{P}_{[N]}\in P\times\ldots\times\mathsf{D}\times\ldots\times P,\ \xi_{[N]}\in S^{\mathbb{N}_0}\times\ldots\times\mathsf{B}\times\ldots\times S^{\mathbb{N}_0}\right\}\\ &= \mathbb{P}\left\{\mathscr{H}\in\mathsf{C},\ \mathscr{P}_n\in\mathsf{D},\ \xi_n\in\mathsf{B}\right\} .\end{aligned}$$

(iv) Since $\mathbb{P}_{(\mathscr{H},\mathscr{P}_{n})} = \mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}_n} = \mathbb{P}_{\mathscr{H}} \otimes \mathbb{P}_{\mathscr{P}}$, this follows directly from (iii).

■

Appendix D. Proof of Lemma 32

Proof By Lemma 21 and Corollary 27 it holds that:

$$\begin{aligned}&\mathbb{E}\left\{\exp\left(\lambda\left(\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\left\{f(\mathscr{P}_n,\xi_n)\mid\mathscr{H},\mathscr{P}_n\right\}-\mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{f\right\}\right)-\frac{\lambda^2}{2N}f_{\max}^2\right)\right\}\\ &= \mathbb{E}\left\{\exp\left(\lambda\left(\mathbb{E}\left\{\frac{1}{N}\sum_{n=1}^{N}f(\mathscr{P}_n,\xi_n)\ \middle|\ \mathscr{H},\mathscr{P}_{[N]}\right\}-\mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{f\right\}\right)-\frac{\lambda^2}{2N}f_{\max}^2\right)\right\} .\end{aligned}$$

Since the other terms are constant w.r.t. Ψ, this is the same as:

$$= \mathbb{E}\left\{\exp\left(\mathbb{E}\left\{\lambda\left(\frac{1}{N}\sum_{n=1}^{N} f(\mathscr{P}_n, \xi_n) - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}\right) - \frac{\lambda^2}{2N} f_{\max}^2 \;\middle|\; \mathscr{H}, \mathscr{P}_{[N]}\right\}\right)\right\}.$$

By Jensen's inequality, this can be bounded by:

$$\leq \mathbb{E}\left\{\mathbb{E}\left\{\exp\left(\lambda\left(\frac{1}{N}\sum_{n=1}^{N} f(\mathscr{P}_n, \xi_n) - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}\right) - \frac{\lambda^2}{2N} f_{\max}^2\right) \;\middle|\; \mathscr{H}, \mathscr{P}_{[N]}\right\}\right\}.$$

Since we take the (total) expectation on the outside, this is the same as:

$$= \mathbb{E}\left\{\mathbb{E}\left\{\exp\left(\lambda\left(\frac{1}{N}\sum_{n=1}^{N} f(\mathscr{P}_n, \xi_n) - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}\right) - \frac{\lambda^2}{2N} f_{\max}^2\right) \;\middle|\; \mathscr{H}\right\}\right\}.$$

Since $f_{\max}$ is a constant, and by the properties of the exponential function, this can be written as:

$$= \mathbb{E}\left\{\exp\left(-\frac{\lambda^2}{2N} f_{\max}^2\right)\mathbb{E}\left\{\prod_{n=1}^{N}\exp\left(\frac{\lambda}{N}\left(f(\mathscr{P}_n, \xi_n) - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}\right)\right) \;\middle|\; \mathscr{H}\right\}\right\}.$$

By Lemma 21 and Fubini's theorem, this is the same as:

$$= \mathbb{E}\left\{\exp\left(-\frac{\lambda^2}{2N} f_{\max}^2\right)\prod_{n=1}^{N}\mathbb{E}\left\{\exp\left(\frac{\lambda}{N}\left(f(\mathscr{P}_n, \xi_n) - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}\right)\right) \;\middle|\; \mathscr{H}\right\}\right\}$$
$$= \mathbb{E}\left\{\exp\left(-\frac{\lambda^2}{2N} f_{\max}^2\right)\prod_{n=1}^{N}\mathbb{E}_{(\mathscr{P}_n,\xi_n)|\mathscr{H}}\left\{\exp\left(\frac{\lambda}{N}\left(f - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}\right)\right)\right\}\right\}.$$

Since the parameters are i.i.d., this is the same as:

$$= \mathbb{E}\left\{\exp\left(-\frac{\lambda^2}{2N} f_{\max}^2\right)\prod_{n=1}^{N}\mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\exp\left(\frac{\lambda}{N}\left(f - \mathbb{E}_{(\mathscr{P},\xi)|\mathscr{H}}\{f\}\right)\right)\right\}\right\}.$$

By Hoeffding's lemma, this can be bounded by:

$$\leq \mathbb{E}\left\{\exp\left(-\frac{\lambda^2}{2N} f_{\max}^2\right)\prod_{n=1}^{N}\exp\left(\frac{\lambda^2}{8N^2}(2f_{\max})^2\right)\right\} \leq 1\,.$$

This concludes the proof. ■

Appendix E. Proof of Lemma 40

Proof That Φ is one-to-one can be observed by just plugging in the formula for Φ_a^{-1}, which then also shows (iii). The first derivative of Φ_a is given by:

$$\frac{\partial}{\partial p}\Phi_a(p) = \frac{1}{a}\frac{1-\exp(-a)}{1-[1-\exp(-a)]p}\,.$$

The only way that this term could be negative is given when $1-\exp(-a)$ is negative. In this case, the whole second term has to be negative, as the denominator is positive. However, since $1-\exp(-a)$ can only be negative, when a is, this shows that $\frac{\partial}{\partial p}\Phi_a(p)$ is positive. Thus, Φ_a is increasing. Similarly, the second derivative of Φ_a is given by:

$$\frac{\partial^2}{\partial p^2}\Phi_a(p) = \frac{1}{a}\left(\frac{1-\exp(-a)}{1-[1-\exp(-a)]p}\right)^2\,.$$

This term is strictly negative/positive, when a is, and therefore shows (ii). ■

Appendix F. Proof of Corollary 41

Proof By Lemma 39 and Lemma 40, for every $\lambda \in \mathbb{R}$ we have the equality:

$$\mathbb{E}\left\{\exp\left(\frac{\lambda}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}\left(\mathscr{P}_n,\xi_n\right)\right)\right\} = \mathbb{E}\left\{\exp\left(-\lambda\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right)\right)\right\}\,,$$

The directly yields the first statement. Further, one gets from Fubini's theorem:

$$\begin{aligned}
&\mathbb{E}_{\mathscr{P}_{[N]}}\left\{\mathbb{E}_{\mathscr{H}}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}\left\{\mathsf{A}\right\}\right]\right)\right\}\right\}\\
&= \mathbb{E}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}\left\{\mathsf{A}\right\}\right]\right)\right\}\\
&= \mathbb{E}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\left\{\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\mid\mathscr{H},\mathscr{P}_n\right\}\right]\right)\right\}\,.
\end{aligned}$$

By Lemma 27, this is the same as:

$$\begin{aligned}
&= \mathbb{E}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \mathbb{E}\left\{\frac{1}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\mid\mathscr{H},\mathscr{P}_{[N]}\right\}\right]\right)\right\}\\
&= \mathbb{E}\left\{\exp\left(\mathbb{E}\left\{\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\mid\mathscr{H},\mathscr{P}_{[N]}\right]\right\}\right)\right\}\,.
\end{aligned}$$

By Jensen's inequality, this can be bounded by:

$$\leq \mathbb{E}\left\{\mathbb{E}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\right]\right) \;\middle|\; \mathscr{H}, \mathscr{P}_{[N]}\right\}\right\}$$
$$= \mathbb{E}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(\mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\left\{\mathsf{A}\right\}\right) - \frac{1}{N}\sum_{n=1}^{N}\mathbb{1}_{\mathsf{A}}(\mathscr{P}_n,\xi_n)\right]\right)\right\} = 1\,.$$

■

Appendix G. Proof of Theorem 42

Proof Abbreviate $p := \mathbb{P}_{(\mathscr{P},\xi)|\mathscr{H}}\{\mathsf{A}\}$ and $\hat{p} := \frac{1}{N}\sum_{n=1}^{N}\mathbb{P}_{(\mathscr{P}_n,\xi_n)|\mathscr{H},\mathscr{P}_n}\{\mathsf{A}\}$. Then, by Corollay 41 it holds that:

$$\mathbb{E}_{\mathscr{P}_{[N]}}\left\{\mathbb{E}_{\mathscr{H}}\left\{\exp\left(\lambda\left[\Phi_{\frac{\lambda}{N}}\left(p(\mathscr{H})\right) - \hat{p}(\mathscr{H})\right]\right)\right\}\right\} \leq 1\,.$$

Therefore, by the Donsker-Varadhan variational formulation, we have:

$$1 \geq \mathbb{E}_{\mathscr{P}_{[N]}}\left\{\exp\left(\sup_{\rho\in\mathcal{P}(\mathbb{P}_{\mathscr{H}})}\ \lambda\rho\left[\Phi_{\frac{\lambda}{N}}\circ p - \hat{p}\right] - D_{\mathrm{KL}}(\rho \parallel \mathbb{P}_{\mathscr{H}})\right)\right\}$$
$$= \mathbb{E}_{\mathscr{P}_{[N]}}\left\{\exp\left(\sup_{\rho\in\mathcal{P}(\mathbb{P}_{\mathscr{H}})}\ \lambda\left(\rho\left[\Phi_{\frac{\lambda}{N}}\circ p\right] - \rho\left[\hat{p}\right]\right) - D_{\mathrm{KL}}(\rho \parallel \mathbb{P}_{\mathscr{H}})\right)\right\}.$$

Since $\lambda > 0$, Φ is convex by Lemma 40. Thus, applying Jensen's inequality yields:

$$\geq \mathbb{E}_{\mathscr{P}_{[N]}}\left\{\exp\left(\sup_{\rho\in\mathcal{P}(\mathbb{P}_{\mathscr{H}})}\ \lambda\left(\Phi_{\frac{\lambda}{N}}\left(\rho\left[p\right]\right) - \rho\left[\hat{p}\right]\right) - D_{\mathrm{KL}}(\rho \parallel \mathbb{P}_{\mathscr{H}})\right)\right\}$$

Then, Markov's inequality yields:

$$\mathbb{P}_{\mathscr{P}_{[N]}}\left\{\sup_{\rho\in\mathcal{P}(\mathbb{P}_{\mathscr{H}})}\ \lambda\left(\Phi_{\frac{\lambda}{N}}\left(\rho\left[p\right]\right) - \rho\left[\hat{p}\right]\right) - D_{\mathrm{KL}}(\rho \parallel \mathbb{P}_{\mathscr{H}}) \geq \log\left(\frac{1}{\varepsilon}\right)\right\} \leq \varepsilon\,,$$

from which the conclusion follows. ■

References

Pierre Alquier. User-friendly introduction to PAC-Bayes bounds. *arXiv preprint arXiv:2110.11216*, 2021.

Pierre Alquier and Benjamin Guedj. Simpler PAC-Bayesian bounds for hostile data. *Machine Learning*, 107(5):887–902, 2018.

Pierre Alquier, James Ridgway, and Nicolas Chopin. On the properties of variational approximations of Gibbs posteriors. *Journal of Machine Learning Research*, 17(1):8374–8414, 2016.

Ron Amit, Baruch Epstein, Shay Moran, and Ron Meir. Integral probability metrics PAC-Bayes bounds. *Advances in Neural Information Processing Systems*, 35:3123–3136, 2022.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Luc Bégin, Pascal Germain, François Laviolette, and Jean-Francis Roy. PAC-Bayesian bounds based on the Rényi divergence. In *Artificial Intelligence and Statistics*, pages 435–444. PMLR, 2016.

Pascal Bianchi, Walid Hachem, and Sholom Schechtman. Convergence of constant step stochastic gradient descent for non-smooth non-convex functions. *Set-Valued and Variational Analysis*, pages 1–31, 2022.

Leon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Olivier Bousquet and André Elisseeff. Algorithmic stability and generalization performance. *Advances in Neural Information Processing Systems*, 13, 2000.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

Gregery T Buzzard, Stanley H Chan, Suhas Sreehari, and Charles A Bouman. Plug-and-play unplugged: Optimization-free reconstruction using consensus equilibrium. *SIAM Journal on Imaging Sciences*, 11(3):2001–2020, 2018.

Camille Castera and Peter Ochs. From learning to optimize to learning optimization algorithms. *arXiv preprint arXiv:2405.18222*, 2024.

Olivier Catoni. *Statistical learning theory and stochastic optimization: Ecole d'Eté de Probabilités de Saint-Flour, XXXI-2001*, volume 1851. Springer Science & Business Media, 2004.

Olivier Catoni. PAC-Bayesian supervised classification: The thermodynamics of statistical learning. *Lecture Notes-Monograph Series*, 56:i–163, 2007.

Stanley H Chan, Xiran Wang, and Omar A Elgendy. Plug-and-play ADMM for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging*, 3(1):84–98, 2016.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *arXiv preprint arXiv:2103.12828*, 2021.

Xiaohan Chen, Jialin Liu, Zhangyang Wang, and Wotao Yin. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. *Advances in Neural Information Processing Systems*, 31, 2018.

Xinshi Chen, Yufei Zhang, Christoph Reisinger, and Le Song. Understanding deep architecture with reasoning layer. *Advances in Neural Information Processing Systems*, 33: 1240–1252, 2020.

Regev Cohen, Michael Elad, and Peyman Milanfar. Regularization by denoising via fixed-point projection. *SIAM Journal on Imaging Sciences*, 14(3):1374–1406, 2021.

Damek Davis and Dmitriy Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Damek Davis and Dmitriy Drusvyatskiy. Graphical convergence of subgradients in nonconvex optimization and learning. *Mathematics of Operations Research*, 47(1):209–231, 2022.

Alexandre Défossez, Leon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and Adagrad. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain Markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.

Gintare Karolina Dziugaite and Daniel M Roy. Data-dependent PAC-Bayes priors via differential privacy. *Advances in neural information processing systems*, 31, 2018.

Gintare Karolina Dziugaite, Kyle Hsu, Waseem Gharbieh, Gabriel Arpino, and Daniel Roy. On the role of data in PAC-Bayes bounds. In *International Conference on Artificial Intelligence and Statistics*, pages 604–612. PMLR, 2021.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 353–360, 2009.

Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

Benjamin Guedj. A primer on PAC-Bayesian learning. In *Proceedings of the second congress of the French Mathematical Society*, volume 33, 2019.

Maxime Haddouche and Benjamin Guedj. PAC-Bayes generalisation bounds for heavy-tailed losses through supermartingales. *arXiv preprint arXiv:2210.00928*, 2022.

Maxime Haddouche and Benjamin Guedj. Wasserstein PAC-Bayes learning: Exploiting optimisation guarantees to explain generalisation, 2023.

Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Jean Honorio and Tommi Jaakkola. Tight bounds for the expected risk of linear classifiers and PAC-Bayes finite-sample guarantees. In *Artificial Intelligence and Statistics*, pages 384–392. PMLR, 2014.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

O. Kallenberg. *Foundations of Modern Probability*. Probability theory and stochastic modelling. Springer, 2021. ISBN 9783030618728.

Ali Kavis, Kfir Yehuda Levy, and Volkan Cevher. High probability bounds for a class of nonconvex algorithms with AdaGrad stepsize. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=dSw0QtRMJkO`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Achim Klenke. *Wahrscheinlichkeitstheorie*. Springer Spektrum, 2013.

Erich Kobler, Alexander Effland, Karl Kunisch, and Thomas Pock. Total deep variation: A stable regularizer for inverse problems. *arXiv preprint arXiv:2006.08789*, 2020.

John Langford and Rich Caruana. (Not) bounding the true error. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

John Langford and John Shawe-Taylor. PAC-Bayes and margins. *Advances in neural information processing systems*, 15, 2002.

Guy Lever, François Laviolette, and John Shawe-Taylor. Tighter PAC-Bayes bounds through distribution-dependent priors. *Theoretical Computer Science*, 473:4–28, 2013.

Jialin Liu, Xiaohan Chen, Zhangyang Wang, Wotao Yin, and HanQin Cai. Towards constituting mathematical structures for learning to optimize. In *International Conference on Machine Learning*, pages 21426–21449. PMLR, 2023.

Ben London. A PAC-Bayesian analysis of randomized learning with application to stochastic gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

David McAllester. Simplified PAC-Bayesian margin bounds. In *Learning theory and Kernel machines*, pages 203–215. Springer, 2003a.

David McAllester. PAC-Bayesian stochastic model selection. *Machine Learning*, 51(1):5–21, 2003b.

Luke Metz, Niru Maheswaranathan, Jeremy Nixon, Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4556–4565. PMLR, 2019.

Luke Metz, C Daniel Freeman, James Harrison, Niru Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. In *Conference on Lifelong Learning Agents*, pages 142–164. PMLR, 2022.

Michael Moeller, Thomas Mollenhoff, and Daniel Cremers. Controlling neural networks via energy dissipation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3256–3265, 2019.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983.

Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

Yuki Ohnishi and Jean Honorio. Novel change of measure inequalities with applications to PAC-Bayesian bounds and Monte Carlo estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 1711–1719. PMLR, 2021.

Emilio Parrado-Hernández, Amiran Ambroladze, John Shawe-Taylor, and Shiliang Sun. PAC-Bayes bounds with data dependent priors. *Journal of Machine Learning Research*, 13(1):3507–3531, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Fabian Pedregosa and Damien Scieur. Average-case acceleration through spectral density estimation. In *International Conference on Machine Learning*, pages 7553–7562. PMLR, 2020.

María Pérez-Ortiz, Omar Rivasplata, John Shawe-Taylor, and Csaba Szepesvári. Tighter risk certificates for neural networks. *Journal of Machine Learning Research*, 22(227): 1–40, 2021.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Omar Rivasplata, Ilja Kuzborskij, Csaba Szepesvári, and John Shawe-Taylor. PAC-Bayes analysis beyond the usual bounds. *Advances in Neural Information Processing Systems*, 33:16833–16845, 2020.

Borja Rodríguez-Gálvez, Ragnar Thobaben, and Mikael Skoglund. More PAC-Bayes bounds: From bounded losses, to losses with general tail behaviors, to anytime validity. *Journal of Machine Learning Research*, 25(110):1–43, 2024.

Ernest Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. In *International Conference on Machine Learning*, pages 5546–5557. PMLR, 2019.

Damien Scieur and Fabian Pedregosa. Universal average-case optimality of Polyak momentum. In *International Conference on Machine Learning*, pages 8565–8572. PMLR, 2020.

Matthias Seeger. PAC-Bayesian generalisation error bounds for Gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Stochastic convex optimization. In *COLT*, volume 2, page 5, 2009.

Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11:2635–2670, 2010.

Suhas Sreehari, S Venkat Venkatakrishnan, Brendt Wohlberg, Gregery T Buzzard, Lawrence F Drummy, Jeffrey P Simmons, and Charles A Bouman. Plug-and-play priors for bright field electron tomography and sparse interpolation. *IEEE Transactions on Computational Imaging*, 2(4):408–423, 2016.

Michael Sucker, Jalal Fadili, and Peter Ochs. Learning-to-optimize with PAC-Bayesian guarantees: Theoretical considerations and practical implementation. *arXiv preprint arXiv:2404.03290*, 2024.

Yu Sun, Brendt Wohlberg, and Ulugbek S Kamilov. An online plug-and-play algorithm for regularized image reconstruction. *IEEE Transactions on Computational Imaging*, 5(3): 395–408, 2019.

Afonso M Teodoro, José M Bioucas-Dias, and Mário AT Figueiredo. Scene-adapted plug-and-play algorithm with convergence guarantees. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2017.

Matthieu Terris, Audrey Repetti, Jean-Christophe Pesquet, and Yves Wiaux. Enhanced convergent pnp algorithms for image restoration. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1684–1688. IEEE, 2021.

Niklas Thiemann, Christian Igel, Olivier Wintenberger, and Yevgeny Seldin. A strongly quasiconvex PAC-Bayesian bound. In *International Conference on Algorithmic Learning Theory*, pages 466–492. PMLR, 2017.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Tom Tirer and Raja Giryes. Image restoration by iterative denoising and backward projections. *IEEE Transactions on Image Processing*, 28(3):1220–1234, 2018.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95, 2002.

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *International conference on machine learning*, pages 3751–3760. PMLR, 2017.

Zhonglin Xie, Wotao Yin, and Zaiwen Wen. Ode-based learning to optimize. *arXiv preprint arXiv:2406.02006*, 2024.

Bo Xin, Yizhou Wang, Wen Gao, David Wipf, and Baoyuan Wang. Maximal sparsity with deep networks? *Advances in Neural Information Processing Systems*, 29, 2016.

Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.

Appendix H. Architecture for the Experiment on Quadratic Functions

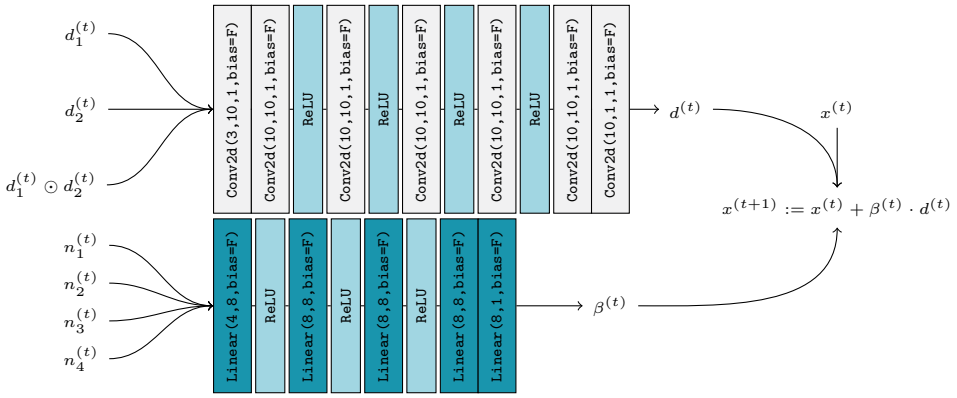

Figure 10: Update step of $\mathcal{A}$ for quadratic problems: The directions $d_1^{(t)}$, $d_2^{(t)}$ and $d_1^{(t)} \odot d_2^{(t)}$ are inserted as different channels into the `Conv2d`-block, which performs 1×1 "convolutions", that is, the algorithm acts coordinate-wise on the input. The scales $n_1^{(t)}$, ..., $n_4^{(t)}$ get transformed separately by the fully-connected block.

Here, the step-size $\beta^{(t)}$ is computed by a fully-connected block (without bias) and ReLU activation functions based on the the inputs $n_1^{(t)} = \log(1 + \|\nabla \ell(\xi^{(t)}, \theta)\|)$, $n_2^{(t)} = \log(1 + \|x^{(t)} - x^{(t-1)}\|)$, $n_3^{(t)} = \log(1 + \ell(\xi^{(t)}, \theta))$ and $n_4^{(t)} = \log(1 + \ell(\xi^{(t-1)}, \theta))$. On the other hand, the direction $d^{(t)}$ is computed by a 1×1-convolutional block, where the input-channels are given by the normalized gradient $d_1 := \frac{\nabla \ell(\xi^{(t)}, \theta)}{\|\nabla \ell(\xi^{(t)}, \theta)\|}$, the normalized momentum term $d_2 := \frac{x^{(t)} - x^{(t-1)}}{\|x^{(t)} - x^{(t-1)}\|}$, and their pointwise product $d_1 \odot d_2$, and we use 10 channels in each hidden layer.

Appendix I. Architecture for the Image Processing Experiment

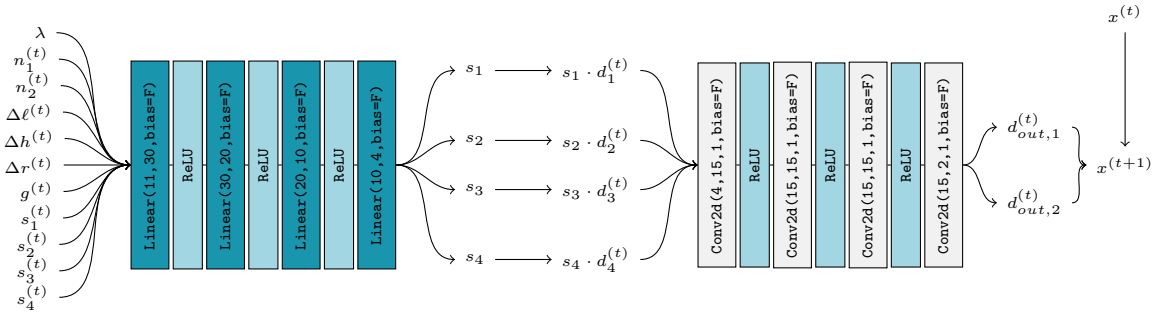

Figure 11: Algorithmic update for the image processing problem: Based on the given eleven features, the first block computes four weights $s_1, ..., s_4$, which are used to perform a weighting of the different directions $d_1^{(t)}, ..., d_4^{(t)}$, which are used in the second block. This second block consists of a 1x1-convolutional blocks, which computes two update direction $d_{out,1}^{(t)}$ and $d_{out,1}^{(t)}$. Then, we update $x^{(t+1)} := x^{(t)} + \frac{\|\nabla \ell(x^{(t)},\theta)\|}{L} d_{out,1}^{(t)} - \frac{1}{L}\nabla \ell(x^{(t)},\theta) + \|x^{(t)} - x^{(t-1)}\| d_{out,2}^{(t)}$.

The update of the learned algorithm reads:

$$x^{(t+1)} := x^{(t)} + \frac{\|\nabla \ell(x^{(t)},\theta)\|}{L} d_{out,1}^{(t)} - \frac{1}{L}\nabla \ell(x^{(t)},\theta) + \|x^{(t)} - x^{(t-1)}\| d_{out,2}^{(t)} \,.$$

Here, the directions $d_{out,1}^{(t)}$ and $d_{out,2}^{(t)}$ are predicted by a 1x1 convolutional block with ReLU-activation functions. These are predicted based on the reweighted directions $d_1^{(t)}, ..., d_4^{(t)}$, which are given as the normalized gradient, the normalized momentum, the normalized gradient of the data-fidelity term, and the normalized gradient of the regularization term. Here, the weights $s_1, ..., s_4$ for the reweighting are predicted by a fully-connected block with ReLU-activation functions based on the features $n_1^{(t)} = \log(1 + \|\nabla \ell(\xi^{(t)},\theta)\|)$, $n_2^{(t)} = \log(1 + \|x^{(t)} - x^{(t-1)}\|)$, $\Delta\ell^{(t)} := \ell(x^{(t)},\theta) - \ell(x^{(t-1)},\theta)$, $\Delta r^{(t)} := r(x^{(t)},\theta) - r(x^{(t-1)},\theta)$, where r is the regularization term, $\Delta h^{(t)} := h(x^{(t)},\theta) - h(x^{(t-1)},\theta)$, where h is the data-fidelity term, $g^{(t)} := \max_{i=1,...,n} |\nabla \ell(\xi^{(t)},\theta)|_i$, the scalarproducts $s_1^{(t)}, ..., s_4^{(t)}$ between the (normalized) gradient and the (normalized) momentum, between the (normalized) gradient of the regularization term and the (normalized) momentum, between the (normalized) gradient of the data-fidelity term and the (normalized) momentum, between the (normalized) gradient of the regularization term and the (normalized) gradient of the data-fidelity term, and, finally, the regularization parameter λ.

Appendix J. Architecture for the LASSO Experiment

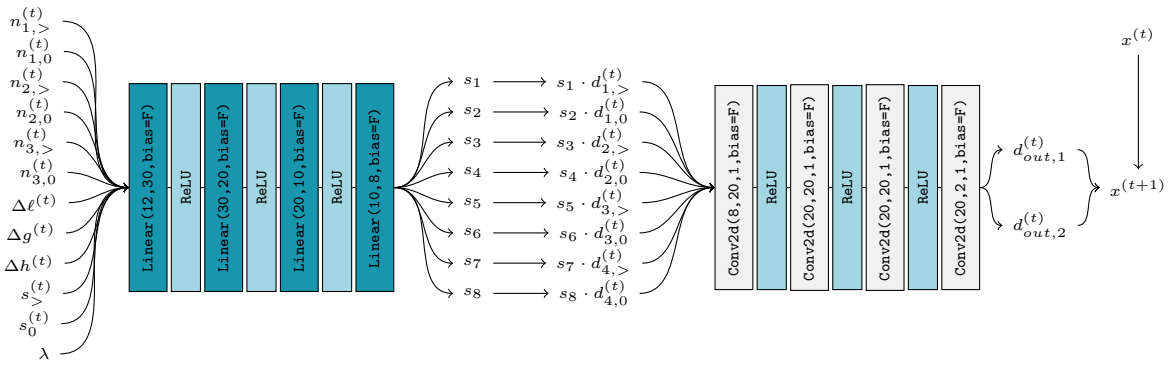

Figure 12: Algorithmic update for the LASSO problem: Based on the given twelve features, the first block computes eight weights, which are used to perform a weighting of the different directions, which are used in the second block. This second block then predicts two directions $d_{out,1}, d_{out,2}$, where $d_{out,1}$ only acts on the non-zero entries, and $d_{out,2}$ acts on the zero entries. These are used in the update $x^{(t+1)} := \mathrm{prox}_{\beta g}\left(x^{(t)} + \left(d^{(t)}_{out,1,>} - \nabla\ell(x^{(t)}) + \|x^{(t)} - x^{(t-1)}\| \cdot d^{(t)}_{out,2,0}\right)/L\right)$.

Since in the LASSO problem the algorithm has to identify the support of the solution, that is, the coordinates which are non-zero, we also treat the zero and non-zero entries of $x^{(t)}$ (and derived quantities) separately. Here, we denote the non-zero entries by $x^{(t)}_{>}$ and the zero entries by $x^{(t)}_{0}$, and similarly for all other quantities. First, we compute weights $s_1, ..., s_8$ with a fully-connected block with ReLU-activation functions. The used features are $n^{(t)}_1 = \log(1+\|\nabla\ell(\xi^{(t)}, \theta)\|)$, $n^{(t)}_2 = \log(1+\|x^{(t)} - x^{(t-1)}\|)$, $n^{(t)}_3 = \log(1+\|p^{(t)}\|)$, where $p^{(t)} = \mathrm{prox}_{\beta g}\left(x^{(t)} - \beta\nabla\ell(x^{(t)}, \theta)\right)$, $\Delta\ell^{(t)} := \ell(x^{(t)}, \theta) - \ell(x^{(t-1)}, \theta)$, $\Delta g^{(t)} := g(x^{(t)}) - g(x^{(t-1)})$, $\Delta h^{(t)} := h(x^{(t)}, \theta) - h(x^{(t-1)}, \theta)$, the scalar products $s^{(t)}_{>}$ and $s^{(t)}_{0}$ between the (normalized) gradient and (normalized) momentum, and the regularization parameter λ. Then, these weights are used to perform a reweighting of the given directions $d^{(t)}_1, ..., d^{(t)}_4$, given by the normalized gradient, the normalized momentum, the normalized residual $x^{(t)} - p^{(t)}$, and the coordinate-wise product between (normalized) gradient and (normalized) momentum. Afterwards, these reweighted directions get fed into a 1x1-convolutional block, which predicts the two directions $d^{(t)}_{out,1}$ and $d^{(t)}_{out,2}$, which are used to compute the final update with the proximal mapping, given by

$$x^{(t+1)} := \mathrm{prox}_{\beta g}\left(x^{(t)} + \left(d^{(t)}_{out,1,>} - \nabla\ell(x^{(t)}) + \|x^{(t)} - x^{(t-1)}\| \cdot d^{(t)}_{out,2,0}\right)/L\right).$$

Appendix K. Architecture for Training the Neural Network

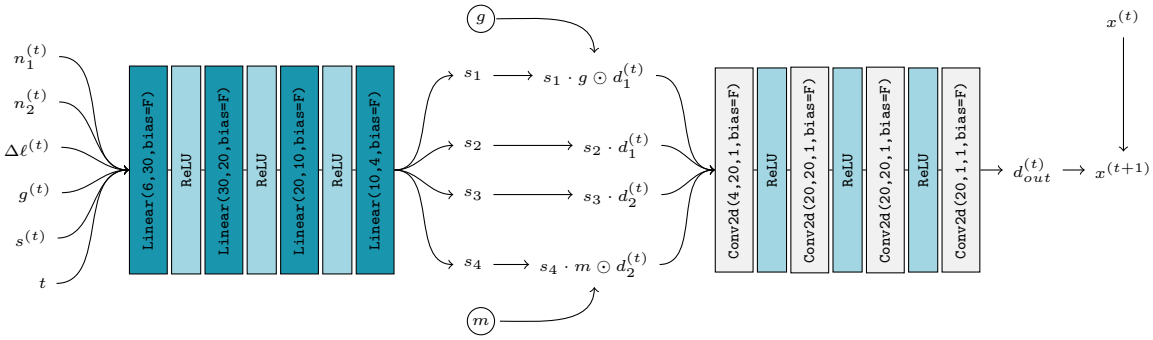

Figure 13: Algorithmic update for training the neural network: Based on the given six features, the first block computes four weights $s_1, ..., s_4$, which are used to perform a weighting of the different directions $g \odot d_1^{(t)}$, $d_1^{(t)}$, $d_2^{(t)}$, $m \odot d_2^{(t)}$, which are used in the second block. This second block consists of a 1x1-convolutional blocks, which compute an update direction $d_{out}^{(t)}$. Then, we update $x^{(t+1)} := x^{(t)} + d_{out}^{(t)}$.

To compute the weights $s_1, ..., s_4$ with the first block, we use the features $n_1{}^{(t)} = \log(1 + \|\nabla\ell(\xi^{(t)}, \theta)\|)$, $n_2{}^{(t)} = \log(1 + \|x^{(t)} - x^{(t-1)}\|)$, $\Delta\ell^{(t)} := \ell(x^{(t)}, \theta) - \ell(x^{(t-1)}, \theta)$, $g^{(t)} := \max_{i=1,...,n} |\nabla\ell(x^{(t)}, \theta)_i|$, the scalar product $s^{(t)}$ between the (normalized) gradient and the (normalized) momentum, and the iteration counter t. Then, these weights are used to perform a weighting of the directions $d_1^{(t)}, d_2^{(t)}, g \odot d_1^{(t)}$ and $m \odot d_2^{(t)}$, where g and m are additional learned vectors of size n, which we use as diagonal preconditioning. Here, $d_1^{(t)}$ is the normalized gradient and $d_2^{(t)}$ is the normalized momentum term. These directions get fed into a 1×1-convolutional block, which predicts the direction $d_{out}^{(t)}$ that is used to update the iterate as $x^{(t+1)} = x^{(t)} + d_{out}^{(t)}$.

Appendix L. Architecture for Stochastic Empirical Risk Minimization

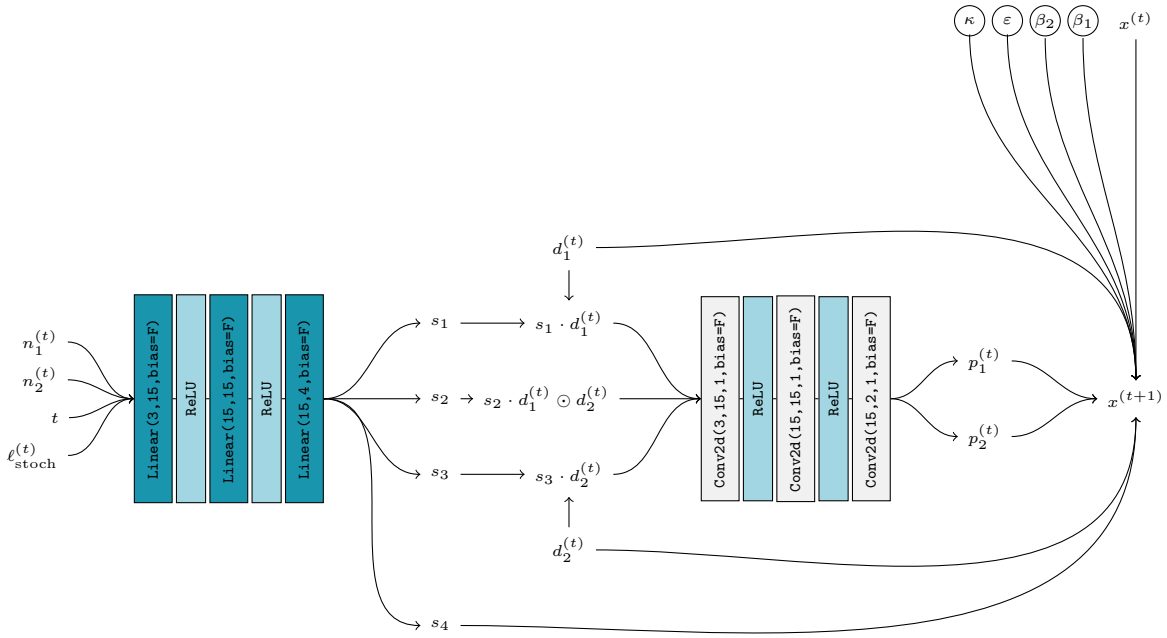

Figure 14: Algorithmic update for stochastic empirical risk minimization: Based on the given four features, the first block computes four weights $s_1, ..., s_4$, where $s_1, ..., s_3$ are used to perform a weighting of the different directions $d_1^{(t)}$, $d_2^{(t)}$, and $d_1^{(t)} \odot d_2^{(t)}$, which get fed into the second block, and s_4 is used afterwards as a step-size. The second block consists of a 1x1-convolutional layers, that is, they act coordinate-wise, and computes two vectors $p_1^{(t)}$, $p_2^{(t)}$, which are used as diagonal "preconditioners" for the update of Adam.

First, we compute the same features as Adam, that is, $m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1)\hat{\nabla}\ell(x^{(t)}, \theta)$, $v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)\hat{\nabla}\ell(x^{(t)}, \theta) \odot \hat{\nabla}\ell(x^{(t)}, \theta)$, where $\hat{\nabla}\ell(x^{(t)}, \theta)$ denotes the stochastic gradient. Then, as for Adam, we set $\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1^t}$ and $\hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^t}$, and split $\hat{m}^{(t)}$ and $\hat{v}^{(t)}$ into the corresponding (logarithmically transformed) norms $n_1^{(t)}$, $n_2^{(t)}$ and unit-vectors $d_1^{(t)}$, and $d_2^{(t)}$, respectively. The norms, together with the current (stochastic) loss and the iteration counter t, get fed into a fully-connected block to compute four weights $s_1, ..., s_4$. Then, s_4 is used as a step-size in the final update, while $s_1, ..., s_3$ get used to weigh the vectors $d_1^{(t)}$, $d_2^{(t)}$, and $d_1^{(t)} \odot d_2^{(t)}$ before they get fed into the 1×1-convolutional block, which outputs vectors p_1 and p_2. These, in turn, are used as a kind of diagonal preconditioner for the update of Adam, that is, the final output is given by the formula $x^{(t)} = x^{(t-1)} - (s_4 \cdot \kappa \cdot d_1 \odot \hat{m}^{(t)})/(0.001 \cdot |d_2| \odot \sqrt{\hat{v}^{(t)}} + \varepsilon)$. Here, the constant 0.001 is just there to stabilize training in the beginning, and for the other constants we use the default values in PyTorch, that is, we set $\kappa = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1 \cdot 10^{-8}$.