Dissertation ETH Zürich No 29641.

Bayesian optimization in the wild: risk-averse and computationally-effective decision-making

A dissertation submitted to attain the degree of

Doctor of Sciences of ETH Zürich

(Dr. Sc. ETH Zürich)

presented by

Anastasiia Makarova

M. Sc. in Applied Mathematics and Physics,
Moscow Institute of Physics and Technology (MIPT)

born 7 September 1994

accepted on the recommendation of
Prof. Dr. Andreas Krause, examiner
Prof. Dr. Ilija Bogunovic, co-examiner
Prof. Dr. Fernando Perez-Cruz, co-examiner

2023

ETH Zürich
LAS - Learning and Adaptive Systems Lab
Institute for Machine Learning, CAB, Universitaetstrasse 6
8092 Zurich, Switzerland

to my family, who are my life teachers and my infinite support

Acknowledgements

I am deeply grateful to those who have crossed paths with me and influenced this journey.

My advisor, Professor Andreas Krause, from whom I have been fortunate to receive support and guidance. I am grateful for this mentorship and contribution to my PhD path, from introducing me to Bayesian optimization to this manuscript. I am grateful to my committee, Professor Ilija Bogunovic, and Professor Fernando Perez-Cruz, for their thorough feedback on my thesis.

Working with my ETH collaborators has been a highlight of my journey. I am grateful to Matteo, for the support and guidance in my early PhD years; Erik, for exploring and mastering a new area together; Ilnura, for courageously getting into challenging ideas; Ilija, for setting high standards; Misha, for never-ending drive to make things (finally!) work; and Scott, for being a great research team (and trailrunning team!) later in my PhD. I am grateful for these collective efforts that have been truly rewarding for me.

My gratitude extends to my LAS group colleagues for the vibrant memories we have created. Thank you for our morning Zurich rides, coffee on the stairs (and everywhere!), cooking and eating, and long and fulfilling conversations – I will keep them in my heart. I will also remember our CAB office, thanks to Mohammad, Mojmir, and Bhavya, as an important place where we have all grown a bit. And, of course, thank you, Rita, for the constant care about us – without your support, none of this would have been possible.

Outside of ETH, my summers at Google DeepMind and Amazon were both enlightening and inspiring. Earlier in my PhD, at Amazon, I witnessed our Bayesian optimization methods applied in real-world scenarios, and it sparked new ideas for my PhD research. I am thankful to my host, Hubin, and my team for their collaboration and ongoing support (which continued even after my internship!). Later, at Google DeepMind, I worked on reinforcement learning for LLM alignment and it was an exciting experience shaping my post-PhD path. I am grateful to my hosts, Olivier and Andrea, and the whole team, for their contributions to my learning journey.

The people at ETH – what an array of bright and open-minded individuals I have met at our CAB building and ML institute! Particularly Octavian, Dario, Antonio, Rita, Pesho, Alina, Misha, and Anna, – your presence has made the ETH campus full of life. Teaching and supervising Master's students was undoubtedly a boost to my growth — I learned so much from them. So, I extend my thanks to the students I co-supervised, including Erik, Ankit, Alicja, Stefan, as well as students in our courses, for their curiosity and engagement.

Outside of research, I am luckily surrounded by beautiful people. My friends in Switzerland, thank you for making this place feel like home, and my friends worldwide, thank you for maintaining a sense of connection and support despite the distances.

I am deeply grateful to my family for their unwavering support and belief in me, particularly my parents, who have set an extraordinary example of dedication, and my brother, a dear partner in outdoor adventures and hours-long conversations.

And finally, my deepest gratitude goes to my loving husband, Oleg. A person with a big heart who always finds a silver lining in the darkest times. Thank you for filling our life with happiness. This thesis, and so much more, would not have been possible without you.

Abstract

Sequential decision-making in some complex and uncertain environments can be formalized as optimizing a black-box function. For example, in drug design, the aim is to maximize compound efficiency by sequentially searching through the vast space of molecules, or, in tuning a particle accelerator, the aim is to maximize the particle beam energy. Interactions with these environments can be costly or time-consuming, demanding decision-making systems to work in small-data regimes. *Bayesian optimization* (BO) is a powerful data-driven framework for global optimization that adaptively chooses actions to evaluate. Its potential for impactful real-world applications is vast, from those mentioned above to widespread automated machine learning (ML) services. There are, however, practical challenges that impede its implementation or result in sub-optimal decisions.

This dissertation aims to advance the applicability of BO by addressing three crucial points: *risk-aversion*, which ensures performance beyond the average only; *query-effectiveness*, which aims at smart use of budgets covering evaluation costs; and *problem-adaptiveness*, which leverages the structure of the decision space. To this end, we propose novel approaches and examine them theoretically and empirically in real scenarios.

First, we tackle balancing *high utility and low risk* – a serious issue in high-stakes applications. For example, the accelerator's maximum pulse energy must be stable to observe chemical reactions; or in drug discovery, the drug must succeed for all individuals. Our approach trades off mean and input-dependent aleatoric uncertainty (both learned on the fly during optimization) and provides theoretical sample complexity results. Moreover, our empirical study on tuning a particle accelerator and ML models shows the benefit of the approach.

Second, we aim at efficient budget allocation, enriching the practical performance heavily dependent on the interaction cost and the available budget covering these costs. For example, the success of a timely drug discovery depends on the smart use of the resources at hand. We tackle two crucial questions of *query efficiency* in BO: (1) how to use cheaper but less accurate evaluations, and (2) when to stop the optimization. We leverage multi-fidelity optimization and incorporate the evaluation costs in a natural information-theoretic manner. Moreover, our automatic termination approach for the BO loop determines the minimal budget required for obtaining a high-quality solution.

Finally, the complex and *non-continuous nature of the decision variables* can make applying Bayesian optimization in areas such as drug design or automated ML difficult. Our approach for mixed-variable BO exploits the structure and interconnections within the discrete subdomain, learning them on the fly during the optimization.

Our approaches make BO applicable to a wider range of critical applications while combining practical simplicity with theoretically grounded reasoning.

Keywords: Bayesian optimization, Gaussian process, Regret bounds, Multi-fidelity optimization, Risk-averse optimization, AutoML

Kurzfassung

Sequentielle Entscheidungsprozesse in komplexen und unsicheren Umgebungen lassen sich als Optimierung einer Black-Box-Funktion beschreiben. Zum beispiel, bei der Justierung eines Teilchenbeschleunigers ist es das Ziel, die Energie des Teilchenstrahls zu maximieren. Die Interaktion mit diesen Umgebungen kann kosten- und zeitintensiv sein, sodass von Entscheidungssystemen erwartet wird, dass sie in datenarmen Szenarien funktionieren. Die *Bayessche Optimierung* (BO) bietet einen effektiven datengetriebenen Rahmen für globale Optimierungsverfahren, welcher adaptiv Handlungen zur Bewertung auswählt. Ihr Potenzial für praktische Anwendungen in der realen Welt ist enorm, von den bereits genannten Beispielen bis zu weit verbreiteten automatisierten Machine-Learning-Dienstleistungen. Dennoch bestehen praktische Herausforderungen, die ihre Umsetzung behindern oder zu suboptimalen Entscheidungen führen können.

Das Ziel dieser Dissertation ist es, die Anwendbarkeit der BO zu erweitern, indem drei Schlüsselaspekte verbessert werden: *Risikoaversion*, die eine Leistung über dem Durchschnitt sicherstellt; *Abfrageeffizienz*, die eine kluge Nutzung von Budgets zur Deckung der Evaluierungskosten anstrebt; und *Problemanpassungsfähigkeit*, die sich die Struktur des Entscheidungsraums zu Nutze macht. Wir schlagen in dieser Arbeit neue Verfahren vor und prüfen diese sowohl theoretisch als auch empirisch in realen Szenarien.

Erstens befassen wir uns mit dem Gleichgewicht von hohem Nutzen und geringem Risiko, welches insbesondere in hochriskanten Anwendungen von Bedeutung ist. Die maximale Pulsenergie eines Beschleunigers muss beispielsweise stabil sein, um chemische Reaktionen beobachten zu können, und bei der Arzneimittelentdeckung sollte das Medikament für jeden Einzelnen erfolgreich sein. Unser Ansatz findet ein Gleichgewicht zwischen Mittelwert und eingangsabhängiger aleatorischer Unsicherheit, die beide während der Optimierung erlernt werden, und liefert theoretische Ergebnisse zur Stichprobenkomplexität. Darüber hinaus zeigt unsere empirische Untersuchung bei der Justierung von Teilchenbeschleunigern und ML-Modellen den Vorteil dieses Ansatzes.

Zweitens zielen wir auf eine effiziente Budgetverteilung ab, wobei die tatsächliche Leistung stark von den Interaktionskosten und dem zur Deckung dieser Kosten verfügbaren Budget abhängt. So hängt der Erfolg einer rechtzeitigen Medikamentenentdeckung maßgeblich von der intelligenten Nutzung der verfügbaren Ressourcen ab. Wir stellen uns zwei zentralen Fragen zur Abfrageeffizienz in BO: (1) Wie kann man kostengünstigere, aber weniger genaue Evaluierungen verwenden? und (2) Wann sollte die Optimierung beendet werden? Wir nutzen Multi-Fidelity-Optimierung und integrieren die Evaluierungskosten auf eine natürliche, informationstheoretische Weise. Darüber hinaus bestimmt unser automatisierter Abschlussansatz für den BO-Zyklus das für eine qualitativ hochwertige Lösung erforderliche Mindestbudget.

Abschließend kann die komplexe und unstetige Natur der Entscheidungsvariablen die Anwendung der Bayesianischen Optimierung in Bereichen wie Arzneimitteldesign oder automatisiertem ML erschweren. Unser Ansatz für die gemischte BO nutzt Struktur und Zusammenhänge im diskreten Bereich, die während der Optimierung erlernt werden.

Unsere Ansätze erweitern die Anwendbarkeit der BO auf eine breitere Palette kritischer Anwendungen und verbinden praktikable Einfachheit mit theoretisch fundierter Argumentation.

The results enclosed in this dissertation are from previously published work, which are products of joint work with Ilnura Usmanova, Erik Daxberger, Matteo Turchetta, Ilija Bogunovic, Huibin Shen, Valerio Perronne, Aaron Klein, Jean Baptiste Faddoul, Matthias Seeger, Cedric Archambeau, Stefan Beyeler, Sebastian Hörl and Andreas Krause. Although this dissertation carries my name, this research would not have been possible without the contribution of these collaborators.

Contents

Principal Notation and Acronyms

General

$\boldsymbol{x}$	Vector (bold italic letters)	
X	Matrix (uppercase letters)	
$\mathbb{R}^d$	Real coordinate space of dimension d	
$\mathbb{R}_+$	Non-negative orthant	
$[m]$	Set of all non-negative integers up to integer m	
$[m]_+$	Set of all positive integers up to integer m	
$\mathbb{P}\{\mathcal{A}\}$	Probability of event $\mathcal{A}$	
$\mathbb{E}(\cdot)$	Expectation of a random variable	
$\mathrm{Cov}(\cdot,\cdot)$	Covariance between two real-valued random variables	
$y \sim \pi$	Random variable y with distribution π	
$D_{KL}(\cdot\|\|\cdot)$	Kullback–Leibler divergence between probability measures	
$\mathcal{N}$	Natural numbers, not including zero	
$\mathbb{Z}$	Integers	
$\mathbb{R}$	Real numbers	
$\langle\cdot,\cdot\rangle$	Inner product between vectors in a Hilbert space	100
$\|\cdot\|$	ℓ_2-norm (Euclidean norm)	100
$O(\cdot)$	Big O complexity	
$\tilde{O}(\cdot)$	Big O up to a multiplicative logarithmic factor	
$\Omega(\cdot)$	Big Ω notation	

Specific

$\mathcal{X}$	Input set	8
$\mathcal{D}$	Data set	8
$\mathcal{L}(\cdot,\cdot)$	Lagrangian function	99
λ	Lagrange dual vector	99
ξ	Random noise	8
ϵ	Accuracy	24
$\kappa(\cdot,\cdot)$	Kernel function	12
$\mathcal{F}$	Function space	
$\mathcal{H}$	Hilbert space	100
$\mathcal{H}_\kappa$	Reproducing kernel Hilbert space (RKHS) induced by kernel function κ	101
$\|f\|_\kappa$	RKHS norm of a function	

$\mathbf{I}$	Identity matrix	
R_t	Cumulative regret	24
r_t	Simple regret	24
$H[\,\cdot\,]$	Entropy	17
$I(\cdot,\cdot)$	Information gain	17
γ_t	Maximum information gain	18

Gaussian processes

$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean vector μ and covariance matrix Σ
$\mathcal{GP}(\mu, \kappa)$	Gaussian process with mean function μ and kernel function Σ

Acronyms and Abbreviations

General

BO	Bayesian optimization	18
GP	Gaussian process	12
SGD	Stochastic gradient descent	
RL	Reinforcement learning	25
MAB	Multi-armed bandit	25
RKHS	Reproducing kernel Hilbert space	101
MDP	Markov decision process	25
NN	Neural network	
MAP	Maximum a posteriori	19, 71

Methods

RAHBO	Risk-averse Heteroscedastic Bayesian optimization	34
MiVaBo	Mixed-variable Bayesian optimization	81
MFMES	Multi-fidelity max-value entropy search	65
GP-UCB	Gaussian process upper confidence bound	22
EI	Expected improvement	22
PI	Probability of improvement	22
MES	Max-value entropy search	23
TS	Thompson sampling	24
VAE	Variational Autoencoder	90

For clarity, some acronyms are defined when they appear in the dissertation for the first time.

CHAPTER 1

Introduction

1.1 Motivation and goals

Systems that interact with complex and uncertain environments, actively learn them, and incorporate the acquired data for decision-making have experienced rapid development over the last decade. Furthermore, machine learning applications in domains with significant social impact are experiencing significant growth. For example, in drug discovery, recent advancements in protein folding have the potential to accelerate the process greatly. Some other vital examples of active learning would include materials discovery [Kor+19; GH20], genetics [Gon+15; Mos+20] or reasoning tasks where we look for explanations for the data, as well as causal discovery underlying data generation. This highlights the need for theoretically grounded and widely practically applicable methods.

Examples such as drug discovery with navigating intricate space of molecules illustrate problems with a clear optimization goal, e.g., maximizing drug efficiency. However, the objective function that maps from the molecules' space to drug efficiency is a *black box* that can only be accessed through observations (see Figure 1.1). In these scenarios, a learner sequentially interacts with the unknown objective function, observes a *noise-perturbed* evaluation, and then navigates the interaction further. The objective function is *expensive to evaluate*, and a constrained budget requires this sequential decision-making to be *data-efficient*. Searching for optimal decision, the learner must balance exploring the unknown with the exploitation of what is already known, the so-called *exploration-exploitation* dilemma. The dilemma arises because the learner makes decisions under *uncertainty*: actions increasing knowledge do not necessarily lead to the optimal function values.

One prominent framework for addressing the uncertainty in learning problems is *Bayesian inference*. The main idea behind Bayesian inference is to model the unknown quantity of interest using probability distribution and update it as we observe evaluations. This results in conditional probabilities defining the relationship between the unknown value and observations and allows for predicting the unobserved values while quantifying and propagating the predictions' uncertainty.

This dissertation contributes to *Bayesian optimization* – a powerful tool for global optimization of an unknown costly black-box function. Bayesian optimization adaptively chooses actions in a query-efficient manner to find the optimal one by adopting the Bayesian perspective. The performance of such sequential optimization can be evaluated

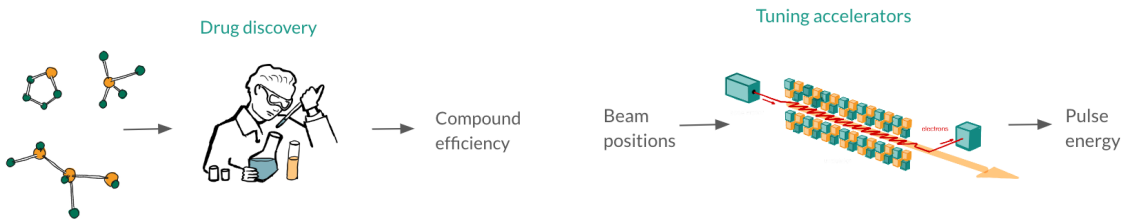


Figure 1.1: Examples of black-box optimization. In drug discovery (left), we want to navigate the intricate space of molecules to maximize drug efficiency. The objective function that maps from the molecules' space to drug efficiency is a black box, i.e., it can only be accessed through observations. Another example is tuning complex systems, such as an accelerator: to maximize some objective, e.g., the pulse energy, we can vary the parameters, e.g., the beam positions, and after observing the objective realizations, choose the next promising parameter candidates.

by how quickly the queried decisions converge to an optimal one. It is measured by *regret*, which is a suboptimality in the function value by virtue of not knowing the optimum in advance. Non-zero regret is inevitable since the learner needs to take non-optimal actions to learn the unknown; however, in the long run, we are interested in the convergence rate guarantees for the accumulated regret. Thereby, the performance of Bayesian optimization methods is studied from two perspectives: empirical performance and theoretical guarantees. Bayesian optimization is applied to many scientific domains, from robotics [BKS16; Cal+16; Mar+16] to hyperparameter tuning of complex learning systems [Kir+19; Che+18; SLA12a]. However, many important applications, such as materials design and drug discovery, are out of the reach of standard approaches or suffer from suboptimal decisions. This happens due to the complex domain structure of the applications and real-world solution requirements that current methods do not consider.

The goals of this dissertation are threefold, addressing three crucial requirements for Bayesian optimization: *(i) risk-aversion*, which ensures performance beyond the average only; *(ii) query-effectiveness*, which aims at smart use of budgets available for function evaluations; and *(iii) problem- adaptiveness*, which leverages the structure of the decision space. To this end, we propose novel approaches and examine them theoretically and empirically in real-world scenarios. Our contributions toward the first goal include considering input-dependent aleatoric uncertainty studied in Chapter 3. Our contributions toward the second goal include the automatic termination of the Bayesian optimization loop and the multi-fidelity approach studied in Chapter 4. Finally, our contributions toward the third goal include the Bayesian optimization over domains with discrete and continuous variables, i.e., mixed-variable domains, studied in Chapter 5. Following these, Chapter 6 concludes the dissertation and discusses the potential for future work. Below, we provide a more detailed thesis outline covering the contributions.

1.2 Outline and contributions

In the following, we present a summary and outline of each chapter.

1.2.1 Chapter 2

In this chapter, we provide the necessary background related to our work. In particular, we briefly introduce Bayesian optimization and its main components, such as probabilistic modeling, acquisition functions, performance metrics, and convergence guarantees. To this end, we present two settings: (i) Bayesian setting with an assumption of a function sampled from a Gaussian process and (ii) frequentist setting with reproducing Hilbert space assumption. We discuss the relationship between both settings and how they are used in Bayesian optimization.

1.2.2 Chapter 3

In this chapter, we study how aleatoric uncertainty affects risk-averse decision-making. Black-box optimization tasks frequently arise in high-stakes applications such as drug and material discovery, genetics, robotics, and hyperparameter tuning of complex learning systems, to name a few. In many of these applications, there is often a trade-off between achieving high utility and minimizing risk. Moreover, uncertain and costly evaluations are an inherent part of black-box optimization tasks, and modern learning methods need to handle these aspects when balancing the previous two objectives. Classical Bayesian optimization approaches are typically *risk-neutral* as they only seek to optimize the expected function value. In practice, however, two solutions might attain similar expected function values, but one might produce significantly noisier realizations. This is of significant importance when it comes to deploying the found solutions. For example, when selecting hyperparameters of a machine learning algorithm, we might prefer configurations that lead to slightly higher test errors but, at the same time, lead to smaller variance, thus making them reliable to apply across different datasets sampled from the data distribution. In this chapter, we consider *risk-averse* Bayesian optimization.

Overview of contributions

- While the focus of standard Bayesian optimization approaches is mainly on trading-off exploration vs. exploitation and optimizing for the expected performance, in this work, we additionally focus on the risk involved when working with noisy objectives.
- We generalize Bayesian optimization to trade the mean and input-dependent variance of the objective, both of which we assume to be unknown a priori. We show that, in general, the solutions to risk-neutral and risk-averse settings do not coincide.
- We propose a novel risk-averse heteroscedastic Bayesian optimization algorithm (RAHBO). RAHBO aims to identify a solution with high return and low noise variance while learning the noise distribution on the fly.

- We theoretically study the convergence guarantees and demonstrate the effectiveness of RAHBO on both synthetic benchmarks and real hyperparameter tuning tasks.

1.2.3 Chapter 4

The efficiency with respect to the function queries is a crucial concern in practical settings, where data collection takes time and can be expensive. This chapter studies these cost-wise aspects of Bayesian optimization from two practical perspectives.

First, we study the automatic termination of the Bayesian optimization loop in a practical application of hyperparameter optimization of ML models (HPO). While the final performance after Bayesian optimization heavily depends on the provided budget, it is hard to pre-specify an optimal value in advance. In Section 4.1, we propose an effective and intuitive termination criterion for Bayesian optimization that automatically stops the procedure if it is sufficiently close to the global optimum. Our key insight is that the discrepancy between the true objective (predictive performance on test data) and the computable target (validation performance) suggests stopping once the statistical estimation error dominates the suboptimality in optimizing the target.

Second, we study the multi-fidelity setting where we can access the cheaper but noisier objective approximations correlated with the objective of interest. In Section 4.2.2, we show how the optimization can be adapted to parallel evaluations and accommodate different levels of approximations.

Overview of contributions

- We propose an effective and intuitive termination criterion for Bayesian optimization. We empirically study the approach across various real-world HPO problems and baselines. The results show that our method has a better trade-off between the test performance and optimization time.

- We propose an information-theoretic method for multi-fidelity Bayesian optimization. We empirically demonstrate its effectiveness in calibrating key parameters of an agent-based transport simulation and how to scale the method to higher dimensions.

1.2.4 Chapter 5

Traditionally, Bayesian optimization methods have focused on objectives with entirely *continuous* domains. More recently, the focus has expanded to address problems with solely *discrete* domains, such as those found in food safety monitoring and model sparsification within multi-component systems [BP18]. However, many real-world optimization problems spanning applied mathematics, engineering, and the natural sciences are of *mixed-variable* nature, involving *both* continuous and discrete input variables, and exhibit complex constraints. For example, tuning the hyperparameters of a convolutional neural network involves both continuous variables, e.g., learning rate and momentum, and discrete ones, e.g., kernel size, stride, and padding. In addition, these hyperparameters

impose validity constraints, e.g., combinations of kernel size, stride, and padding lead to invalid networks. Further examples of mixed-variable, potentially constrained optimization problems include sensor placement [KSG08], drug discovery [NFP11], a configuration of optimization solvers [HHL10], and many others. This work introduces an algorithm that can efficiently optimize mixed-variable functions subject to known constraints.

Overview of contributions

- We introduce MiVaBo, a novel algorithm for efficiently optimizing mixed-variable functions subject to known integer linear and quadratic constraints.
- We present two alternatives to optimize the resulting acquisition function that can incorporate known linear and quadratic constraints (Section 5.2.3). To our knowledge, this makes MiVaBo the first Bayesian optimization method to handle constraints over discrete variables.
- We provide the first convergence analysis of a mixed-variable Bayesian optimization algorithm.
- Finally, we demonstrate the effectiveness of MiVaBo on complex hyperparameter optimization tasks, such as deep generative model tuning, where it outperforms state-of-the-art methods and performs comparably to human expert tuning.

1.3 Publications

This thesis contains a selected collection of results from the studies as a Ph.D. candidate. The work presented in the thesis is published in the following series of papers.

[Mak+21a] "*Automatic Termination for Hyperparameter Optimization*", **A. Makarova**, Huibin Shen, Valerio Perrone, Aaron Klein, Jean Baptiste Faddoul, Andreas Krause, Matthias Seeger, Cedric Archambeau, International Conference on Automated Machine Learning (AutoML Conf), 2022;

[Mak+21b] "*Risk-averse Heteroscedastic Bayesian Optimization*", **A. Makarova**, I.Usmanova, I.Bogunovic, A. Krause, International Conference on Neural Information Processing Systems (NeurIPS), 2021;

[Dax+20] "*Mixed-Variable Bayesian Optimization*", E. Daxberger*, **A. Makarova***[1], M.Turchetta, A. Krause; International Joint Conference on Artificial Intelligence (IJCAI), 2020;

[Tur+18] "*Multi-fidelity Bayesian optimization for calibration of transport system simulations*", **A. Makarova**, M.Turchetta, S. Horl, S. Beyeler, A. Krause; 18th Swiss Transport Research Conference (STRC 2018)

[1]* indicates equal contribution.

Other publications

The following papers were published during the doctoral studies but are not included in the thesis.

[SMK23] "*Model-based Causal Bayesian Optimization*", S.Sussex, **A. Makarova**, A. Krause, International Conference on Machine Learning (ICLR), 2023;

[Sus+23] "*Adversarial Causal Bayesian Optimization*", S.Sussex, P.G.Sessa, **A. Makarova**, A. Krause, 2023;

[Koe+23] "*Safe Risk-averse Bayesian Optimization for Controller Tuning*", M. Ozols, C. Konig, **A. Makarova**, Efe C. Balta, A. Krause, A. Rupenyan, 2022;

[Usv+21] "*Cherry-Picking Gradients: Learning Low-Rank Embeddings of Visual Data via Differentiable Cross-Approximation*", M. Usvyatsov, **A. Makarova**, R. Ballester-Ripoll, M. Rakhuba, A. Krause, K. Schindler, International Conference on Computer Vision (ICCV), 2021;

[Dha+20] "*Hierarchical Image Classification using Entailment Cone Embeddings*", A. Dhall, **A. Makarova**, O. Ganea, D. Pavllo, M. Greeff, A. Krause, CVPR Workshop on Differential Geometry (DiffCVML), 2020.

CHAPTER 2

Background and Related Work

We study how one can actively learn about an unknown quantity of interest and optimize it. This chapter provides the necessary mathematical background and a review of the related literature.

In the first part of the chapter, we provide a short introduction to the machinery and intuitions relevant throughout the thesis. Section 2.1 covers Bayesian inference and types of uncertainties important for reasoning about the unknown. Section 2.2 introduces the Gaussian processes, a stochastic process widely used in statistical modeling of unknown functions. Section 2.3 covers the non-Bayesian approach, introduces reproducing kernel Hilbert spaces, a space of smooth functions that are possible to learn, and connects to the functions sampled from a Gaussian process with the same kernel. Section 2.4 discusses the notion of information capacity that measures the difficulty of learning the unknown function.

The second part of the chapter Section 2.5 describes how to solve global optimization problems using Bayesian optimization. We introduce the main building blocks of Bayesian optimization and the most common performance metrics. We give a summary of the main existing theoretical results and open challenges of the most relevance for this thesis. To start, let us state the general problem we are interested to tackle.

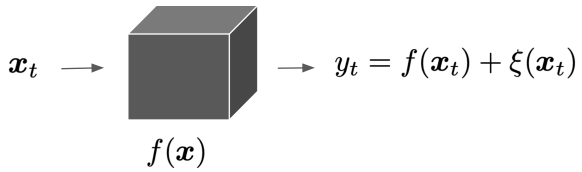


Figure 2.1: Illustration of the interaction with the unknown function at the iteration t. Throughout the thesis, we use variations of this visualization adapted to problems considered in the thesis chapters.

Problem Let $\mathcal{X}$ be a given compact set of inputs ($\mathcal{X} \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$). We consider a problem of sequentially interacting with an unknown objective function $f : \mathcal{X} \to \mathbb{R}$. Consider the example from the previous section, in which $f(\boldsymbol{x})$ represents the compound efficiency of some drug $\boldsymbol{x}$ from a vast space $\mathcal{X}$ of possible compounds. At every round t of this sequential procedure, the agent selects an action $\boldsymbol{x}_t \in \mathcal{X}$, and obtains an observation

perturbed by some, possibly input-dependent noise $\xi(\boldsymbol{x}_t)$, $\xi : \mathcal{X} \to \mathbb{R}$:

$$y_t = f(\boldsymbol{x}_t) + \xi(\boldsymbol{x}_t). \tag{2.1}$$

Figure 2.1 provides a schematic visualization of such an interaction. That results in observations denoted $\boldsymbol{y}_{1:t} = [y_1, \ldots, y_t]^\top$ and the dataset of pairs $\mathcal{D}_t = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_t, y_t)\}$.

As in the example, the agent aims to find the most effective drug compound, i.e., finding $\boldsymbol{x}$ that would maximize the unknown function f while learning f on the fly. Efficient solving of the learning problem requires further assumptions on the unknown function $f(\boldsymbol{x})$ and noise process $\xi(\boldsymbol{x})$: what can we say about the function f and its smoothness prior to observing evaluations? How much do we already know about the noise ξ, and how much should we learn?

When learning an unknown quantity from evaluations, we can consider two perspectives: Bayesian and frequentist approaches.[1] At their core, these two points of view differ in the base postulates and consider different classes of functions. The *Bayesian approach* assumes the unknown function to be *random* and operates with probabilities to describe the degree of belief. In contrast, the non-Bayesian, *frequentist*, approach assumes the function to be *fixed* and refers to probabilities as a limiting frequency, i.e., if we repeat the evaluation, then the probability is the frequency of the outcome. Both approaches have a notion of uncertainty allowing one to reason about the unknown quantity of interest. Though this uncertainty is based on different definitions of probability, the approaches share some common notions, such as confidence intervals, and have cross-connections, such as kernel regression and Gaussian process regression. In this thesis, we use Bayesian machinery for modeling unknown functions. This chapter examines approaches and conditions for Bayesian modeling and optimization.

We start with the *inference* problem, that is, estimating functional dependency based on the collected data, and in Section 2.5, we return to the optimization and introduce it formally.

2.1 Bayesian Inference

Bayesian inference is a fundamental framework to reason about the quantity of interest, i.e., the function f, based on data $\mathcal{D}_t$ by adopting a Bayesian perspective. At its core, it requires three ingredients. First, f is assumed to be random, and the learner encodes prior belief about f through a *prior* distribution over a given function space, $p(f)$. Second, the learner indicates how the observed data is related to the function f through the *likelihood* function, $p(\mathcal{D}_t|f)$. Finally, applying the Bayes theorem, the learner gets the *posterior* probability distribution over the function given the data as follows:

$$p(f|\mathcal{D}_t) = \frac{p(\mathcal{D}_t|f)p(f)}{p(\mathcal{D}_t)}. \tag{2.2}$$

[1] Which one to choose depends on the particular problem at hand and the philosophy of the learner. For a deeper understanding, we refer the reader to [Was10].

Bayesian inference offers a natural framework for the *regression*, i.e., estimating functional dependence between the action $\boldsymbol{x}$ and output y, and *decision* problems, i.e., deciding on the next actions $\boldsymbol{x}$ to be taken. Endowing the unknown function with a prior distribution allows not only making predictions of the unobserved function values but also quantifying and propagating the uncertainty of these predictions. This uncertainty further helps a learner decide on the next actions to pursue a goal. Particularly, it enables the learner to navigate the so-called exploration-exploitation trade-off, that is balancing what could be learned by exploring the unknown with what is already known. We return to the theory of decision, which is the focus of this dissertation, later in the introduction in Section 2.5 and, by now, focus on the regression problem. To this end, we start with the simplest example of Bayesian linear regression to write down the three components of Bayesian inference, i.e., prior, likelihood, and posterior, and introduce notions useful later in the thesis.

Simple example Let us assume linear generative process $f(\boldsymbol{x}) = \omega^\top \boldsymbol{x}$ parameterized by the unknown weights vector $\omega \in \mathbb{R}^d$. Further, assume the noise is i.i.d zero-mean Gaussian noise $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$. By now, we assume the noise is equally distributed for all $\boldsymbol{x} \in \mathcal{X}$, so-called *homoscedastic* noise. Later in the thesis, we also study *heteroscedastic noise* the distribution of which depends on the input, e.g., through the input-dependent variance of the normal distribution $\xi(\boldsymbol{x}) \sim \mathcal{N}(0, \sigma_\xi^2(\boldsymbol{x}))$. We aim to solve the regression problem, i.e., predict the value $y = f(\boldsymbol{x}) + \xi$ for some new input $\boldsymbol{x}$. To this end, we use Bayesian inference with prior, likelihood, and posterior components as follows.

The likelihood of observed data is a Normal distribution $p(y|\omega) = \mathcal{N}(\omega^\top \boldsymbol{x}, \sigma_\xi^2)$. As the learner collects t evaluations $\boldsymbol{x}_{1:t} \in \mathbb{R}^{t \times d}$ and $\boldsymbol{y}_{1:t} \in \mathbb{R}^t$, the likelihood takes the form of $p(\boldsymbol{y}_{1:t}|\omega) = \mathcal{N}(\boldsymbol{x}_{1:t}\omega, \sigma_\xi^2 \mathbf{I})$. Before observing the evaluations, we assume the weight vector ω follows a zero-mean isotropic Gaussian $p(\omega|\sigma) = \mathcal{N}(0, \sigma^2 \mathbf{I})$. Then as we incorporate the evaluations, we derive the posterior distribution of ω following Bayes' rule:

$$p(\omega|\boldsymbol{y}_{1:t}) = \mathcal{N}\Bigg(\underbrace{\sigma_\xi^{-2} \Sigma_t^{-1} \boldsymbol{x}_{1:t}^\top \boldsymbol{y}_{1:t}}_{\mu_t \in \mathbb{R}^d}, \underbrace{\sigma_\xi^{-2} \boldsymbol{x}_{1:t}^\top \boldsymbol{x}_{1:t} + \sigma^2 \mathbf{I}}_{\Sigma_t^{-1} \in \mathbb{R}^{d \times d}} \Bigg). \tag{2.3}$$

Intuitively, we consider various plausible explanations of how the initial data $\boldsymbol{y}_{1:t}$ was generated. Due to the chosen generative process and the prior, they follow a normal distribution with the posterior μ_t and Σ_t^{-1}. And then, we make predictions for a new input $\boldsymbol{x}$ using all possible regression parameters ω weighed by their posterior probability:

$$p(y|\boldsymbol{x}, \mathcal{D}_{1:t}, \sigma, \sigma_\xi) = \int p(\omega|\boldsymbol{y}_{1:t}) p(y|x, \omega) d\omega \tag{2.4}$$

$$= \mathcal{N}(\underbrace{\mu_t^\top \boldsymbol{x}}_{\mu_{pred}}, \underbrace{\boldsymbol{x}^\top \Sigma \boldsymbol{x} + \sigma_\xi^2}_{\sigma_{pred}^2}). \tag{2.5}$$

Thus, the Bayesian approach models all the possibilities for the prediction y, i.e., predictive posterior distribution, which follows a normal distribution here. This lets us quantify uncertainty in the predictions and observe its changes as we acquire more

evaluations. Given this predictive distribution for a simple linear regression example, let us now introduce two notions of uncertainty.

Sources of uncertainty The variance $\sigma^2_{pred}(\boldsymbol{x})$ in Eq. (2.5) accounts for two types of uncertainty. First, so-called *epistemic uncertainty* $\boldsymbol{x}^\top \Sigma^{-1} \boldsymbol{x}$, covers the uncertainty about the model due to the data scarcity. Second, so-called *aleatoric uncertainty*, σ^2_ξ covers the uncertainty about the value y given $f(\boldsymbol{x})$, originating from the inherent randomness. The epistemic uncertainty incorporates any knowledge about the function that we could know but do not know a priori, and it disappears as we interact with the function and collect data. The aleatoric uncertainty does not disappear due to the irreducible noise that can be known or unknown, depending on the problem, and thus should be learned.

Both types of uncertainties are important throughout the thesis. On the one hand, in high-stakes applications such as drug design, where avoiding unreliable actions is highly important, it is crucial to consider noise: two different actions might attain similar expected function values, but one might produce significantly noisier realizations. This risk awareness is associated with aleatoric uncertainty. On the other hand, well-calibrated bounds of epistemic uncertainty *quantify what we do not know* and allow sequentially decide on the evaluation budget for further exploration of the environment and improving our incomplete understanding of it. Failure to accurately quantify the uncertainty in high-risk decisions can lead to severe consequences [HW21; Dep19; Dep+18].

The following section shows how these uncertainties are quantified and propagated in the two common function models, a linear model over non-linear feature mappings and Gaussian processes. While here we assume the aleatoric uncertainty to be known and fixed, later Chapter 3 studies aleatoric uncertainty in the context of risk-aware sequential decision-making.

2.2 Probabilistic model

A probabilistic function model encodes the belief about f based on the observations following the Bayesian inference. This section briefly overviews two approaches for regression problems: Bayesian linear regression with feature mapping and Gaussian processes. We refer to [DFO20; RW06] for a more in-depth introduction.

Any probabilistic model aims at balancing two conflicting goals: the model expressiveness versus the feasibility of Bayesian inference. Linear models defined over non-linear feature mappings, $f(\boldsymbol{x}) = \omega^\top \boldsymbol{\phi}(\boldsymbol{x})$ with features $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^M$ and weights $\omega \in \mathbb{R}^M$, are a class of flexible parametric models that strike a good tradeoff between model capacity, interpretability and ease of use through the definition of the features. While the complexity of the model is controlled by the number of features, M, its capacity depends on their definition. Bayesian linear regression in Eq. (2.5) is a particular example, and similarly, we can write the three components of the probabilistic modeling in the case of arbitrary, possibly non-linear, features. These are likelihood $p(y_t|\omega) = \mathcal{N}(\omega^\top \boldsymbol{\phi}(\boldsymbol{x}_t), \sigma^2_\xi)$, the weights prior $p_i(w_i|\alpha)$, $i \in [M]$, parametrized by some $\alpha > 0$ and the posterior $p(\omega|\boldsymbol{y}_{1:t}, \alpha) \propto p(\boldsymbol{y}_{1:t}|\omega)p(\omega|\alpha)$. In the case of the weights prior $\mathcal{N}(0, \sigma^2)$ that takes the form of Eq. (2.3) but with the feature vector $\phi(\boldsymbol{x})$ instead of the vector $\boldsymbol{x}$. In a general

case, the inference difficulty crucially depends on the choice of the prior potentials.

In this dissertation, we consider Gaussian and sparse priors. By a sparse prior, we mean distributions that set some of the components w_i of the vector ω a priori to zero, which results in non-zero coefficients for features defined over small subsets of the input variables. We discuss it in Chapter 5 in the context of mixed-variable Bayesian optimization. There, the key is that only low-order interactions, represented as some polynomial features, between the variables, contribute significantly to the objective function. The sufficiency of low-order interactions was shown to be the case for many practical problems [Hoa+18; Rol+18; MK18], including hyperparameter tuning of deep neural networks [HKY17]. Thus, a large portion of possible features can be discarded a priori, simplifying the design space.

To conclude, we find the posterior distribution on the weights vector ω and use it to derive the predictive distribution on the unobserved function values. In contrast, the following section considers inference *directly in the function space*.

2.2.1 Gaussian processes

One way to work directly in the function space is a stochastic process. A stochastic process is a collection of random variables $\{f(\boldsymbol{x})\}_{\boldsymbol{x}\in\mathcal{X}}$ indexed by some deterministic variable in a given set $\mathcal{X}$. A stochastic process can be interpreted as a random element in a function space, intuitively generalizing the concept of standard random vectors to vectors with infinitely many components. That allows for using stochastic processes to describe probability distribution over a space of functions. This section introduces a special class of stochastic processes, Gaussian processes, and their two main building components: its kernel function encoding the process smoothness and computation of the posterior, allowing to perform regression.

A Gaussian process (GP) is a stochastic process for which any finite set of random function values $\{f(\boldsymbol{x}_i)\}_{i=1}^n$ follows a joint normal distribution. Any GP is fully specified by a mean function $\mu : \mathcal{X} \to \mathbb{R}$, and a covariance function, or *kernel*, $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. For every input $\boldsymbol{x} \in \mathcal{X}$, $f(\boldsymbol{x})$ is Gaussian random variable with mean $\mathbb{E}[f(\boldsymbol{x})] = \mu(\boldsymbol{x})$ and variance $\mathbb{E}[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))^2] = \sigma^2(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \boldsymbol{x})$. GPs are a rich class of non-parametric models, where the mean and the kernel specify the prior belief about the unknown f, denoted as $f \sim \mathcal{GP}(\mu, \kappa)$. The kernel function is an important notion that encodes the expressiveness of a Gaussian process, and we provide more details about it below.

Kernel function The kernel $\kappa(\cdot, \cdot)$ indicates how function values at different locations co-vary with each other, and, thus, it expresses our belief about the smoothness of f. An arbitrary function of input pairs does not necessarily define a valid kernel, and to do so, it should meet the following requirements. The function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if and only if there exists a Hilbert space $\mathcal{H}_\kappa$ induced by κ and a feature map $\phi : \mathcal{X} \to \mathcal{H}_\kappa$ such that the value $\kappa(\boldsymbol{x}, \boldsymbol{x}')$ for $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ is an inner product in $\mathcal{H}_\kappa$ of the feature maps $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$. The feature map $\phi(\boldsymbol{x})$ exists if and only if κ is a positive semi-definite function (PSD function, see Definition 9). The function must be symmetric, i.e., $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}', \boldsymbol{x})$. The *covariance matrix* $\boldsymbol{K}$ is the Gram matrix for the kernel κ,

i.e., a matrix with entries $\kappa_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $\boldsymbol{x}_i, \boldsymbol{x}_j$ from some finite set of points. It allows judging the validity of the kernel: Particularly, if the Gram matrix is a positive semi-definite matrix (PSD matrix, see Definition 8) for every finite set of data points, κ is a kernel.

An important class of kernels is stationary kernels. These kernels are functions of $\|\boldsymbol{x} - \boldsymbol{x}'\|$, i.e., invariant to translation in the input space, and the dependence on $\boldsymbol{x}, \boldsymbol{x}'$ can be written as $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \kappa_s(\|\boldsymbol{x} - \boldsymbol{x}'\|)$. A common stationary kernel is a squared exponential kernel, also known as Gaussian kernel:

$$\kappa_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_s^2 \exp\left(- \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2l} \right). \tag{2.6}$$

l is the so-called length-scale that carries the function smoothness per dimension in $\mathbb{R}^d$ and σ_s^2 is the signal variance that determines the variation of function values from the mean. We can conclude several facts about the Gaussian kernel from Eq. (2.6). First, the covariance function decays with $\|\boldsymbol{x} - \boldsymbol{x}'\| \to \infty$, i.e., intuitively, the further the points are, the less their function values correlate. Second, it is infinitely differentiable; thus, it imposes a strong smoothness assumption on the modeled function. Such strong smoothness assumptions might be unrealistic in practice, and another example of a stationary kernel imposing weaker assumptions is Matérn kernel [RW06] widely applied in practice. Formally, Matérn kernel with smoothness parameters ν is defined as follows:

$$\kappa_{\mathrm{Mat}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{l} \right)^{\nu} K_\nu \left(\sqrt{2\nu} \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{l} \right), \tag{2.7}$$

where l again denotes the length-scale, Γ and K_ν denote the gamma and modified Bessel function, respectively. Intuitively, the larger parameter ν results in smoother functions, and in its limit, i.e., as $\nu \to \infty$, the Matérn kernel coincides with the squared exponential kernel. Beyond Euclidean spaces, kernels can be defined over various structured domains such as spaces of graphs, groups, and lists [Bor+21; Hut+21; Bor+22].

Gaussian process posterior In the general case of arbitrary stochastic processes, computation of the posterior distribution such as in Eq. (2.5) is not possible. GPs are an exception allowing the posterior to be computed in the closed form we derive below. Without loss of generality, let us assume that the prior mean of a GP is zero everywhere. This is a common assumption, though practical knowledge of a problem might motivate a particular choice. Let us also assume the observation model from Eq. (2.1) with i.i.d. noise $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ and $\mathcal{D}_t = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_t, y_t)\}$ be a dataset of noise-perturbed function evaluations. Then, the posterior distribution of $f(\boldsymbol{x})$ is a Gaussian distribution, $\mathcal{N}(\mu_t(\boldsymbol{x}), \sigma_t^2(\boldsymbol{x}))$, with mean, variance and covariance defined as follows:

$$\mu_t(\boldsymbol{x}) = \mathbf{k}_t(\boldsymbol{x})^{\mathrm{T}} (\mathbf{K}_t + \sigma_\xi^2 \mathbf{I})^{-1} \boldsymbol{y}_{1:t}, \tag{2.8}$$

$$\sigma_t^2(\boldsymbol{x}) = \kappa_t(\boldsymbol{x}, \boldsymbol{x}), \tag{2.9}$$

$$\kappa_t(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}, \boldsymbol{x}') - \mathbf{k}_t(\boldsymbol{x})^{\top} (\mathbf{K}_t + \sigma_\xi^2 \mathbf{I})^{-1} \mathbf{k}_t(\boldsymbol{x}'). \tag{2.10}$$

Here $\mathbf{k}_t(\boldsymbol{x}) = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, \kappa(\boldsymbol{x}_t, \boldsymbol{x})]^\top$, $\mathbf{K}_t = [\kappa(\boldsymbol{x}, \boldsymbol{x}')]_{\boldsymbol{x},\boldsymbol{x}' \in \mathcal{D}_t}$ and $\mathbf{I} \in \mathbb{R}^{t \times t}$ are the covariance and identity matrices, respectively.

The posterior distribution above concludes the main components of the GP regression for a given kernel κ. Essentially, GP regression can be viewed as a Bayesian linear regression with a possibly infinite number of features. Particularly, following the definition, we have $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle$ that defines a way to go from the *explicit* computation of the feature vectors $\phi(\cdot)$ and their inner product to the *implicit* computation of $\kappa(\cdot, \cdot)$, and back. In order to construct a set of features corresponding to some kernel, one can use Mercer's theorem of eigenfunction expansion for the kernel. Replacing the inner product of features with a kernel function is known as *kernel trick*, and it allows the modeled function to be represented in some infinite-dimensional Hilbert space while requiring only a finite amount of computation. We discuss the connection to the Hilbert spaces in more detail in Section 2.3.

Credible intervals

> *"It ain't what you do not know that gets you into trouble. It is what you know for sure that just ain't so."*
>
> — Mark Twain

In the Bayesian setting, we quantify uncertainty by a range of values from the posterior probability distribution that covers, for example, 95% of the probability – this is called a *95% credible interval*. For example, a 95% credible interval at a time t and some point $\boldsymbol{x}$ can be computed based on the posterior Gaussian distribution in Eq. (2.5). So far, we have considered *regression* only, and later in the thesis, we focus on *sequential decision-making* where the learner's decision to acquire the subsequent evaluation depends on past observations and the obtained probabilistic model. This requires the credible intervals to be valid for a random stopping time, i.e., *anytime valid*, which is in contrast to being valid at a single moment of time. To this end, time-uniform concentration inequalities have been used, resulting in sequences of valid time-uniform confidence sets.

This section introduces the notion of calibrated credible intervals and an example of setting it up. In literature, some particular examples of applying it to sub-Gaussian families include [Abb12] for linear bandits, [DMP18] for kernel regression, and [KK18] in bandits with heteroscedastic noise. Deriving anytime valid confidence intervals is an actively developing area, e.g., [WR20; Cho+22].

Consider an unknown function $f(\boldsymbol{x})$ modeled by a GP as introduced in Section 2.2.1. Intuitively, the model is said to be calibrated if it covers the function f jointly over the whole domain of $\boldsymbol{x} \in \mathcal{X}$ with high probability. Formally, we say the model with the posterior mean $\mu_t(\boldsymbol{x})$ and variance $\sigma_t^2(\boldsymbol{x})$ is well-calibrated with respect to f if the corresponding credible intervals are well-calibrated as defined below:

Definition 1 (Calibrated credible intervals)**.** *Let $\mu_{t-1}(\boldsymbol{x})$ and $\sigma_{t-1}(\boldsymbol{x})$ be the posterior mean and posterior standard deviation of the predictive distribution for the function $f(x)$.*

We say that the credible interval $[\mu_{t-1}(\boldsymbol{x}) - \beta_t\sigma_{t-1}(\boldsymbol{x}), \mu_{t-1}(\boldsymbol{x}) + \beta_t\sigma_{t-1}(\boldsymbol{x})]$ *is calibrated with* $\delta \in (0,1)$ *with respect to* f *over input set* $\mathcal{X}$*, if there exist a sequence* $\{\beta_t\}_{t\geq 0}$ *with* $\beta_t = \beta_t(\delta)$ *and* $\beta_t \in \mathbb{R}_{\geq 0}$*, such that jointly* $\forall t \geq 0$ *and* $\forall \boldsymbol{x} \in \mathcal{X}$ *we have*

$$P\big(|\mu_{t-1}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \beta_t\sigma_{t-1}(\boldsymbol{x})\big) \geq 1 - \delta. \tag{2.11}$$

The definition above states that one should provide a sequence of $\{\beta_t\}$ to build an anytime-valid calibrated model. This sequence depends on both the input domain $\mathcal{X}$, δ, and the assumption on the function f. As an example, consider the following lemma for a function $f(\boldsymbol{x})$ being a GP sample:

Lemma 1 ([Sri+10])**.** *Let* $\mathcal{X}$ *be finite and* $\delta \in (0,1)$*. For each* t *define* $\beta_t = \sqrt{2\log(|\mathcal{X}|\pi_t/\delta)}$ *where* $\sum_{t\geq 1}\pi_t^{-1} = 1, \pi_t > 0$*. Then, with probability at least* $1-\delta$*, for all* $\boldsymbol{x} \in \mathcal{X}$ *and* $t \geq 1$ *we have* $|\mu_{t-1}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \beta_t\sigma_{t-1}(\boldsymbol{x})$*.*

Lemma 1 gives an example of how the sequence of calibrated credible intervals looks. Later in the thesis, in Chapter 3, this lemma is the key to obtaining convergence guarantees for Bayesian optimization, and we discuss it in more detail later.

The following two sections consider the following questions: (1) Can we go beyond the Bayesian assumption? and (2) How can we measure the complexity of learning the function? Particularly, so far, we assumed that f is a *random* function sampled from a $GP(0,\kappa)$, i.e., the Bayesian setting. Section 2.3 considers the frequentist setting in which one assumes that f is an arbitrary *fixed* function. In any of the two settings, we naturally expect that as we acquire more evaluations, the confidence bounds shrink, i.e., as $t \to \infty$: $|\mu_t(\boldsymbol{x}) - f(\boldsymbol{x})| \to 0, \quad \forall \boldsymbol{x} \in \mathcal{X}$. Section 2.4 introduces the measure of information gain that quantifies how much we can reduce the uncertainty in the model from a finite sample set. Later, we use the information gain in convergence guarantees in sequential decision-making.

2.3 Non-Bayesian setting: Reproducing kernel Hilbert spaces

This section considers the frequentist setting in which the unknown function f is an arbitrary *fixed* function estimated from the observed data. Without further assumptions, inferring a reasonable underlying function from the data is impossible. The frequentist approach looks at the regression problem from a regularization perspective, i.e., penalizing the complexity of the function. For example, the least squares estimator with regularization $R(f)$ over data $\{\boldsymbol{x}_i, y_i\}_{i=1}^t$ is defined as follows:

$$\hat{f}_t = \operatorname*{arg\,min}_{f\in\mathcal{F}} \sum_{i=0}^{t} \big(f(\boldsymbol{x}_i) - y_i\big)^2 + R(f). \tag{2.12}$$

This section introduces the complexity defined by the value of the function norm in a reproducing kernel Hilbert spaces. In short, reproducing kernel Hilbert spaces are a particularly useful and rich family of hypothesis spaces that allow for elegant and efficient learning problems. The section discusses the close connection between RKHSs and GPs,

where the latter is used as a model, and how the credible intervals obtained from Bayesian posterior match frequentist confidence intervals.

Let $\mathcal{F}(\mathcal{X})$ be a vector space of real-valued functions over a compact set $\mathcal{X}$. A Hilbert space $\mathcal{H}(\mathcal{X}) \subset \mathcal{F}(\mathcal{X})$ is a vector space endowed with an inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ that is complete with respect to the norm defined by the inner product $\|f\| = \langle f, f \rangle^{1/2}$. Hilbert spaces allow us to apply finite-dimensional linear algebra for infinite-dimensional spaces of functions, e.g., completeness of the space guarantees convergence of some algorithms. Reproducing kernel Hilbert spaces (RKHS, [RW06; Wah90]) is of particular interest and is formally defined as follows:

Definition 2 (RKHS)**.** *A Hilbert space $\mathcal{H}_\kappa(\mathcal{X}) \subset \mathcal{F}(\mathcal{X})$ with inner product $\langle \cdot, \cdot \rangle_\kappa$ is called* reproducing kernel Hilbert space (RKHS) *if two properties hold: (i) existence of the representer $\kappa(\boldsymbol{x}, \boldsymbol{x}') \in \mathcal{H}_\kappa(\mathcal{X}) \;\; \forall \boldsymbol{x}' \in \mathcal{X}$; (ii) reproducing property $\langle f, \kappa(\cdot, \boldsymbol{x}) \rangle_\kappa = f(\boldsymbol{x}), \; \forall f \in \mathcal{H}_\kappa(\mathcal{X})$.*

Intuitively, the definition says that for any $\boldsymbol{x}$, there exists a function $\kappa_{\boldsymbol{x}}$ living in the space $\mathcal{H}_\kappa$, so-called representer of $\mathcal{H}_\kappa$, such that evaluating any f from that space at $\boldsymbol{x}$ corresponds to the inner product with the representer. Importantly, any RKHS uniquely defines a reproducing kernel and vice versa, i.e., a kernel defines a unique RKHS. For more details, we refer to Section A.2.

The induced RKHS norm measures the smoothness, or complexity, of a function $f \in \mathcal{H}_\kappa(\mathcal{X})$, which the following reasoning can intuitively explain. Since any function can be expanded as $f(\boldsymbol{x}) = \sum_{i=0}^{\infty} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i), \alpha_i \in \mathbb{R}, \boldsymbol{x}, \boldsymbol{x}_i \in \mathcal{X}$, its norm can be written as $\|f\|_\kappa^2 = \langle f, f \rangle_\kappa = \sum_{i=1}^{\infty} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Then, intuitively, the complexity of f and the value of its norm are related through the coefficients $\{\alpha_i\}_{i=0}^{\infty}$ as how fast α_i values should converge to zero as $i \to \infty$.

With such an intuition on the RKHS norm, we can write the regularized problem Eq. (2.12) over the hypothesis space $f \in \mathcal{H}_k$ with a kernel-dependent penalization $R(f) = \|f\|_\kappa$. Similar to kernels that impose smoother samples from a GP in the Bayesian setup, a larger penalty $\|\cdot\|_\kappa$ corresponds to RKHS with smoother functions in the frequentist setup. But how do we find a solution for such a regularized problem? The *representer theorem* [SHS01] provides a way to find a (unique) solution for Eq. (2.12) in the (potentially) infinite-dimensional RKHS in a compact and computable way: representing the function via the finite sum over the training points, i.e., $\hat{f}_t(\boldsymbol{x}) = \sum_{i=0}^{t} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i)$. Intuitively, we represent the minimizer $\hat{f} \in \mathcal{H}_\kappa$ as a finite linear combination of the kernel terms living in $\mathcal{H}_\kappa$.

The following Section 2.3.1 explains how this minimizer relates to the GP posterior, allowing the use of the same Bayesian posterior updates (2.8) and (2.9)for modeling an unknown function $f \in \mathcal{H}_\kappa$. In addition, the latter allows for setting valid confidence bounds for the function based on the GP posterior, which we discuss in Section 2.3.2.

2.3.1 Relation between RKHS and Gaussian processes

We would like to model a function $f \in \mathcal{H}_\kappa$ with a prior $\mathcal{GP}(0, \kappa)$ and use the nice machinery of GPs introduced so far in Section 2.2.1. There is, however, a trick when using them for

RKHS functions. On the one hand, the assumption of f being a sample from a GP, i.e., $f \sim GP(0, \kappa)$, leads to (almost surely) unbounded norm $\|f\|_\kappa$ ([Wah90]). On the other hand, the smoothness assumption on f in RKHS is imposed via the fixed value of the norm in RKHS $\|f\|_\kappa \leq \mathcal{B}, \mathcal{B} > 0$. That leads to the fact that the function sets under a-priori Bayesian assumption and frequentist assumptions do not intersect. However, GP posterior mean is, in fact, κ-norm bounded, and thus, both the Bayesian and frequentist smoothness assumptions reflect similar smoothness properties of the kernel. *That allows modeling unknown $f \in \mathcal{H}_\kappa$ with prior $f \sim GP(0, \kappa)$ though the prior is actually misspecified.* This relation between RKHS and GPa is used in proofs in the thesis. Below, we provide the norm bounds for the samples and the mean that explain the gap.

RKHS norm for the samples. First, we show the unbounded RKHS norm for the GP samples. Let the function be a sample from a GP $f \sim GP(0, \kappa)$. Following Mercer's theorem, the kernel κ of the prior can be represented via eigenfunctions expansion $\{\phi_i\}_{i=0}^\infty$ as $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=0}^\infty \lambda_i \phi_i(\boldsymbol{x}) \phi_i(\boldsymbol{x}')$. Then, the function f can be represented as $f = \sum_{i=0}^\infty f_i \phi_i(\boldsymbol{x})$ with the coefficients $f_i \sim \mathcal{N}(0, \lambda_i)$. Then, the expectation of the function norm can be written as

$$\mathbb{E}\|f\|_\kappa^2 = \mathbb{E}\langle f, f \rangle_\kappa = \sum_{i=0}^\infty \frac{\mathbb{E} f_i^2}{\lambda_i} = \sum_{i=0}^\infty \frac{\lambda_i}{\lambda_i}$$

and it is unbounded for any kernel with an infinite number of non-zero eigenvalues. Since the norm $\|f\|_\kappa$ equals to ∞ almost surely, the function f sampled from a GP does not belong to the space $\mathcal{H}_\kappa$. Intuitively, it states that the sample paths from GPs are rougher than RKHS functions.

RKHS norm for the mean. In contrast to the samples, the GP posterior mean has a bounded norm. In particular, we can show that for the optimization problem Eq. (2.12) with the RKHS norm regularization, the posterior $\mu_t(\boldsymbol{x})$ in Eq. (2.8) coincides with the estimator $\hat{f}_t$. Following the representer theorem, let's write the solution as $\hat{f}_t(\boldsymbol{x}) = \sum_{i=0}^t \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i)$ and reformulate the problem in terms of the multipliers $\{\alpha_i\}_{i=0}^t$. Then solving the problem leads to the minimizer

$$\hat{\alpha} = (\kappa_t + \sigma_\xi I)^{-1} \boldsymbol{y}_{1:t}.$$

This solution coincides with GP the posterior mean. Since the representation sum for $\hat{f}_t$ is finite, μ_t resides in $\mathcal{H}_\kappa$ and allows for modeling a function $f \in \mathcal{H}_\kappa$ with Gaussian prior $\mathcal{GP}(0, \kappa)$.

2.3.2 Confidence intervals

The described relation between the RKHS and GP kernels allows for building frequentist confidence intervals for the unknown f. Since the estimator $\hat{f}_t$ in Eq. (2.12) coincides with the posterior mean, the key is to set the confidence parameter β_t in Eq. (2.11) to ensure that f remains within the confidence bounds $\hat{f}_t(\boldsymbol{x}) \pm \beta_{t+1} \sigma_t(\boldsymbol{x})$ for any $t \geq 0$. Similar to the Bayesian setting, below, we provide an example for the sequence of β_t to build an anytime-valid calibrated model.

Lemma 2 ([Sri+10]). *Let $\delta \in (0,1)$. For each t define $\beta_t = 2\|f\|_k^2 + 300\gamma_t \ln^3(t/\delta)$. Then, with probability at least $1-\delta$, for all $\boldsymbol{x}$ and t we have $|\mu_{t-1}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \beta_t \sigma_{t-1}(\boldsymbol{x})$.*

Lemma 2 uses an essential notion of information gain γ_t – a measure that quantifies the speed at which we learn about f and directly relates to the complexity of the kernel function κ. We talk about this notion in the following section below.

2.4 Information gain

We are interested in how fast the uncertainty about f is reducing from revealing the evaluations. To measure this, we rely on information theory, which studies how to quantify information about the unknown value one can gain from observing another correlated value. That leads to the notion of information gain, which becomes a measure of the complexity of learning the unknown function from its noisy evaluations.

The amount of uncertainty about a random variable can be expressed by the information-theoretic measure, its *entropy*, denoted as $H(\cdot)$. Consider $H(\cdot)$ for a set of random function values $f_{1:t} = [f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_t)]^\top$. Formally, the definition of entropy for a joint probability density function $p(\cdot)$ is given as follows:

$$H(f_{1:t}) = \mathbb{E}[-\log p(f_{1:t})]. \tag{2.13}$$

Conditional entropy $H(f_{1:t}|\boldsymbol{y}_{1:t})$ conditioned on the evaluations $\boldsymbol{y}_{1:t}$ is defined in the same spirit. Intuitively, it expresses the amount of uncertainty about $f_{1:t}$ left after observing their realizations; in other words, what $\boldsymbol{y}_{1:t}$ does not tell about $f_{1:t}$. These two measures, $H(f_{1:t})$ and $H(f_{1:t}|\boldsymbol{y}_{1:t})$, allow quantifying the amount of information one can gain about the function values $f_{1:t}$ once observing the corresponding noisy evaluations $\boldsymbol{y}_{1:t}$, which is known as *information gain*. Particularly, it is measured by mutual information $I(\boldsymbol{y}_{1:t}, f_{1:t})$ ([CT06]) defined as follows:

$$I(\boldsymbol{y}_{1:t}, f_{1:t}) \triangleq H(f_{1:t}) - H(f_{1:t}|\boldsymbol{y}_{1:t}). \tag{2.14}$$

Mutual information is a symmetric function that can also be rewritten the other way around as $I(\boldsymbol{y}_{1:t}, f_{1:t}) \triangleq H(\boldsymbol{y}_{1:t}) - H(\boldsymbol{y}_{1:t}|f_{1:t})$. This general information-theoretic measure is valid in both the Bayesian (f is a sample from a GP) and frequentist (f lives in an RKHS) settings; however, its further computation depends on the model choice. In the case of modeling f with a Gaussian process with posterior $\mathcal{GP}(\mu_{t-1}(\boldsymbol{x}), \sigma_{t-1}^2(\boldsymbol{x}))$ defined in (2.8) and (2.9) under Gaussian homoscedastic noise, $\xi \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma_\xi^2)$, the information gain can be expressed in terms of the predictive variances:

$$I(\boldsymbol{y}_{1:t}, f_{1:t}) = \frac{1}{2}\log\det(\mathbf{I} + \sigma_\xi^{-1}\boldsymbol{K}_t)\frac{1}{2} = \sum_{i=1}^{t}\log(1 + \sigma_\xi^{-2}\sigma_{t-1}^2(\boldsymbol{x}_i)). \tag{2.15}$$

Intuitively, the resulting value reveals the informativeness of points $\boldsymbol{x}_{1:t}$ since the value of $I(\boldsymbol{y}_{1:t}, f_{1:t})$ depends only on the inputs $\boldsymbol{x}_{1:t}$ but not on the corresponding evaluations.

The complexity of learning the unknown function f with a given $\mathcal{GP}(0, \kappa)$ is then

quantified by the maximum amount of information that we could get about the function from a set $\boldsymbol{x}_{1:t}$ of evaluations:

$$\gamma_t := \max_{\boldsymbol{x}_{1:t} \subset \mathcal{X}} I(\boldsymbol{y}_{1:t}, f_{1:t}). \tag{2.16}$$

We refer to γ_t as the *maximum information gain*. It is a problem-dependent quantity relying on the properties of both the choice of kernel κ and the input space $\mathcal{X}$. The scaling of maximum information gain is known under both the Bayesian and frequentist settings. In case of $\mathcal{X} \in \mathbb{R}^d$ and homoscedastic noise, the upper bounds on γ_t for widely used kernels are provided in [Sri+10] and typically scale sublinearly in t, e.g., for linear kernel $\gamma_t = \mathcal{O}(d \log t)$, and in case of squared exponential kernel $\gamma_t = \mathcal{O}(d \log^{d+1} t)$. We make use of these bounds in the thesis. In case of the non-standard settings with other data-generating assumptions, such as heteroscedasticity in noise [Mak+21b], or new kernels, such as neural tangent kernel [KK21], these bounds should be shown separately. We discuss how to prove the bounds in the case of risk-aware heteroscedastic Bayesian optimization in Chapter 3.

The quantity of maximum information gain will be particularly interesting in the next section, where we introduce Bayesian optimization. In the following, we are interested not only in performing inference (regression) for the unknown function f but also in iteratively deciding about the next inputs to evaluate the function on.

2.5 Bayesian Optimization

In this section, we focus on Bayesian optimization and online decision-making. In online learning problems, an agent needs to learn on the fly how to choose optimal actions. The crux of these problems is the explore-exploit trade-off: the agent needed to balance increasing the knowledge about the unknown and maximizing performance based on the collected knowledge. In this section, we introduce Bayesian optimization – a well-established paradigm for such problems that can naturally incorporate the exploration-exploitation trade-off.

Bayesian optimization (BO; [Moč75]) refers to a family of sequential model-based optimization methods for *costly-to-evaluate, complex, black-box objectives*. The unknown function is a black box providing no information about gradients and is only accessible through point evaluations. Moreover, the functions are usually complex, meaning they are non-convex and multimodal. Finally, each point evaluation comes with a price, measured in time or budget, motivating for methods effective under small dataset settings. BO is a well-established paradigm successfully applied to many scientific domains, such as drug and material discovery [Kor+19; GH20; NFP11], genetics [Gon+15; Mos+20], robotics [BKS16; Cal+16; Mar+16], hyperparameter tuning of complex learning systems [Kir+19; Che+18; SLA12a], to name a few.

BO algorithms leverage two components: (i) a *probabilistic function model*, that encodes the belief about f based on the observations available, and (ii) an *acquisition function* that helps to navigate the queries of new inputs. Intuitively, the acquisition function expresses

the informativeness of any input $\boldsymbol{x} \in \mathcal{X}$ about the location of the optimizer $\boldsymbol{x}_*$. Thus, based on the probabilistic model of f, we query the best input measured by the acquisition function, then update the model with the observation and repeat this procedure. Algorithm 1 describes a generic loop for implementing BO. The effectiveness of such a sequential procedure is measured by some *performance metric*. In the following, we describe these three building blocks: the probabilistic model, acquisition function, and performance metrics.

Algorithm 1 Generic Bayesian optimization

Input: Objective f, prior model $p(f)$, acq. function $\alpha(\cdot)$, budget T
Initialize data $\mathcal{D}_0 = \emptyset$
for $t = 1, \ldots, T$ **do**
  Select input $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x} | \mathcal{D}_{t-1})$
  Acquire output $y_t = f(\boldsymbol{x}_t) + \xi(\boldsymbol{x}_t)$
  Augment data $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t, y_t)\}$
  Compute posterior $p(f | \mathcal{D}_t)$
end for
Output: reported solution $\hat{\boldsymbol{x}}_T$

2.5.1 Probabilistic model

Using prior knowledge about the objective function to design a model specifically tailored to the problem can lead to faster convergence in Bayesian optimization. This is because a very general, non-parametric function class has a larger hypothesis space, which can make it more difficult to find the optimal function. On the other hand, using a simpler, parametric function class can make the optimization process more efficient. This is because Occam's Razor, a principle in statistics, suggests that the simplest model that is sufficiently accurate for the problem should be used. Therefore, it is important to carefully consider the statistical model chosen for optimization, as it can be more influential on the efficiency of the process than the acquisition function [Sha+16].

Gaussian processes

The most common option is to use a Gaussian process prior $f \sim \mathcal{GP}(0, \kappa)$ for some pre-defined kernel κ. One main reason for using a GP model is the ability to conveniently perform probabilistic inference that we discuss in detail in Section 2.2.1.

In addition to the inference, in practice, one has to decide on the hyperparameters of the GP, e.g., length-scale in Gaussian kernel Eq. (2.6). Let us denote the set of parameters as θ. From the Bayesian perspective, it seems natural to introduce prior distributions over the hyperparameters, so-called hyper-prior $p(\theta)$. Following the Bayesian inference, as given the collected data $\boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t}$, we can write down the posterior distribution over the hyperparameters as

$$p(\theta | \boldsymbol{y}_{1:t}, \boldsymbol{x}_{1:t}) = \frac{p(\boldsymbol{y}_{1:t}, | \boldsymbol{x}_{1:t}, \theta) p(\theta)}{p(\boldsymbol{y}_{1:t}, | \boldsymbol{x}_{1:t})}. \tag{2.17}$$

Given this hyper-posterior, one can consider two approaches: being *fully Bayesian*, i.e., incorporating the hyper-posterior further in the predictive distribution for y, or using a *point estimate* of the hyperparameters. The latter is usually more pragmatic from the computational reasons than fully Bayesian. A common practical choice for hyperparameters estimation is the maximum a posteriori (MAP) estimate computed by maximizing the numerator of Eq. (2.17). Alternatively, one can use Markov Chain Monte Carlo (MCMC) methods to draw the samples from the hyper-posterior.

The main drawback of GPs is their computational complexity since the posterior inference scales cubically with the number of collected evaluations d. Particularly, this complexity is caused by the inversion of the covariance matrix perturbed by noise variance Eq. (2.10). On the one hand, this inversion time might be neglected compared to the time-consuming evaluation of the objective function. On the other hand, we might be interested in faster inference, especially in cases where we can collect more function evaluations at a descent time cost. For example, in Chapter 4, we apply BO for functions that can be evaluated at different fidelities, so-called multi-fidelity BO, where we can access less accurate but less time-consuming function evaluations.

In contrast, Bayesian linear regression Eq. (2.3) and consequently the Bayesian linear regression with features $\phi(\boldsymbol{x}) \in \mathbb{R}^m$, scales in the number of features m. Random Fourier features (RFF) are a way to return from the kernel representation to the feature representation, i.e., finding the feature map $\phi(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}^m$ such that $\kappa(\boldsymbol{x}, \boldsymbol{x}') \approx \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x}')$. RFFs are proven to be effective for large-scale kernel methods and have also been successfully applied in BO [Jen+17; Per+17].

Neural networks

One of BO's main challenges is to develop more scalable and flexible methods while still accurately accounting for model uncertainties. This is necessary because BO relies on a principled treatment of model uncertainties. GP models perform well in small-scale settings, are well-studied, and have proven to enjoy guarantees for calibration, concentration, and learning complexity rates. In contrast, models based on probabilistic neural networks are less theoretically understood but often perform better at a larger scale.

To address the small-scale limitation, alternative methods that adaptively learn the feature expansion in a data-driven fashion were proposed, for example, DNGO [SLA12b] and BOHAMIANN [Spr+16]. They have a more efficient computational complexity in terms of data set size compared to traditional GP models. Specifically, the computational cost of inference scales linearly with the number of data points for both models. DNGO is a Bayesian linear model with features learned by a feed-forward neural network. BOHAMIANN follows fully Bayesian neural network computing the predictive distribution by marginalizing the weights with stochastic Hamiltonian Monte Carlo sampling (sampling from the weights posterior). The uncertainty estimates, however, produced by GPs are generally better, and both methods can be interpreted as approximations of a GP.

Some recent uncertainty estimation techniques for deep neural networks include en-

semble methods, heteroscedastic regression, and concrete dropout [LPB17; GG16], where the focus is on variance as a proxy of epistemic uncertainty. Alternatively, Jain, Lahlou, Nekoei, Butoi, Bertin, Rector-Brooks, Korablyov, and Bengio [Jai+22] propose a direct estimation of epistemic uncertainty by training a secondary learner to predict the model's generalization error and subtracting an estimate of aleatoric uncertainty. When the models need to be re-calibrated after training, a simple re-calibration of Bayesian and non-Bayesian models was proposed by Kuleshov, Fenner, and Ermon [KFE18].

Common assumptions for probabilistic models in BO The vast majority of previous BO works assume (sub-) Gaussian and *homoscedastic* noise (i.e., input independent and of some known fixed level). Both assumptions can be restrictive in practice. For example, as demonstrated in [Cow+21], most hyperparameter tuning tasks exhibit heteroscedasticity. A few works relax the first assumption and consider, e.g., heavy-tailed noise models [RG19a], and adversarially corrupted observations [BKS20]. The second assumption is typically generalized via *heteroscedastic* Gaussian process (GP), allowing an explicit dependence of the noise distribution on the evaluation points [BGL18; Bog+16; Bin+19; KK18]. Similarly, in Chapter 3, we consider heteroscedastic GP models, but unlike the previous works, we specifically focus on the risk associated with querying and reporting noisy points.

2.5.2 Acquisition functions

The goal of the acquisition function is to simultaneously learn about inputs that are likely to be optimal and about poorly explored regions of the input space, i.e., to trade-off exploitation against exploration. Thus, BO reduces the original hard black-box optimization problem to a series of cheaper problems

$$\boldsymbol{x}_t \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha_t(\boldsymbol{x}). \tag{2.18}$$

A great number of different acquisition functions have been developed over the years, including a significant number of variants of popular algorithms such as GP-UCB [Sri+10], Expected Improvement [Moč75], and Thompson Sampling [CG17]. Below, we discuss these methods in more detail.

Optimism in the face of uncertainty

Optimism in the face of uncertainty ([LR85b]) is a general paradigm for many stochastic online learning problems, e.g., various bandit problems, Bayesian optimization, and online reinforcement learning problems ([Abb12; Sri+10]). The principle tackles the exploration-exploitation dilemma by maintaining a confidence set for the unknown value of interest and choosing an action based on the optimistic estimate.

Bayesian methods use the multi-armed bandit problem formulation and exploit the notion of *confidence bounds* around the posterior mean to define a region that includes f with high probability. These are celebrated methods for regret minimization, including

the class of Gaussian process upper confidence bound (GP-UCB) algorithms [Sri+10], which suggests selecting the point with the highest *upper confidence bound*, i.e.,

$$\alpha_t^{\text{GP-UCB}}(\boldsymbol{x}) = \mu_{t-1}(\boldsymbol{x}) + \beta_t \sigma_{t-1}(\boldsymbol{x}). \tag{2.19}$$

Thus, the exploration-exploitation dilemma is implemented by balancing the posterior mean and posterior variance. Particularly, the strategy is to favor a point $\boldsymbol{x}_t$ that maximizes both expected value based on the current belief of f, $\mu(\boldsymbol{x}_t)$, which promotes exploitation, and the uncertainty in the outcome, posterior variance $\sigma^2(\boldsymbol{x}_t)$, which encourages exploration. The parameter $\beta_t > 0$ optimally balances the trade-off between exploration and exploitation to minimize cumulative regret.

Improvement-based acquisition

Improvement-based approaches rely on the concept of making progress towards the optimal solution and use the notion of *improvement* over the current optimum. Two common policies in this category are the probability of improvement (PI; [Kus64]) and expected improvement (EI; [Moč75]). Let the current best solution found among all previously evaluated points $\mathcal{D}_t$ be denoted as $f(\boldsymbol{x}_t^*)$ and defined by

$$\boldsymbol{x}_t^* = \arg\max_{\boldsymbol{x} \in \mathcal{D}_t} f(\boldsymbol{x}) \ . \tag{2.20}$$

PI suggests maximizing the probability of improving over the current best, i.e.,

$$\alpha_t^{\text{PI}}(\boldsymbol{x}) = P(f(\boldsymbol{x}) \geq f(\boldsymbol{x}_{t-1}^*)) = \Phi\left(\frac{\mu_t(\boldsymbol{x}) - f(\boldsymbol{x}_{t-1}^*)}{\sigma_t(\boldsymbol{x})}\right) \tag{2.21}$$

where $\Phi(\cdot)$ denotes the Gaussian cumulative distribution function. One major limitation is that PI focuses solely on exploitation, which may lead to being stuck at local maxima because it lacks an exploration component. Some variations of PI address this issue by introducing a trade-off parameter [BCD10] to balance exploitation and exploration.
EI is an alternative to PI, which estimates the value of the improvement instead of the probability:

$$\alpha_t^{\text{EI}}(\boldsymbol{x}) = \mathbb{E}[\max\{0, f(\boldsymbol{x}) - f(\boldsymbol{x}_t^*)\} | \mathcal{D}_{t-1}] = (\mu_t(\boldsymbol{x}) - f(\boldsymbol{x}_t^*))\, \Phi(Z) + \sigma_t(\boldsymbol{x}) \phi(Z)$$

with $Z \triangleq \frac{\mu_t(\boldsymbol{x}) - f(\boldsymbol{x}_t^*)}{\sigma_t(\boldsymbol{x})}$ and where $\Phi(\cdot)$ and $\phi(\cdot)$ denote the Gaussian cumulative distribution function and probability distribution function, respectively. Intuitively, EI is high for the inputs expected to yield a high payoff and for those with high predicted uncertainty. This allows for a balance between exploitation and exploration. The EI criterion can also be modified by including a parameter that controls the exploration/exploitation trade-off [BCD10].

Information-theoretic acquisition

Information-theoretic approaches aim at reducing as much as possible the entropy of the distribution over the optimizer/optimum of the objective, respectively. In other words,

they aim to identify the input that maximizes the corresponding expected information gain Eq. (2.14). These approaches include *entropy search* (ES) [HS12] and *predictive entropy search* (PES) [HHG14]. These methods use the negative differential entropy of the posterior probability distribution as a measure of information about the global maximum and select the point that maximizes the expected reduction in this quantity. Formally, they can be written as:

$$\alpha_t^{\mathrm{ES}}(\boldsymbol{x}) = H[p(\boldsymbol{x}_t^*|\boldsymbol{x}_{1:t-1})] - \mathbb{E}[H[p(\boldsymbol{x}_t^*|\boldsymbol{x}_{1:t-1}, \boldsymbol{x})]] \tag{2.22}$$

where $H[\cdot]$ denotes the differential entropy Eq. (2.13) and the expectation is taken with respect to the posterior predictive distribution of y given $\boldsymbol{x}_t$. The acquisition function for these approaches can be difficult to evaluate directly. Hence, ES uses approximations while PES reformulates the acquisition function in terms of the mutual information between the global maximum and the predicted output, resulting in an acquisition function based on the entropies of predictive distributions.

Max-value Entropy Search (MES; [WJ17]) is another approach to deal with the scalability in high-dimensional problems: instead of looking at the entropy of a variable $\boldsymbol{x}^* \in \mathcal{X} \subset \mathbb{R}^d$, the authors propose to look at a max-value point $y^* = f(\boldsymbol{x}^*) \in \mathbb{R}$. Formally, let x^* and y^* denote the true optimizer and optimum in a noiseless setting, respectively,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}), \qquad y^* = f(\boldsymbol{x}^*).$$

MES quantifies the informativeness of a point $\boldsymbol{x}$ using the gain in mutual information between the minimum y^* and the pair $\{\boldsymbol{x}, y\}$ with the evaluation $y = f(\boldsymbol{x})$. Since the mutual information $I(\cdot, \cdot)$ is symmetrical, we can write by two equivalent formulations:

$$I(\{\boldsymbol{x}, y\}; y^*|\mathcal{D}_t) = H[p(y^*|\mathcal{D}_t)] - \mathbb{E}_y(H[p(y^*|\mathcal{D}_t \cup \{\boldsymbol{x}, y\})]) = \tag{2.23}$$

$$I(y^*; \{\boldsymbol{x}, y\}|\mathcal{D}_t) = H[p(y|\mathcal{D}_t, \boldsymbol{x})] - \mathbb{E}_{y^*}[H[p(y|\mathcal{D}_t, \boldsymbol{x}, y^*)]]. \tag{2.24}$$

The expectation in (2.23) and (2.24) are taken over $p(y|\mathcal{D}_t, \boldsymbol{x})$ and $p(y^*|\mathcal{D}_t)$ respectively. Eq. (2.23) has an intuitive interpretation: the utility of a candidate $\boldsymbol{x}$ is equal to the expected reduction in entropy obtained by querying $y = f(\boldsymbol{x})$ for the distribution over the optimum y^*. Though Eq. (2.23) is more intuitive, MES uses the equivalent definition in Eq. (2.24) as it has an effective approximation. The expectation over $p(y^*|\mathcal{D}_t)$ is approximated with a Monte Carlo estimate by sampling M function optima y^*

$$\mathbb{E}\big[H[p(y|\mathcal{D}_t, \boldsymbol{x}, y^*)]\big] \approx \frac{1}{M} \sum_{y^*} H[p(y|\mathcal{D}_t, \boldsymbol{x}, y^*)].$$

Thus, the first term $H[p(y|\mathcal{D}_t, \boldsymbol{x})]$ of Eq. (2.24) can be computed in a closed form for the entropy of a Gaussian univariate variable. For the second term, note that $p(y|\mathcal{D}_t, \boldsymbol{x}, y^*)$ is distributed as a truncated Gaussian since $y \leq y^*$. Thus, it can be computed in a closed

form, leading to the following approximation:

$$\alpha^{MES}(\boldsymbol{x}) \approx \frac{1}{M} \sum_{y^*} \left[\frac{\gamma_{y^*}(\boldsymbol{x})\psi(\gamma_{y^*}(\boldsymbol{x}))}{2\Psi(\gamma_{y^*}(\boldsymbol{x}))} + \log(\Psi(\gamma_{y^*}(\boldsymbol{x})) \right], \tag{2.25}$$

where ψ and Ψ indicate the probability density function and the cumulative density function of the standard Gaussian distribution respectively, and $\gamma_{y^*}(\boldsymbol{x}) = \frac{y^* - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})}$. Finally, the approximation Eq. (2.25) requires sampling the optima y^* and there are two alternative methods: (1) using the Gumbel distribution [Fis30] and (2) using Thompson sampling ([Tho33]).

Sampling-based acquisition

Finally, Thompson sampling (TS) [Tho33; Tho35] is an acquisition function that involves sampling an input with probability proportional to the likelihood that this input is the optimal solution for the objective function. Particularly, TS samples a function $\tilde{f}_t$ from the predictive posterior distribution $p(f|\mathcal{D}_t)$ and then selects the point that maximizes this sampled function, i.e.:

$$\alpha_t^{\mathrm{TS}}(\boldsymbol{x}) = \tilde{f}_t(\boldsymbol{x}). \tag{2.26}$$

2.5.3 Performance metrics

The convergence of the sequential optimization is assessed by the two metrics *cumulative regret* and *simple regret*. The notion of regret comes from *multi-armed bandit* literature.

Cumulative regret: Let $\boldsymbol{x}^*$ be an optimizer of the unknown function, i.e., $\boldsymbol{x}^* \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. Then cumulative regret over a time horizon T is given as the total loss in the value of the objective function f at a sequence of observation points $\boldsymbol{x}_{1:T}$ compared to the global maximum:

$$R_T = \sum_{t=1}^{T} \Big[f(\boldsymbol{x}^*) - f(\boldsymbol{x}_t) \Big]. \tag{2.27}$$

Simple regret: The simple regret of an algorithm at any stopping time T is a measure of how far the value of the reported solution $\hat{\boldsymbol{x}}_T$ is from the true optimal solution $f(\boldsymbol{x}^*)$. In general, the reported point might differ from the point $\boldsymbol{x}_T$ acquired at the iteration T. Formally, the simple regret is defined as

$$r_T = f(\hat{\boldsymbol{x}}_T) - f(\boldsymbol{x}^*). \tag{2.28}$$

Intuitively, the simple regret measures how good the algorithm predicts the optimum at the iteration T. This metric is useful for problems in which the learner seeks to simultaneously minimize the number of expensive function evaluations T while still finding a solution that is close to the optimal one. Namely, for a given accuracy $\epsilon \geq 0$, we often wish to report a single "good" point $\hat{\boldsymbol{x}}_T \in \mathcal{X}$ after a total of T rounds, that satisfies the following

inequality:

$$f(\hat{\boldsymbol{x}}_T) \geq f(\boldsymbol{x}^*) - \epsilon. \tag{2.29}$$

Both metrics, i.e. cumulative regret and simple regret, are important for choosing solutions, and which one is preferred depends on the application at hand. For example, cumulative regret R_T might be of greater interest in online recommendation systems while reporting a single point with a high function value might be more suitable when tuning machine learning hyperparameters. We consider both metrics in the thesis.
The desired asymptotic behavior of a BO algorithm is a sublinear growth of R_T with T, also called *no regret* that implies vanishing average regret

$$\lim_{T \to \infty} \frac{R_T}{T} = 0.$$

Intuitively, this implies the existence of some t such that $f(\boldsymbol{x}_t)$ is arbitrarily close to the optimal value $f(\boldsymbol{x}^*)$. The regret analysis of BO algorithms typically consists of two main components: (1) confidence bound for random processes and (2) bound on the maximum information gain.

2.5.4 Review of some theoretical results

In the dissertation, particularly the methods proposed in Chapter 3 and Chapter 5, enjoy convergence guarantees under some regularity assumptions. The well-known upper confidence bound strategy with Gaussian processes is known to have regret guarantees in different settings [Sri+10; Abb12; CG17]. We provide one of the main regret results for the Bayesian setting from [Sri+10] below:

Theorem 1 ([Sri+10], Theorem 1)**.** *Let the objective $f \sim \mathcal{GP}(0, \kappa)$ and $\mathcal{X}$ be finite. Pick $\delta \in (0, 1)$ and let β_t be defined as in Lemma 1. When running GP-UCB Eq. (2.19) with the prior $\mathcal{GP}(0, \kappa)$ and known noise model, the following holds with probability at least $1 - \delta$ for any $T \geq 1$:*

$$R_T \leq \sqrt{C_1 T \beta_T \gamma_T}, \tag{2.30}$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

In the case of Thomson sampling, the frequentist regret guarantees for multi-armed bandits are known in the linear case [AL+17] (Theorem 1) and for RKHS function [CG17].

2.5.5 In the context of sequential decision-making research

To this point, we have considered the Bayesian formalism being used for Bayesian optimization, i.e., optimizing the unknown function of interest $f : \mathcal{X} \to \mathbb{R}$. Here we talk about closely-related setups of sequential decision-making, namely multi-armed bandits and reinforcement learning, and discuss BO within their context. The motivation to introduce those comes from the rich cross-connections between the setups and, thus, the methods to approach them. Consequently, the related literature is frequently cited and discussed

throughout the dissertation. Many successful BO approaches such as kernelized methods [Sri+10; CG17] are inspired by seminal works in the bandits literature, e.g., [LR85a; AL+17; AG13]. Some of the BO methods in this dissertation are also inspired by theoretically justified approaches from bandits literature. We do not delve into much detail about the setups here, and for a more in-depth treatment, refer to, for example, [SB18; LS20; Sli19].

Reinforcement learning (RL) deals with the problem of teaching an agent to make a sequence of decisions in an environment in order to maximize a reward. The agent learns through trial and error, receiving positive or negative feedback in the form of rewards or penalties for its actions. The mathematical formulation of reinforcement learning can be given via the (e.g., discrete-time) *Markov decision process* – a stochastic process consisting of a set of states $\mathcal{S}$, actions $\mathcal{A}$, and transition probabilities between states $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, together with a reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and a discount factor $\gamma \in (0, 1)$ that determines the importance of future rewards in relation to current rewards. The goal is to maximize the *expected cumulative return*, i.e., the total discounted reward accumulated over the interactions. To this end, it employs a control policy $\pi : \mathcal{S} \to \mathcal{A}$ that assigns a probability of taking a particular action in some state. Obviously, different policies yield different state trajectories and, therefore, different rewards. To maximize the expected cumulative reward, the agents should balance the exploration-exploitation trade-off.

The multi-armed bandit (MAB) problem is a classic problem that involves an agent trying to maximize its reward by choosing between multiple options, so-called arms, each of which has an associated probability distribution of rewards. The multi-armed bandit problem can be viewed as a simple reinforcement learning problem, in which there is no variable state, i.e., $|\mathcal{S}| = 1$, while in general RL, the action choice can influence transitions between the states. In this case, the reward function is the expected reward for each arm, and the agent's policy is the probability of selecting each arm.

In Bayesian optimization, like in the multi-armed bandit problem, there is no state transition, i.e., $|\mathcal{S}| = 1$. Compared to BO, however, the classical MAB formulation assumes no correlation between the arms, i.e., sampling one arm says nothing about the other. For example, this shared information in BO is treated by the Gaussian process's kernel that encodes the correlation between the arms. Similarly to RL and MAB, we seek the exploration-exploitation trade-off that naturally requires proper uncertainty quantification to decide on action. The latter is performed via acquisition policy, designated by the acquisition function in the case of Bayesian optimization. In contrast to RL, where the ultimate goal is maximizing cumulative (discounted) reward, BO also addresses problems where not only cumulative performance matters but a *single solution* must be provided by the end of the optimization, as discussed in Section 2.5.3. Finally, Bayesian optimization is powerful in small-data regimes where we do not have at least millions of data points but tens; that might not always be the case for the general RL framework.

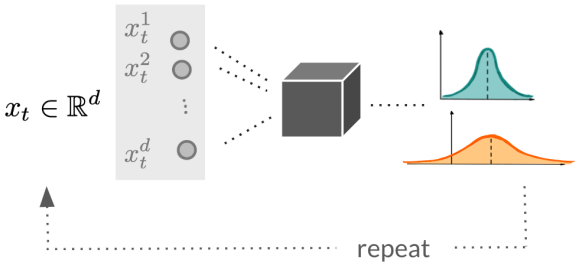
Conclusion

In conclusion, this chapter has provided the necessary mathematical background and literature review relevant to the thesis. The first part of the chapter introduced Bayesian

inference and types of uncertainties, Gaussian processes, reproducing kernel Hilbert spaces, and information capacity. The second part of the chapter focused on Bayesian optimization and described the main building blocks and performance metrics, along with existing theoretical results and open challenges. This overview sets the foundation for addressing the global optimization problems that we consider in the following chapters of the thesis.

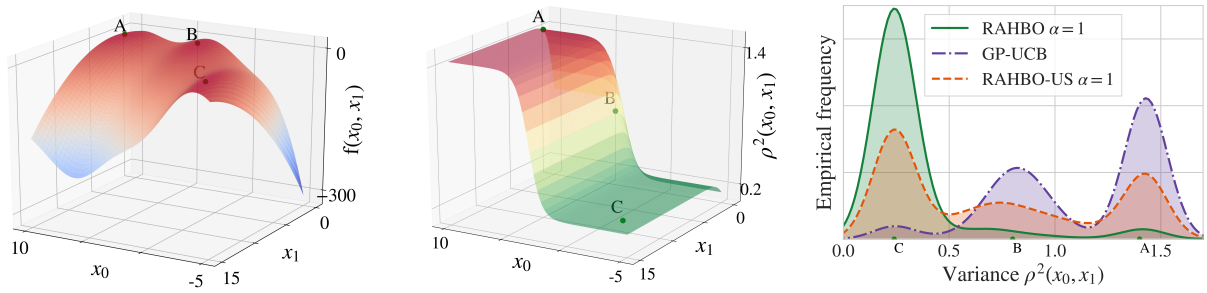
CHAPTER 3

Risk-averse decision-making



Many black-box optimization tasks arising in high-stakes applications require risk-averse decisions. Bayesian optimization is a powerful framework for optimizing costly black-box functions from noisy zeroth-order evaluations. Classical BO approaches are typically *risk-neutral* as they seek to optimize the expected function value only. However, optimization without taking input noise input noise can lead to catastrophic decisions when subjected to input noise at the implementation stage. For example, when designing a new drug, we might prefer compounds that would perform well not only on average over the population but would minimize the variance of the worse results. Similarly, when selecting hyperparameters of a machine learning algorithm, we might prefer configurations that lead to slightly higher test errors but at the same time lead to smaller variance.

In the chapter, we study attempts to make Bayesian optimization risk-aware and risk-averse. On the methodological side, we generalize BO to trade off mean and input-dependent noise variance when sequentially querying points and outputting final solutions. We introduce a practical setting where *both* the black-box objective and input-dependent noise variance are *unknown* a priori, and the learner needs to estimate them on the fly. We propose a novel optimistic risk-averse algorithm that makes sequential decisions by simultaneously balancing between *exploration* (learning about uncertain actions), *exploitation* (choosing actions that lead to high gains), and *risk* (avoiding unreliable actions). This chapter is based on our paper "Risk-averse Heteroscedastic Bayesian Optimization" [Mak+21b].



(a) Unknown objective f **(b)** Unknown variance ρ^2 **(c)** Histogram of variance

Figure 3.1: When there is a choice between identical optima with different noise levels, standard BO tends to query points corresponding to higher noise. (a) The unknown objective with 3 global maxima marked as (A, B, C); (b) Heteroscedastic noise variance over the same domain: the noise level at (A, B, C) varies according to the sigmoid function; (c) Empirical variance distribution at all points acquired during BO procedure (over 9 experiments with different seeds). The three bumps correspond to the three global optima with different noise variances. RAHBO dominates in choosing the risk-averse optimum, consequently yielding lower risk-averse regret in Figure 3.5a.

3.1 Risk-aversion in sequential decision-making

While the focus of standard BO approaches is mainly on trading-off exploration vs. exploitation and optimizing for the expected performance, in this work, we additionally focus on the risk that is involved when working with noisy objectives, as illustrated in Figure 3.1. Several works have recently considered robust and risk-averse aspects in BO. Their central focus is on designing robust strategies and protecting against the change/shift in uncontrollable covariates. They study various notions including worst-case robustness [Bog+18], distributional robustness [Kir+20; Ngu+20a], robust mixed strategies [Ses+20] and other notions of risk-aversion [IIT21; Cak+20; Ngu+21b], and while some of them report robust regret guarantees, their focus is primarily on the robustness in the homoscedastic GP setting. Instead, in our setting, we account for the risk that comes from the realization of random noise with *unknown* distribution. Rather than optimizing the expected performance, in our risk-averse setting, we prefer inputs with *lower variance*. To this end, we incorporate the learning of the noise distribution into the optimization procedure via a *mean-variance* objective. The closest to our setting is risk-aversion with respect to noise in multi-armed bandits [SLM12]. Their approach, however, fails to exploit correlation in rewards among similar arms.

The robust decision-making in BO is developing rapidly bringing new methods such as Bayesian optimization of risk measures [Ngu+21a] or multi-objective BO method that is robust to input noise [Dau+22]. The awareness of risk has been the focus of the research in general sequential-decision making problems beyond BO, such as in multi-armed bandits [VZ16; SLM12; GST13], and in RL [MT11; MA12]. The mean-variance approach in MDPs [MT11] formalizes risk-aware reinforcement learning as a multi-objective RL. In [MA12], the authors proceed by considering an exponential utility function, that (when rewritten as Taylor expansion) covers all the moments of the desired utility function (e.g., mean and variance). Fei, Yang, Chen, Wang, and Xie [Fei+20] demonstrate the first regret analysis

of the method that implements risk-sensitive optimism in the face of uncertainty for the exponential utility function in RL.

Objective	**Probabilistic model** (noise assumptions)
$\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$	homoscedastic sub-Gaussian
BO of risk measures [Cak+20; IIT21]:	
$\max_{\boldsymbol{x}\in\mathcal{X}} \text{CVaR}_\alpha f(\boldsymbol{x}, c)$	
$\max_{\boldsymbol{x}\in\mathcal{X}} \text{VaR}_\alpha f(\boldsymbol{x}, c)$	
Distributionally robust BO [Kir+20; Ngu+20b]:	heavy-tailed [RG19b]
$\max_{\boldsymbol{x}\in\mathcal{X}} \max_{\mathcal{Q}\in\mathcal{U}} \mathbb{E}_{c\in\mathcal{Q}}[f(\boldsymbol{x}, c)]$	
Worst-case robust BO [Bog+18; Ngu+20b]:	
$\max_{\boldsymbol{x}\in\mathcal{X}} \min_{c\in\Delta} f(\boldsymbol{x}, c)$	
Aversness to noise [Mak+21b]:	heteroscedastic sub-Gaussian [KK18; BGL18]
$\max_{\boldsymbol{x}\in\mathcal{X}} [f(\boldsymbol{x}) - \alpha\rho^2(\boldsymbol{x})]$	

Table 3.1: Research directions in risk-aware Bayesian optimization.

Our Contributions. We propose a novel algorithm that generalizes BO to *trade mean and input-dependent variance* of the objective, both of which we assume to be unknown a priori. In particular, we introduce risk-averse heteroscedastic Bayesian optimization algorithm (RAHBO) that aims to identify a solution with high return and low noise variance, while learning the noise distribution on the fly. To this end, we model both expectation and variance as (unknown) RKHS functions and propose a novel risk-aware acquisition function. The core idea of our approach is to combine the classical optimistic principle and careful estimation of the confidence intervals to trade off the expectation and uncertainty of the mean-variance objective function.

In our theoretical analysis, we establish rigorous sublinear regret guarantees for our algorithm (Theorem 1). These results also provide a non-trivial trade-off between the sample complexity and a better estimation of the noise model. We also provide a robust reporting rule and the number of samples required to output a single near-optimal risk-averse solution (Corollary 1). In Section 3.4, we demonstrate the effectiveness of RAHBO on synthetic benchmarks, as well as on hyperparameter tuning tasks for the Swiss free-electron laser and a machine learning model.

3.2 Problem Formulation

Let $\mathcal{X}$ be a given compact set of inputs ($\mathcal{X} \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$). We consider a problem of sequentially interacting with a fixed and unknown objective $f : \mathcal{X} \to \mathbb{R}$. At every round of this procedure, the learner selects an action $\boldsymbol{x}_t \in \mathcal{X}$, and obtains a noisy observation

$$y_t = f(\boldsymbol{x}_t) + \xi(\boldsymbol{x}_t), \tag{3.1}$$

where $\xi(\boldsymbol{x}_t)$ is zero-mean noise independent across different time steps t. In this work, we consider sub-Gaussian heteroscedastic noise that depends on the query location.

Optimization objective. Unlike the previous works that mostly focus on sequential optimization of f in the homoscedastic noise case, in this work, we consider the trade-off between risk and return in the heteroscedastic case. While there exist a number of risk-averse objectives, we consider the simple and frequently used mean-variance objective (MV) [SLM12]. Here, the objective value at $\boldsymbol{x} \in \mathcal{X}$ is a trade-off between the (mean) return $f(\boldsymbol{x})$ and the risk expressed by its variance-proxy $\rho^2(x)$:

$$\mathrm{MV}(\boldsymbol{x}) = f(\boldsymbol{x}) - \alpha\rho^2(\boldsymbol{x}), \tag{3.2}$$

where $\alpha \geq 0$ is a so-called *coefficient of absolute risk tolerance*. In this work, we assume α is fixed and known to the learner. In the case of $\alpha = 0$, maximizing $\mathrm{MV}(\boldsymbol{x})$ coincides with the standard BO objective.

Performance metrics. We aim to construct a sequence of input evaluations $\boldsymbol{x}_t$ that eventually maximizes the risk-averse objective $\mathrm{MV}(\boldsymbol{x})$. To assess this convergence, we consider two metrics. The first metric corresponds to the notion of cumulative regret similar to the one used in standard BO and bandits Eq. (2.27). Here, the learner's goal is to maximize its risk-averse cumulative reward over a time horizon T, or equivalently minimize its *risk-averse cumulative regret*:

$$R_T = \sum_{t=1}^{T} \Big[\mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_t) \Big], \tag{3.3}$$

where $\boldsymbol{x}^* \in \arg\max_{\boldsymbol{x}\in\mathcal{X}} \mathrm{MV}(\boldsymbol{x})$.

The second metric is used when the learner seeks to simultaneously minimize the number of expensive function evaluations T. Namely, for a given accuracy $\epsilon \geq 0$, we report a single "good" risk-averse point $\hat{\boldsymbol{x}}_T \in \mathcal{X}$ after a total of T rounds, that satisfies:

$$\mathrm{MV}(\hat{\boldsymbol{x}}_T) \geq \mathrm{MV}(\boldsymbol{x}^*) - \epsilon. \tag{3.4}$$

Regularity assumptions. We consider standard smoothness assumptions [Sri+10; Bog+18] when it comes to the unknown function $f : \mathcal{X} \to \mathbb{R}$. In particular, we assume that $f(\cdot)$ belongs to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_\kappa$, i.e., $f \in \mathcal{H}_\kappa$, induced by a kernel function $\kappa(\cdot,\cdot)$. We also assume that $\kappa(\boldsymbol{x},\boldsymbol{x}') \leq 1$ for every $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}$. Moreover, the RKHS norm of $f(\cdot)$ is assumed to be bounded $\|f\|_\kappa \leq \mathcal{B}_f$ for some fixed constant $\mathcal{B}_f > 0$. We assume that the noise $\xi(\boldsymbol{x})$ is $\rho(\boldsymbol{x})$–sub-Gaussian with variance-proxy $\rho^2(\boldsymbol{x})$ uniformly bounded $\rho(\boldsymbol{x}) \in [\underline{\varrho}, \bar{\varrho}]$ for some constant values $\bar{\varrho} \geq \underline{\varrho} > 0$.

3.3 The RAHBO Algorithm

We first recall the Gaussian process (GP) based framework for sequential learning of RKHS functions from observations with heteroscedastic noise. Then, in Section 3.3.2, we consider a simple risk-averse Bayesian optimization problem with *known* variance-proxy, and later

on in Section 3.3.3, we focus on our main problem setting in which the variance-proxy is *unknown*.

3.3.1 Bayesian optimization with heteroscedastic noise

Before addressing the risk-averse objective, we briefly recall the standard GP-UCB algorithm ([Sri+10]) in the setting of heteroscedastic sub-Gaussian noise. The regularity assumptions permit the construction of confidence bounds via GP model. Particularly, to decide which point to query at every round, GP-UCB uses the posterior GP mean and variance denoted by $\mu_t(\cdot)$ and $\sigma_t^2(\cdot)$, respectively. They are computed based on the previous measurements $y_{1:t} = [y_1, \dots, y_t]^\top$ and the given kernel $\kappa(\cdot, \cdot)$:

$$\mu_t(\boldsymbol{x}) = \kappa_t(\boldsymbol{x})^T (K_t + \lambda \Sigma_t)^{-1} y_{1:t}, \tag{3.5}$$

$$\sigma_t^2(\boldsymbol{x}) = \frac{1}{\lambda}\big(\kappa(\boldsymbol{x}, \boldsymbol{x}) - \kappa_t(\boldsymbol{x})^\top (K_t + \lambda \Sigma_t)^{-1} \kappa_t(\boldsymbol{x})\big), \tag{3.6}$$

where $\Sigma_t := \operatorname{diag}(\rho^2(\boldsymbol{x}_1), \dots, \rho^2(\boldsymbol{x}_t))$, $(K_t)_{i,j} = \kappa(\boldsymbol{x}_i, x_j)$, $\kappa_t(\boldsymbol{x})^T = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \dots, \kappa(\boldsymbol{x}_t, \boldsymbol{x})]^T$, $\lambda > 0$ and prior modelling assumptions are $\xi(\cdot) \sim \mathcal{N}(0, \rho^2(\cdot))$ and $f \sim GP(0, \lambda^{-1}\kappa)$.

At time t, GP-UCB maximizes the upper confidence bound of $f(\cdot)$, i.e.,

$$\boldsymbol{x}_t \in \arg\max_{x \in \mathcal{X}} \underbrace{\mu_{t-1}(\boldsymbol{x}) + \beta_t \sigma_{t-1}(\boldsymbol{x})}_{=:\mathrm{ucb}_t^f(\boldsymbol{x})}. \tag{3.7}$$

If the noise $\xi_t(\boldsymbol{x}_t)$ is heteroscedastic and $\rho(\boldsymbol{x}_t)$-sub-Gaussian, the following confidence bounds hold:

Lemma 3 (Lemma 7 in [KK18]). *Let $f \in \mathcal{H}_\kappa$, and $\mu_t(\cdot)$ and $\sigma_t^2(\cdot)$ be defined as in (3.5) and (3.6) with $\lambda > 0$. Assume that the observations $(\boldsymbol{x}_t, y_t)_{t \geq 1}$ satisfy Eq. (3.1). Then the following holds for all $t \geq 1$ and $x \in \mathcal{X}$ with probability at least $1 - \delta$:*

$$|\mu_{t-1}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \underbrace{\left(\sqrt{2 \ln\left(\frac{\det(\lambda \Sigma_t + K_t)^{1/2}}{\delta \det(\lambda \Sigma_t)^{1/2}}\right)} + \sqrt{\lambda}\|f\|_\kappa\right)}_{:=\beta_t} \sigma_{t-1}(\boldsymbol{x}). \tag{3.8}$$

Here, β_t stands for the parameter that balances between exploration vs. exploitation and ensures the validity of confidence bounds. The analogous concentration inequalities in case of homoscedastic noise were considered in [Abb12; CG17; Sri+10].

Failure of GP-UCB in the risk-averse setting. GP-UCB is guaranteed to achieve sublinear cumulative regret with high probability in the risk-neutral (homoscedastic/heteroscedastic) BO setting [Sri+10; CG17]. However, for the risk-averse setting in Eq. (3.2), the maximizers $x^* \in \arg\max_{x \in \mathcal{X}} \mathrm{MV}(\boldsymbol{x})$ and $x_f^* \in \arg\max_{x \in \mathcal{X}} f(\boldsymbol{x})$ might not coincide, and consequently, $\mathrm{MV}(\boldsymbol{x}^*)$ can be significantly larger than $\mathrm{MV}(\boldsymbol{x}_f^*)$. This is illustrated in Figure 3.1, where GP-UCB most frequently chooses optimum A of the highest risk.

3.3.2 Warm up: Known variance-proxy

We remedy the previous issue with GP-UCB by proposing a natural *Risk-averse Heteroscedastic* BO (RAHBO) in case of the known variance-proxy $\rho^2(\cdot)$. At each round t, RAHBO chooses the action:

$$\boldsymbol{x}_t \in \arg\max_{x\in\mathcal{X}} \ \mu_{t-1}(\boldsymbol{x}) + \beta_t\sigma_{t-1}(\boldsymbol{x}) - \alpha\rho^2(\boldsymbol{x}), \tag{3.9}$$

where β_t is from Lemma 3 and α is from Eq. (3.2). In the next section, we further relax the assumption of the variance-proxy and consider a more practical setting when $\rho^2(\cdot)$ is unknown to the learner. For the current setting, the performance of RAHBO is formally captured in the following proposition.

Proposition 1. *Consider any $f \in \mathcal{H}_\kappa$ with $\|f\|_\kappa \leq \mathcal{B}_f$ and sampling model from Eq. (3.1) with known variance-proxy $\rho^2(\boldsymbol{x})$. Let $\{\beta_t\}_{t=1}^T$ be set as in Lemma 3 with $\lambda = 1$. Then, with probability at least $1-\delta$,* RAHBO *attains cumulative risk-averse regret $R_T = \mathcal{O}\big(\beta_T\sqrt{T\gamma_T(\bar{\varrho}^2+1)}\big)$.*

Here, γ_T denotes the maximum information gain [Sri+10] at time T defined in Eq. (2.16) via mutual information $I(y_{1:T}, f_{1:T})$ between evaluations $y_{1:T}$ and $f_{1:T} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_t)]^\top$ at points $A \subset \mathcal{X}$. In case of heteroscedastic noise, the maximum information gain can be written as follows:

$$\gamma_T := \max_{A\subset\mathcal{X},\ |A|=T} I(y_{1:T}, f_{1:T}), \tag{3.10}$$

$$\text{where} \quad I(y_{1:T}, f_{1:T}) = \frac{1}{2}\sum_{t=1}^{T} \ln\left(1 + \tfrac{\sigma^2_{t-1}(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right) \tag{3.11}$$

We provide the derivation in Section B.1.1. The upper bounds on γ_T are provided in [Sri+10] widely used kernels. These upper bounds typically scale sublinearly in T; for linear kernel $\gamma_T = \mathcal{O}(d\log T)$, and in case of squared exponential kernel $\gamma_T = \mathcal{O}(d(\log T)^{d+1})$. While these bounds are derived assuming the homoscedastic GP setting with some fixed constant noise variance, we show (in Section B.1.3) that the same rates (up to a multiplicative constant factor) apply in the heteroscedastic case.

3.3.3 RAHBO for unknown variance-proxy

In the case of unknown variance-proxy, the confidence bounds for the unknown $f(\boldsymbol{x})$ in Lemma 3 can not be readily used, and we construct new ones on the combined mean-variance objective. To learn about the unknown $\rho^2(\boldsymbol{x})$, we make some further assumptions.

Assumption 1. *The variance-proxy $\rho^2(\boldsymbol{x})$ belongs to an RKHS induced by some kernel κ^{var}, i.e., $\rho^2 \in \mathcal{H}_{\kappa^{var}}$, and its RKHS norm is bounded $\|\rho^2\|_{\kappa^{var}} \leq \mathcal{B}_{var}$ for some finite $\mathcal{B}_{var} > 0$. Moreover, the noise $\xi(\boldsymbol{x})$ in Eq. (3.1) is strictly $\rho(\boldsymbol{x})$–sub-Gaussian, i.e., $\mathbb{V}ar[\xi(\boldsymbol{x})] = \rho^2(\boldsymbol{x})$ for every $x \in \mathcal{X}$.*

Algorithm 2 Risk-averse Heteroscedastic Bayesian Optimization (RAHBO)

Require: Parameters $\alpha, \{\beta_t, \beta_t^{var}\}_{t\geq 1}, \lambda$, k, Prior $\mu_0^f = \mu_0^{var} = 0$, Kernel functions κ, κ^{var}

for $t = 1, 2, \ldots$ **do**
  Construct confidence bounds $\mathrm{ucb}_t^{var}(\cdot)$ and $\mathrm{lcb}_t^{var}(\cdot)$ as in (3.14) and (3.15)
  Construct $\mathrm{ucb}_t^{f}(\cdot)$ as in Eq. (3.7)
  Select $\boldsymbol{x}_t \in \arg\max_{x\in\mathcal{X}} \mathrm{ucb}_t^f(\boldsymbol{x}) - \alpha\, \mathrm{lcb}_t^{var}(\boldsymbol{x})$
  Observe k samples: $y_i(\boldsymbol{x}_t) = f(\boldsymbol{x}_t) + \xi_i(\boldsymbol{x}_t) \quad$ for every $\quad i \in [k]$
  Use samples $\{y_i(\boldsymbol{x}_t)\}_{i=1}^k$ to compute sample mean $\hat{m}_k(\boldsymbol{x}_t)$ and variance $\hat{s}_k^2(\boldsymbol{x}_t)$ as in Eq. (5.2)
  Use $\boldsymbol{x}_t, \hat{s}_k^2(\boldsymbol{x}_t)$ to update posterior $\mu_t^{var}(\cdot)$ and $\sigma_t^{var}(\cdot)$ as in (B.15) and (B.16)
  Use $\mathrm{ucb}_t^{var}(\cdot)$ to compute $\hat{\Sigma}_t$ as in Eq. (3.16)
  Use $\boldsymbol{x}_t, \hat{m}_k(\boldsymbol{x}_t)$ and $\hat{\Sigma}_t$ to update posterior $\mu_t(\cdot)$ and $\sigma_t(\cdot)$ as in (3.5) and (3.6)
end for

As a consequence of our previous assumption, we can now focus on estimating the variance since $\mathbb{V}ar[\xi(\cdot)]$ and $\rho^2(\cdot)$ coincide. In particular, to estimate $\mathbb{V}ar[\xi(\cdot)]$ we consider a *repeated experiment setting*, where for each $\boldsymbol{x}_t$ we collect $k > 1$ evaluations $\{y_i(\boldsymbol{x}_t)\}_{i=1}^k$, $y_i(\boldsymbol{x}_t) = f(\boldsymbol{x}_t) + \xi_i(\boldsymbol{x}_t)$. Then, the sample mean and variance of $\xi(\boldsymbol{x}_t)$ are given as:

$$\hat{m}_k(\boldsymbol{x}_t) = \frac{1}{k}\sum_{i=1}^{k} y_i(\boldsymbol{x}_t) \quad \text{and} \quad \hat{s}_k^2(\boldsymbol{x}_t) = \frac{1}{k-1}\sum_{i=1}^{k} \big(y_i(\boldsymbol{x}_t) - \hat{m}_k(\boldsymbol{x}_t)\big)^2. \tag{3.12}$$

The key idea is that for strictly sub-Gaussian noise $\xi(\boldsymbol{x})$, $\hat{s}_{1:t}^2 = [\hat{s}_k^2(\boldsymbol{x}_1), \ldots, \hat{s}_k^2(\boldsymbol{x}_t)]^\top$ yields *unbiased, but noisy* evaluations of the unknown variance-proxy $\rho_{1:t}^2 = [\rho^2(\boldsymbol{x}_1), \ldots, \rho^2(\boldsymbol{x}_t)]^\top$, i.e.,

$$\hat{s}_k^2(\boldsymbol{x}_t) = \rho^2(\boldsymbol{x}_t) + \eta(\boldsymbol{x}_t) \tag{3.13}$$

with zero-mean noise $\eta(\boldsymbol{x}_t)$. In order to efficiently estimate $\rho^2(\cdot)$, we need an additional assumption.

Assumption 2. *The noise $\eta(\boldsymbol{x})$ in Eq. (3.13) is $\rho_\eta(\boldsymbol{x})$–sub-Gaussian with known $\rho_\eta^2(\boldsymbol{x})$ and the realizations $\{\eta(\boldsymbol{x}_t)\}_{t\geq 1}$ are independent between t.*

We note that a similar assumption is made in [SLM12] in the multi-armed bandit setting. The fact that $\rho_\eta^2(\cdot)$ is known is rather mild as Assumption 1 allows controlling its value. For example, in case of strictly sub-Gaussian $\eta(\boldsymbol{x})$ we show (in Section B.2) that $\mathbb{V}ar[\eta(\cdot)] = \rho_\eta^2(\cdot) \leq 2\rho^4(\cdot)/(k-1)$. Then, given that $\rho^2(\cdot) \leq \bar{\varrho}^2$, we can utilize the following (rather conservative) bound as a variance-proxy, i.e., $\rho_\eta^2(\boldsymbol{x}) = 2\bar{\varrho}^4/(k-1)$.

RAHBO algorithm. We present our Risk-averse Heteroscedastic BO approach for unknown variance-proxy in Algorithm 2. Our method relies on building the following two GP models.

Firstly, we use sample variance evaluations $\hat{s}_{1:t}^2$ to construct a GP model for $\rho^2(\cdot)$. The corresponding $\mu_{t-1}^{var}(\cdot)$ and $\sigma_{t-1}^{var}(\cdot)$ are computed as in (3.5) and (3.6) by using kernel κ^{var}, variance-proxy $\rho_\eta^2(\cdot)$ and noisy observations $\hat{s}_{1:t}^2$. Consequently, we build the upper and lower confidence bounds $\mathrm{ucb}_t^{var}(\cdot)$ and $\mathrm{lcb}_t^{var}(\cdot)$ of the variance-proxy $\rho^2(\cdot)$ and we set β_t^{var}

according to Lemma 3:

$$\mathrm{ucb}_t^{var}(\boldsymbol{x}) := \mu_{t-1}^{var}(\boldsymbol{x}) + \beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}), \tag{3.14}$$

$$\mathrm{lcb}_t^{var}(\boldsymbol{x}) := \mu_{t-1}^{var}(\boldsymbol{x}) - \beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}). \tag{3.15}$$

Secondly, we use sample mean evaluations $\hat{m}_{1:t} = [\hat{m}_k(\boldsymbol{x}_1), \dots, \hat{m}_k(\boldsymbol{x}_t)]^\top$ to construct a GP model for $f(\cdot)$. The mean $\mu_t(\cdot)$ and variance $\sigma_t^2(\cdot)$ in (3.6) and (3.5), however, rely on the unknown variance-proxy $\rho^2(\cdot)$ in Σ_t, an we thus use its upper confidence bound $\mathrm{ucb}_t^{var}(\cdot)$ truncated with $\bar{\varrho}^2$:

$$\hat{\Sigma}_t := \tfrac{1}{k}\mathrm{diag}\big(\min\{\mathrm{ucb}_t^{var}(\boldsymbol{x}_1), \bar{\varrho}^2\}, \dots, \min\{\mathrm{ucb}_t^{var}(\boldsymbol{x}_t), \bar{\varrho}^2\}\big), \tag{3.16}$$

where $\hat{\Sigma}_t$ is corrected by k since every evaluation in $\hat{m}_{1:t}$ is an average over k samples. This substitution of the unknown variance-proxy by its conservative estimate guarantees that the confidence bounds $\mathrm{ucb}_t^f(\boldsymbol{x}) := \mu_{t-1}(\boldsymbol{x}) + \beta_t\sigma_{t-1}(\boldsymbol{x})$ on f also hold with high probability (conditioning on the confidence bounds for $\rho(\cdot)$ holding true; see Section B.3 for more details).

Finally, we define the acquisition function as $\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}) := \mathrm{ucb}_t^f(\boldsymbol{x}) - \alpha\mathrm{lcb}_t^{var}(\boldsymbol{x})$, i.e., selecting $\boldsymbol{x}_t \in \arg\max_{x\in\mathcal{X}} \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x})$ at each round t.

The proposed algorithm leads to new maximum information gains $\hat{\gamma}_T = \max_A I(\hat{m}_{1:T}, f_{1:T})$ and $\Gamma_T = \max_A I(\hat{s}^2_{1:T}, \rho^2_{1:T})$ for sample mean $\hat{m}_{1:T}$ and sample variance $\hat{s}^2_{1:T}$ evaluations. The corresponding mutual information in $\hat{\gamma}_T$ and Γ_T is computed according to Eq. (3.11) for heteroscedastic noise with variance-proxy $\bar{\varrho}^2/k$ and ρ^2_η, respectively (see Section B.4). The performance of RAHBO is captured in the following theorem.

Theorem 1. *Consider any $f \in \mathcal{H}_\kappa$ with $\|f\|_\kappa \leq \mathcal{B}_f$ and sampling model in Eq. (3.1) with unknown variance-proxy $\rho^2(\boldsymbol{x})$ that satisfies Assumptions 1 and 2. Let $\{\boldsymbol{x}_t\}_{t=1}^T$ denote the set of actions chosen by* RAHBO *(Algorithm 2) over T rounds. Set $\{\beta_t\}_{t=1}^T$ and $\{\beta_t^{var}\}_{t=1}^T$ according to Lemma 3 with $\lambda = 1$, $\mathcal{R}^2 = \max_{x\in\mathcal{X}} \rho^2_\eta(\boldsymbol{x}_t)$ and $\rho(\cdot) \in [\underline{\varrho}, \bar{\varrho}]$. Then, the risk-averse cumulative regret R_T of* RAHBO *is bounded as follows:*

$$\Pr\left\{R_T \leq \beta_T k\sqrt{\frac{2T\hat{\gamma}_T}{\ln(1+k/\bar{\varrho}^2)}} + \alpha\beta_T^{var}k\sqrt{\frac{2T\Gamma_T}{\ln(1+\mathcal{R}^{-2})}}, \quad \forall T \geq 1\right\} \geq 1-\delta. \tag{3.17}$$

The risk-averse cumulative regret of RAHBO depends sublinearly on T for most of the popularly used kernels. This follows from the implicit sublinear dependence on T in β_T, β_T^{var} and $\hat{\gamma}_T, \Gamma_T$ (the bounds in case of heteroscedastic noise replicate the ones used in Proposition 1 as shown in Sections B.1.2 and B.1.3). Finally, the result of Theorem 1 provides a non-trivial trade-off for the number of repetitions k where larger k increases sample complexity but also leads to better estimation of the noise model. Furthermore, we obtain the bound for the number of rounds T required for identifying an ϵ-optimal point:

Corollary 1. *Consider the setup of Theorem 1. Let $A = \{\boldsymbol{x}_t\}_{t=1}^T$ denote actions selected by* RAHBO *over T rounds. Then, with probability at least $1-\delta$, the reported point $\hat{\boldsymbol{x}}_T := \arg\max_{\boldsymbol{x}_t\in A} \mathrm{lcb}_t^{MV}(\boldsymbol{x}_t)$, where $\mathrm{lcb}_t^{MV}(\boldsymbol{x}_t) = \mathrm{lcb}_t^f(\boldsymbol{x}) - \alpha\ \mathrm{ucb}_t^{var}(\boldsymbol{x})$, achieves ϵ-*

accuracy, i.e., $MV(\boldsymbol{x}^*) - MV(\hat{\boldsymbol{x}}_T) \leq \epsilon$, *after* $T \geq \frac{4\beta_T^2 \hat{\gamma}_T / \ln(1+k/\bar{\varrho}^2) + 4\alpha(\beta_t^{var})^2 \Gamma_T / \ln(1+\mathcal{R}^{-2})}{\epsilon^2}$ *rounds.*

The previous result demonstrates the sample complexity rates when a single risk-averse reported solution is required. We note that both Theorem 1 and Corollary 1 provide guarantees for choosing risk-averse solutions, and depending on application at hand, we might consider either one of the proposed performance metrics. We demonstrate use-cases for both in the following section.

3.4 Experiments

In this section, we experimentally validate RAHBO on two synthetic examples and two real hyperparameter tuning tasks, and compare it with the baselines. We provide an open-source implementation of our method.[1]

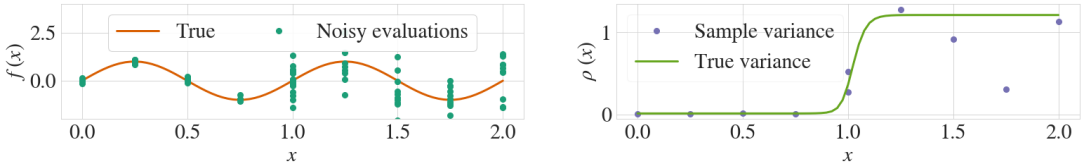
Baselines. We compare against two baselines: As the first baseline, we use GP-UCB with heteroscedastic noise as a standard risk-neutral algorithm that optimizes the unknown $f(\boldsymbol{x})$. As the second one, we consider a risk-averse baseline that uniformly learns variance-proxy $\rho^2(\boldsymbol{x})$ *before* the optimization procedure, in contrast to RAHBO which learns the variance-proxy on the fly. We call it RAHBO-US, standing for RAHBO with uncertainty sampling. It consists of two stages: (i) uniformly learning $\rho^2(\boldsymbol{x})$ via uncertainty sampling, (ii) GP-UCB applied to the mean-variance objective, in which instead of the unknown $\rho^2(\boldsymbol{x})$ we use the mean of the learned model. Note that RAHBO-US is the closest to the contextual BO setting [IIT21], where the context distribution is assumed to be known.

Experimental setup. At each iteration t, an algorithm queries a point x_t and observes sample mean and sample variance of k observations $\{y_i(\boldsymbol{x}_t)\}_{i=1}^k$. We use a heteroscedastic GP for modelling $f(\boldsymbol{x})$ and a homoscedastic GP for $\rho^2(\boldsymbol{x})$. We set $\lambda = 1$ and $\beta_t = 2$, which is commonly used in practice to improve performance over the theoretical results. Before the BO procedure, we determine the GP hyperparameters maximizing the marginal likelihood. To this end, we use initial points that are same for all the baselines and are chosen via Sobol sequence that generates low discrepancy quasi-random samples. We repeat each experiment several times, generating new initial points for every repetition. We use two metrics: (a) risk-averse cumulative regret R_t computed for the acquired inputs; (b) simple regret $\mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\hat{\boldsymbol{x}}_T)$ computed for inputs as reported via Corollary 1. For each metric, we report its mean $\pm$ two standard errors over the repetitions.

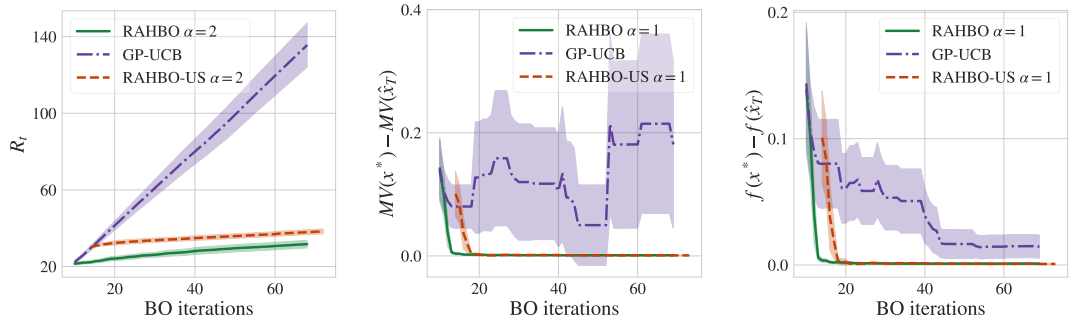
3.4.1 Synthetic experiments

Example function We first illustrate the methods performance on a sine function depicted in Figure 3.2a. This function has two global optimizers. We induce a heteroscedastic zero-mean Gaussian noise on the measurements. We use a sigmoid function for the noise variance, as depicted in Figure 3.2a, that induces small noise on $[0, 1]$ and higher noise on $(1, 2]$. We initialize the algorithms by selecting 10 inputs x at random and

[1] https://github.com/Avidereta/risk-averse-hetero-bo



(a) Illustration of the sine function (left) and noise variance (right)



(b) Cumulative regret **(c)** Suboptimality w.r.t. MV **(d)** Suboptimality w.r.t. f

Figure 3.2: (a) Unknown true objective along with noisy evaluations with varying noise level (left) and unknown true noise variance and its evaluations (right). (b) Cumulative regret. (c) Simple MV regret for reporting rule $\hat{\boldsymbol{x}}_T = \arg\max_{x_t} \text{lcb}_T(\boldsymbol{x}_t)$. (c) Simple regret $f(\boldsymbol{x}^*) - f(\hat{\boldsymbol{x}}_T)$ for the unknown function at the reported point $\hat{\boldsymbol{x}}_T$ from (d). RAHBO not only leads to strong results in terms of MV but also in terms of the mean objective $f(\boldsymbol{x})$.

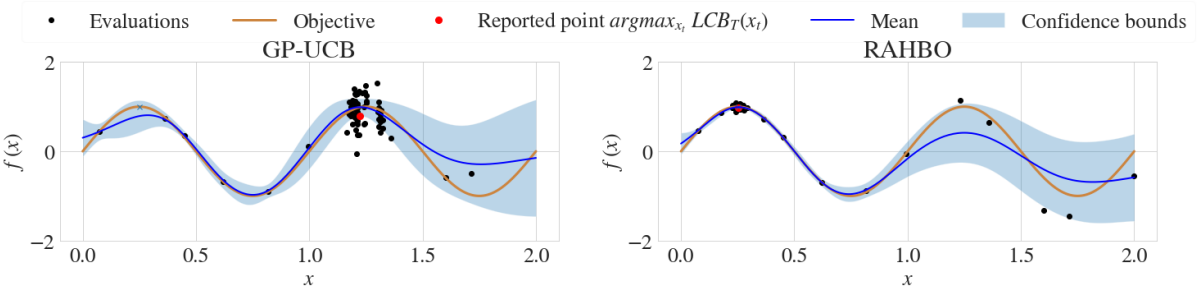


Figure 3.3: GP models fitted for GP-UCB (left) and RAHBO (right) for sine function. After initialization with the same sampled points, GP-UCB concentrates on the high-noise region whereas RAHBO prefers small variance. Additional plots are presented in Chapter C.

keep these points the same for all the algorithms. We use $k = 10$ samples at each chosen x_t. The number of acquisition rounds is $T = 60$. We repeat the experiment 30 times for each method and show their average performances in Figure 3.2.

Branin benchmark Next, we evaluate the methods on the (negated) Branin benchmark function in Figure 3.1a, achieving its optimum value $f^* = -0.4$ at $(-\pi, 12.3), (\pi, 2.3), (9.4, 2.5)$. The heteroscedastic variance function illustrated in Figure 3.1b defines different noise variances for the three optima. We initialize all algorithms by selecting 10 inputs. We use $k = 10$ samples to estimate the noise variance. The number of acquisition rounds is $T = 150$. We repeat BO 25 times and show the results in Figures 3.5a and 3.1c. Figure 3.1c provides more intuition behind the observed regret: UCB exploits the noisiest maxima the most, while RAHBO prefers smaller variance.

3.4.2 Tuning Swiss free-electron laser

In this experiment, we tune the parameters of Swiss X-ray free-electron laser (SwissFEL), an important scientific instrument that generates very short pulses of X-ray light and enables researchers to observe extremely fast processes. The main objective is to maximize the pulse energy measured by a gas detector, that is a time-consuming and repetitive task during the SwissFEL operation. Such (re-)tuning takes place while user experiments on SwissFEL are running, and thus cumulative regret is the metric of high importance in this application.

We use real SwissFEL measurements collected in [Kir+19] to train a neural network surrogate model, and use it to simulate the SwissFEL objective $f(\boldsymbol{x})$ for new parameter settings x. We similarly fit a model of the heteroscedastic variance by regressing the squared residuals via a GP model. Here, we focus on the calibration of the four most sensitive parameters.

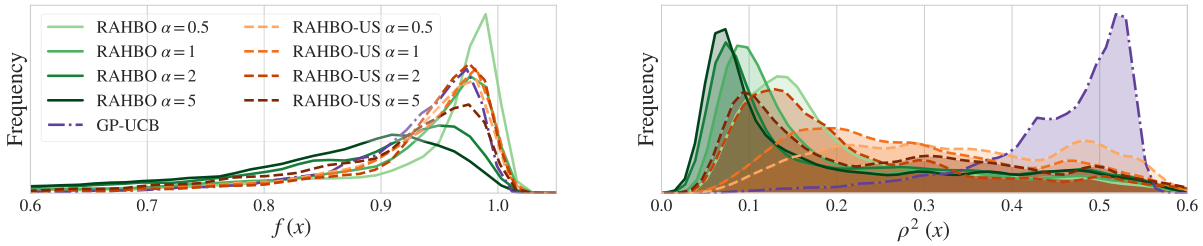
We report our comparison in Figure 3.4 where we also assess the effect of varying the coefficient of absolute risk tolerance α. We use 30 points to initialize the baselines and then perform 200 acquisition rounds. We repeat each experiment 15 times. In Figure 3.4a we plot the empirical frequency of the true (unknown to the methods) values $f(\boldsymbol{x}_t)$ and $\rho^2(\boldsymbol{x}_t)$ at the inputs x_t acquired by the methods. The empirical frequency for $\rho^2(\boldsymbol{x})$ illustrates the tendency of risk-neutral GP-UCB to query points with higher noise, while risk-averse achieves substantially reduced variance and minimal reduction in mean performance. Sometimes, risk-neutral GP-UCB also fails to succeed in querying points with the highest f-value. That tendency results in lower cumulative regret for RAHBO in Figures 3.4c and 3.4d. We also compare the performance of the reporting rule from Corollary 1 in Figure 3.4b, where we plot error bars with standard deviation both for $f(\hat{\boldsymbol{x}}_T)$ and $\rho^2(\hat{\boldsymbol{x}}_T)$ at the reported point $\hat{\boldsymbol{x}}_T$. As before, RAHBO drastically reduces the variance compared to GP-UCB, while having only slightly lower mean performance. Additional results are presented in Figure C.5 in the Appendix (see Chapter B).

3.4.3 Random Forest tuning

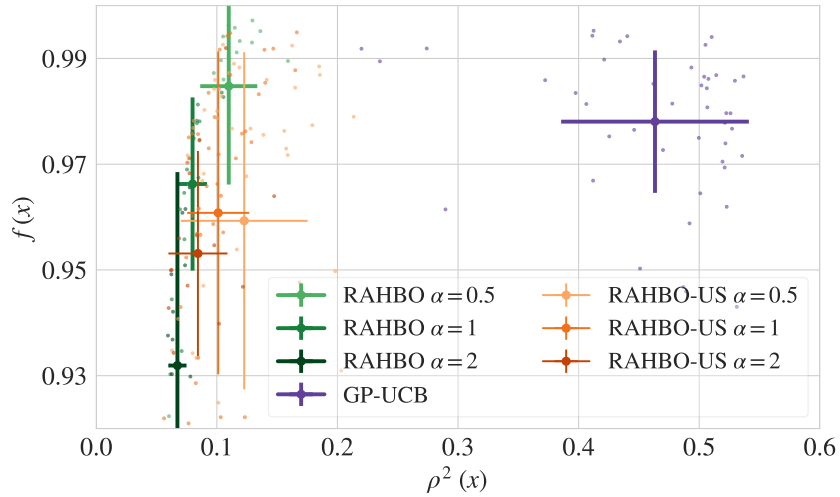
BO is widely used by cloud services for tuning machine learning hyperparameters and the resulting models might be then used in high-stakes applications such as credit scoring or fraud detection. In k-fold cross-validation, the average metric over the validation sets is optimized – a canonical example of the *repeated experiment setting* that we consider in the paper. High across-folds variance is a practical problem [Mak+21a] where the mean-variance approach might be beneficial.

In our experiment, we tune hyperparameters of a random forest classifier (RF) on a dataset of fraudulent credit card transactions [LB21].[2] It consists of 285k transactions with 29 features (processed due to confidentiality issues) that are distributed over time, and only 0.2% are fraud examples (see Appendix, Section C.2, for more details). The search space for the RF hyperparameters is also provided in Section C.2. We use the balanced accuracy score and 5 validation folds, i.e., $k = 5$, and each validation fold is

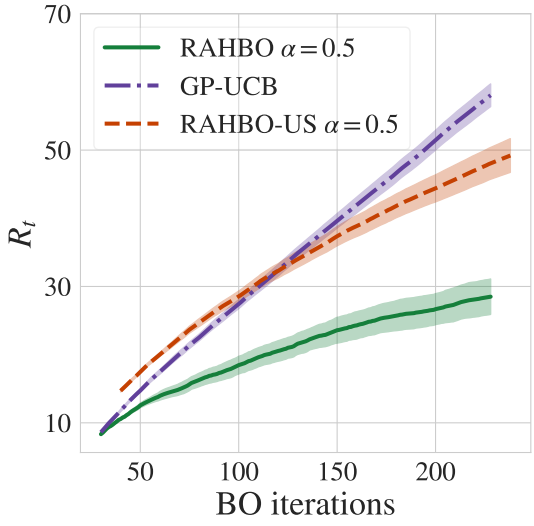
[2] https://www.kaggle.com/mlg-ulb/creditcardfraud



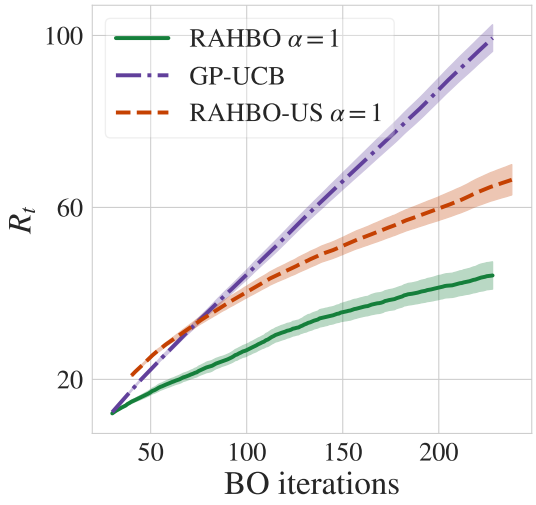
(a) Empirical distribution of true $f(\boldsymbol{x})$ (left) and $\rho^2(\boldsymbol{x})$ (right) for SwissFEL



(b) Mean-variance tradeoff



(c) Cum. regret ($\alpha = 0.5$)

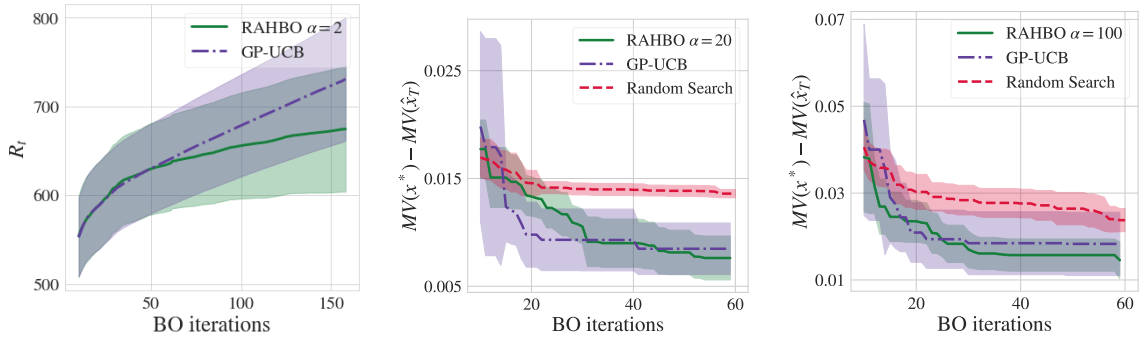


(d) Cum. regret ($\alpha = 1$)

Figure 3.4: Experimental results for SwissFEL: **(a)** Distributions of $f(\boldsymbol{x})$ and $\rho^2(\boldsymbol{x})$ for *all points* queried during the optimization. GP-UCB queries points with higher noise (but not necessarily high return f) in contrast to the risk-averse methods. **(b)** Mean $f(\hat{\boldsymbol{x}}_T)$ and variance $\rho^2(\hat{\boldsymbol{x}}_T)$ at the *reported* $\hat{\boldsymbol{x}}_T = \arg\max_{x_t} \mathrm{lcb}_T(\boldsymbol{x}_t)$: for each method, we repeat BO experiment 15 times (separate points) and plot corresponding standard deviation error bars. RAHBO reports solutions with reasonable mean-variance tradeoff, while GP-UCB produces solutions with high mean value but also high noise variance. **(c-d)** Cum. regret for $\alpha = 0.5$ and $\alpha = 1$ (see more in Section C.3).

shifted in time with respect to the training data. We seek not only high performance *on average* but also low variance across the validation folds that have different time shifts with respect to the training data.

We initialize the algorithms by selecting 10 hyperparameter settings and keep these points the same for all algorithms. We use Matérn 5/2 kernels with Automatic Relevance Discovery (ARD) and normalize the input features to the unit cube. The number of acquisition rounds in one experiment is 50 and we repeat each experiment 15 times. We demonstrate our results in Figures 3.5b and 3.5c where we plot mean $\pm$ 2 standard errors. While both RAHBO and GP-UCB perform comparably in terms of the mean error, its standard deviation for RAHBO is smaller.



(a) Branin benchmark **(b)** RF Tuning **(c)** RF Tuning

Figure 3.5: Branin: (a) Cumulative regret. **Random Forest**: (b-c) Simple regret for the reported $\hat{\boldsymbol{x}}_T = \arg\max_{x_t} MV(\boldsymbol{x}_t)$ for (b) $\alpha = 20$ and (c) $\alpha = 100$. While both methods have comparable mean, RAHBO has consistently lower variance.

3.5 Conclusion

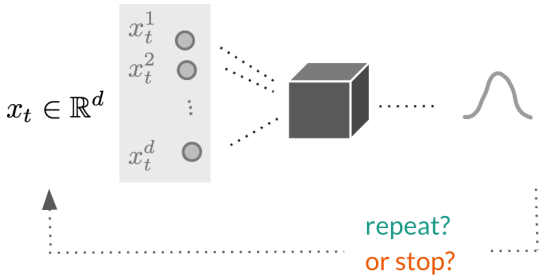
In this chapter, we generalize Bayesian optimization to the risk-averse setting and propose RAHBO algorithm aiming to find an input with both large expected return and small input-dependent noise variance. Both the mean objective and the variance are assumed to be unknown a priori and hence are estimated online. RAHBO is equipped with theoretical guarantees showing (under reasonable assumptions) sublinear dependence on the number of evaluation rounds T both for cumulative risk-averse regret and ϵ-accurate mean-variance metric. The empirical evaluation of the algorithm on synthetic benchmarks and hyperparameter tuning tasks demonstrate promising examples of heteroscedastic use cases benefiting from RAHBO.

CHAPTER 4

Computationally-effective Bayesian optimization

Bayesian optimization is particularly useful for expensive-to-evaluate functions. Improving the computational efficiency of BO is important in a wide range of applications because it allows us to optimize more complex functions or optimize them more quickly, which can lead to better solutions or faster turnaround times for optimization tasks. In this chapter, we explore two approaches that make BO more efficient in practice. First, we aim at saving computational resources by avoiding unnecessary evaluations and propose an *automatically determining when to stop BO* in Section 4.1. Second, we study the incorporation of multiple sources of information with different costs and accuracies and propose an *information-theoretic method for multi-fidelity BO* in Section 4.2. The methodological contributions of this chapter are particularly driven by the application needs arising in hyperparameter optimization of ML models and calibrating complex multi-agent transport systems simulators. We demonstrate the effectiveness of the proposed approaches through empirical examination in these specific contexts.

4.1 Automatic Termination of Bayesian optimization



While the performance of machine learning algorithms crucially depends on their hyperparameters, setting them correctly is typically a tedious and expensive task. Hyperparameter optimization (HPO) emerged as a new sub-field in machine learning that tries to automatically determine how to configure a machine learning model. One of the most successful strategies for HPO is Bayesian optimization which sequentially optimizes the predictive performance of a model configured with certain hyperparameters. BO

iteratively searches for better predictive performance via (i) training a probabilistic model on the evaluations of the models performance and (ii) selecting the most promising next hyperparameter candidate.

In practice, the quality of the solution found by BO heavily depends on a pre-defined budget, such as the number of BO iterations or wall-clock time. If this budget is too small, BO might result into hyperparameters of poor predictive performance. If the budget is too large, compute resources will be wasted. The latter can be especially fragile in HPO when one cannot fully reduce the discrepancy between the validation and test errors, thus resulting in *overfitting* as we show in our experiments. A naive approach suggests terminating BO if the best-found solution remains unchanged for some subsequent BO iterations. Though the idea is sensible, it might be challenging to define a suitable number, since it is a fixed, predetermined choice, that does not take the observed data into account. Another approach is to track the probability of improvement ([Lor+16]) or the expected improvement ([Ngu+17]), and stop the optimization process once it falls below a given threshold. However, determining this threshold may in practice be less intuitive than setting the number of iterations or the wall clock time. Instead of stopping BO completely, in [MRO18], it is proposed to switch to local optimization when the global regret is smaller than a pre-defined target. This condition can also be used to terminate BO early, but it comes with additional complexity such as identifying a (convex) region for local optimization and again a predefined budget.

Automatically terminating the sequential procedure of BO is a rather under-explored topic, in contrast to the more widely considered orthogonal direction of speeding up HPO via stopping the model training. A seminal idea there is to avoid the computation of low-performing hyperparameters, e.g., by learning curves ([SSA14]), multi-fidelity approach Hyperband ([Li+17]), its combination ([Kle+17]), and further modifications like Hyperband BOHB ([FKH18]), asynchronous Hyperband ([Li+20]) and its model based version ([Kle+20]). The profound distinction of our method with this line of works is in the problem setup: instead of stopping the training, our method aims to terminate the whole HPO process. This orthogonality allows for a combination of both ways of to achieve overall larger speed-ups.

Our contributions. In this work, we propose a simple and interpretable automatic termination criterion for BO. The criterion consists of two main ingredients: (i) high-probability confidence bound on the regret (i.e., the difference of our current solution to the global optimum) and (ii) the termination threshold. The first already allows a user to specify a desired tolerance that defines how accurate should the final solution be compared to the global optimum. For the case when cross-validation is used, we recommend a threshold based on the statistical properties of the cross-validation estimator. This threshold takes into account the irreducible discrepancy between the actual HPO objective (i.e., performance on new data) and the target function optimized via BO (i.e., the validation error). Our extensive empirical evaluation on a variety of HPO and neural architecture search (NAS) benchmarks suggests that our method is more robust and effective in maintaining the final solution quality than common baselines. We also surface

overfitting effects in HPO on both small and large datasets, arguably an overlooked problem, and demonstrate that our termination criterion helps to mitigate it. This section is based on our paper "Automatic Termination for Hyperparameter Optimization" [Mak+21a].

4.1.1 Problem Formulation

In this work, we consider the objective $f : \mathcal{X} \to \mathbb{R}$ that we optimize in an iterative manner. At every step t, we select an input $\boldsymbol{x}_t \in \mathcal{X}$ and observe a noisy output

$$y_t \triangleq f(\boldsymbol{x}_t) + \xi_t.$$

where ξ_t is assumed to be i.i.d. (sub)-Gaussian noise with some variance σ_ξ^2. Here, we focus on the GP-based probabilistic model $f \sim GP(\mu, \kappa)$. As observations $y_{1:t} = [y_1, \dots, y_t]^\top$ for the selected inputs $\mathcal{D}_t = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_t\}$ are being collected, they are used to update the posterior belief of the model defined by the posterior mean $\mu_t(\boldsymbol{x})$ and variance $\sigma_t^2(\boldsymbol{x})$ defined in (2.8) and (2.9). The convergence of BO can be quantified via *(simple) regret*:

$$r_t := f(\boldsymbol{x}_t^*) - f(\boldsymbol{x}^*), \tag{4.1}$$

where $\boldsymbol{x}^*$ is the global optimizer of f and $\boldsymbol{x}_t^* = \arg\min_{\boldsymbol{x} \in \mathcal{D}_t} f(\boldsymbol{x})$. Specifying adequate tolerance that defines how small the regret should be to terminate BO is of high importance as it determines both the quality and the cost of the solution. However, this criterion cannot be directly evaluated in practice, as the input $\boldsymbol{x}^*$ and the optimum $f(\boldsymbol{x}^*)$ are not known.

Hyperparameter optimization (HPO) is a classical application for BO. Consider a supervised learning problem that requires training a machine learning model (e.g., a neural network) $\mathcal{M}$ on some feature-response data points $\mathrm{D} = \{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^n$ sampled i.i.d. from some unknown data distribution P. The model is obtained by running a training algorithm (e.g., optimizing the weights of the neural network via SGD) on D, both of which depend on *hyperparameters* $\boldsymbol{x}$ (e.g., learning rates used, batch size, etc.). We use the notation $\mathcal{M}_{\boldsymbol{x}}(\mathrm{x}; \mathrm{D})$ to refer to the prediction that the model produced by $\mathcal{M}$ makes for an input x, when trained with hyperparameters $\boldsymbol{x}$ on data D. Given some loss function $\ell(\cdot, \cdot)$, the *population risk* of the model on unseen data points is given by the expected loss $\mathbb{E}_P[\ell(\mathrm{y}, \mathcal{M}_{\boldsymbol{x}}(\mathrm{x}, \mathrm{D}))]$. The main objective of HPO is to identify hyperparameters $\boldsymbol{x}$, such that the resulting model minimizes the population risk:

$$f(\boldsymbol{x}) = \mathbb{E}_{(\mathrm{x},\mathrm{y})\sim P}\big[\ell\big(\mathrm{y}, \mathcal{M}_{\boldsymbol{x}}(\mathrm{x}, \mathrm{D})\big)\big], \qquad \boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}). \tag{4.2}$$

In practice, however, the population risk cannot be evaluated since P are unknown. Thus, typically, it is estimated on a separate finite validation set D_V drawn from the same distribution P. Practical HPO focuses on minimizing the *empirical estimator* $\hat{f}(\boldsymbol{x})$ of the expected loss $f(\boldsymbol{x})$ leading to the optimizer $\boldsymbol{x}_{\mathcal{D}}^*$:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{|\mathrm{D}_V|} \sum_{(\mathrm{x}_i, \mathrm{y}_i) \in \mathrm{D}_V} \ell\big(\mathrm{y}_i, \mathcal{M}_{\boldsymbol{x}}(\mathrm{x}_i, \mathrm{D})\big), \qquad \boldsymbol{x}_{\mathcal{D}}^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \hat{f}(\boldsymbol{x}). \tag{4.3}$$

At its core, BO-based HPO sequentially evaluates the empirical estimator $\hat{f}(\boldsymbol{x}_t)$ for promising hyperparameters $\boldsymbol{x}_t$ and terminates after some specified number of BO rounds, reporting the solution $\boldsymbol{x}_t^* = \arg\min_{\boldsymbol{x} \in \mathcal{D}_t} \hat{f}(\boldsymbol{x})$, where $\mathcal{D}_t = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$ are the solutions considered so far. We can define the simple regret for the reported solution w.r.t. the validation loss by

$$\hat{r}_t := \hat{f}(\boldsymbol{x}_t^*) - \hat{f}(\boldsymbol{x}_{\mathcal{D}}^*). \tag{4.4}$$

Inconsistency in the optimization objective. Importantly, the true HPO objective $f(\boldsymbol{x})$ in Eq. (4.2) and the empirical surrogate $\hat{f}(\boldsymbol{x})$ in Eq. (4.3) used for tuning by BO generally do not coincide. Therefore, existing BO approaches may yield sub-optimal solutions to the population risk minimization, even if they succeed in globally optimizing $\hat{f}(\boldsymbol{x})$. This issue, however, is typically neglected in practical HPO, as well as a potential overfitting to the validation error. In contrast, we propose a termination condition for BO motivated by the discrepancy in the objectives.

4.1.2 Termination criterion for Hyperparameter Optimization

This subsection firstly motivates why early termination of HPO can be beneficial and then addresses the following two questions: (1) How to estimate the unknown simple regret and (2) What threshold of the simple regret can be used to stop HPO.

Motivation for the termination criterion

We start by analysing the effect of optimizing $\hat{f}$ in lieu of f. We observe that challenges in optimizing f are both due to the statistical error of the empirical BO objective $\hat{f}(\boldsymbol{x})$ and the sub-optimality of the BO candidates, encoded in the simple regret $\hat{r}_t$. The key insight of the following proposition is that iteratively reducing $\hat{r}_t$ to 0 may not bring any benefits if the statistical error dominates.

Proposition 1. *Consider the expected loss f and its estimator $\hat{f}$ defined, respectively, in (4.2) and (4.3), and assume the statistical error of the estimator is bounded as $||\hat{f} - f||_\infty \leq \epsilon_{st}$ for some $\epsilon_{st} \geq 0$. Let $\boldsymbol{x}^*$ and $\boldsymbol{x}_{\mathcal{D}}^*$ be their optimizers: $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \boldsymbol{x}} f(\boldsymbol{x})$ and $\boldsymbol{x}_{\mathcal{D}}^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \hat{f}(\boldsymbol{x})$. Let $\boldsymbol{x}_t^*$ be some candidate solution to $\min_{\boldsymbol{x} \in \mathcal{X}} \hat{f}(\boldsymbol{x})$ with sub-optimality in function value $\hat{r}_t := \hat{f}(\boldsymbol{x}_t^*) - \hat{f}(\boldsymbol{x}_{\mathcal{D}}^*)$. Then the gap in generalization performance $f(\boldsymbol{x}_t^*) - f(\boldsymbol{x}^*)$ can be bounded as follows:*

$$f(\boldsymbol{x}_t^*) - f(\boldsymbol{x}^*) = \underbrace{f(\boldsymbol{x}_t^*) - \hat{f}(\boldsymbol{x}_t^*)}_{\leq \epsilon_{st}} + \underbrace{\hat{f}(\boldsymbol{x}_t^*) - \hat{f}(\boldsymbol{x}_{\mathcal{D}}^*)}_{= \hat{r}_t} + \underbrace{\hat{f}(\boldsymbol{x}_{\mathcal{D}}^*) - \hat{f}(\boldsymbol{x}^*)}_{\leq 0} + \underbrace{\hat{f}(\boldsymbol{x}^*) - f(\boldsymbol{x}^*)}_{\leq \epsilon_{st}} \tag{4.5}$$

$$\leq 2\epsilon_{st} + \hat{r}_t. \tag{4.6}$$

Moreover, without further restrictions on f, $\hat{f}$, $\boldsymbol{x}_t^*$ and $\boldsymbol{x}^*$, the upper bound is tight.

Proof: The equality in Eq. (4.5) is due to adding and subtracting the same values.

The inequality in Eq. (4.6) results from the following bounds:

$$(1)\ f(\boldsymbol{x}_t^*) - \hat{f}(\boldsymbol{x}_t^*) \leq |f(\boldsymbol{x}_t^*) - \hat{f}(\boldsymbol{x}_t^*)| \leq \max_{\boldsymbol{x}\in\mathcal{X}} |f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})| = ||\hat{f} - f||_\infty \leq \epsilon_{st},$$

$$(2)\ \boldsymbol{x}_{\mathcal{D}}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \hat{f}(\boldsymbol{x}) \quad \Rightarrow \quad \forall \boldsymbol{x} \in \mathcal{X} : \hat{f}(\boldsymbol{x}_{\mathcal{D}}^*) - \hat{f}(\boldsymbol{x}) \leq 0 \quad \Rightarrow \quad \hat{f}(\boldsymbol{x}_{\mathcal{D}}^*) - \hat{f}(\boldsymbol{x}^*) \leq 0.$$

■

The proposition bounds the sub-optimality of the target objective f in terms of the statistical error ϵ_{st} and the simple regret $\hat{r}_t$. This naturally suggests terminating HPO at a candidate $\boldsymbol{x}_t^*$ for which the simple regret $\hat{r}_t$ is of the same magnitude as the statistical error ϵ_{st}, since further reduction in $\hat{r}_t$ may not improve notably the true objective. However, neither of the quantities ϵ_{st} and $\hat{r}_t$ are known.

Below, we propose a termination criterion that relies on estimates of both quantities. Firstly, we show how to use confidence bounds on $\hat{f}(\boldsymbol{x})$ to obtain high probability upper bounds on the simple regret $\hat{r}_t$ ([Sri+10; Ha+19]). Secondly, we estimate the statistical error ϵ_{st} in the case of cross-validation ([Sto74; Gei75]) where the model performance is defined as an average over several training-validation runs. To this end, we rely on the statistical characteristics (i.e., variance or bias) of such cross-validation-based estimator that are theoretically studied by [NB03] and [Bay+20].

Building blocks of the termination criterion

Upper bound for the simple regret $\hat{r}_t$. The key idea behind bounding $\hat{r}_t$ is that, as long as the GP-based approximation of $\hat{f}(\cdot)$ is well-calibrated, we can use it to construct high-probability confidence bounds for $\hat{f}(\cdot)$. In particular, [Sri+10] show that, as long as $\hat{f}$ has a bounded norm in the reproducing kernel Hilbert space (RKHS) associated with the covariance function κ used in the GP, $\hat{f}(\boldsymbol{x})$ is bounded (with high probability) by lower and upper confidence bounds $\text{lcb}_t(\boldsymbol{x}) = \mu_t(\boldsymbol{x}) - \sqrt{\beta_t}\sigma_t(\boldsymbol{x})$ and $\text{ucb}_t(\boldsymbol{x}) = \mu_t(\boldsymbol{x}) + \sqrt{\beta_t}\sigma_t(\boldsymbol{x})$. Hereby, β_t is a parameter that ensures validity of the confidence bounds (see Section 4.2.3 for practical discussion and ablation study).

Consequently, we can bound the unknown $\hat{f}(\boldsymbol{x}_t^*)$ and $\hat{f}(\boldsymbol{x}_{\mathcal{D}}^*)$ that define the sub-optimality $\hat{r}_t$:

$$\hat{r}_t = \hat{f}(\boldsymbol{x}_t^*) - \hat{f}(\boldsymbol{x}_{\mathcal{D}}^*) \leq \min_{\boldsymbol{x}\in\mathcal{D}_t} \text{ucb}_t(\boldsymbol{x}) - \min_{\boldsymbol{x}\in\mathcal{X}} \text{lcb}_t(\boldsymbol{x}) =: \bar{r}_t, \tag{4.7}$$

where the inequality for $\hat{f}(\boldsymbol{x}_t^*)$ is due to the definition of the reporting rule $\boldsymbol{x}_t^* = \arg\min_{\boldsymbol{x}\in\mathcal{D}_t} \hat{f}(\boldsymbol{x})$ over the evaluated points $\mathcal{D}_t = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$. We illustrate the idea with an example in Figure 4.2.

Termination threshold. We showed how to control the optimization error via the (computable) regret upper bound $\bar{r}_t$ above. We now explain when to stop BO, i.e., how to choose some threshold ϵ_{BO} and an iteration $T : \bar{r}_T \leq \epsilon_{BO}$. Following Proposition 1, we suggest setting ϵ_{BO} to be of similar magnitude as the statistical error ϵ_{st} of the empirical estimator (since smaller regret $\hat{r}_t$ is not beneficial when ϵ_{st} dominates). In case of cross-validation being used for HPO, one can estimate this statistical error ϵ_{st} and we

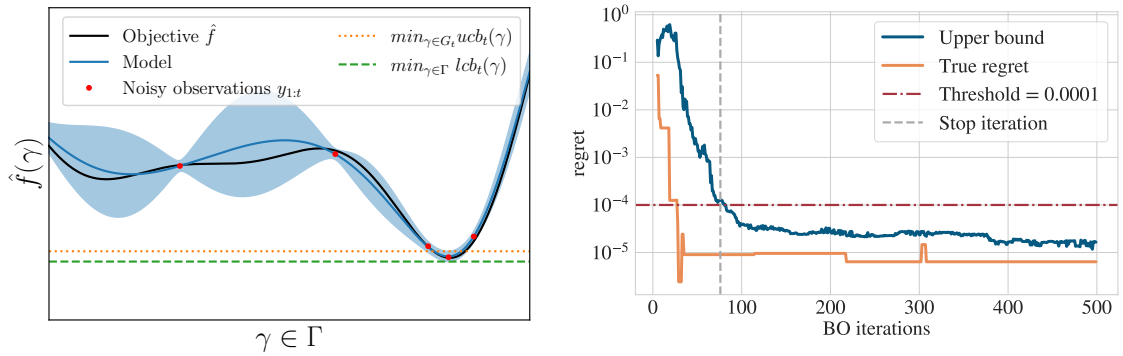


Figure 4.2: Left: Visualization of the upper bound for $\hat{r}_t$. The gap between green and orange lines is the estimate of the upper bound for $\hat{r}_t$. **Right**: Illustration of automated BO termination when tuning MLP on the `naval` dataset from HPO-Bench ([KH19]) with the BORE optimizer ([Tia+21]).

further discuss how it can be done.

Cross-validation is the standard approach to compute an estimator $\hat{f}(\boldsymbol{x})$ of the population risk. The data D is partitioned into k equal-sized sets $\mathrm{D}_1, \ldots, \mathrm{D}_k$ used for (a) training the model $\mathcal{M}_{\boldsymbol{x}}(\cdot; \mathrm{D}_{-i})$, where $\mathrm{D}_{-i} = \cup_{j \neq i} \mathrm{D}_i$ (i.e., training on all but the i-th fold), and (b) validating $\mathcal{M}_{\boldsymbol{x}}(\cdot; \mathrm{D}_i)$ on the i-th fold of the data. These two steps are repeated in a loop k times, and then the average over k validation results is computed.

The statistical error ϵ_{st} of an estimator can be characterized in terms of its variance $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}) = \mathbb{E}[(\hat{f}(\boldsymbol{x}) - \mathbb{E}\hat{f}(\boldsymbol{x}))^2]$ and bias $B(\boldsymbol{x}) = \mathbb{E}[\hat{f}(\boldsymbol{x})] - f(\boldsymbol{x})$, where the latter can be neglected in case of cross-validation ([Bay+20]). Though the variance $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x})$ of the cross-validation estimate is generally unknown, [NB03] propose an unbiased estimate for it. Specifically, for the sample variance (denoted as s^2_{cv}) of k-fold cross-validation, a simple post-correction technique to estimate the variance $\mathrm{Var}\hat{f}(\boldsymbol{x})$ is

$$\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}) \approx \left(\frac{1}{k} + \frac{|\mathrm{D}_i|}{|\mathrm{D}_{-i}|}\right) s^2_{\mathrm{cv}}(\boldsymbol{x}), \tag{4.8}$$

where $|\mathrm{D}_i|, |\mathrm{D}_{-i}|$ are the set sizes. For example, in the case of 10-fold cross-validation we have $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}) \approx 0.21 s^2_{\mathrm{cv}}(\boldsymbol{x})$. We are now ready to propose our termination condition in the following.

Termination condition for BO. *Consider the setup of Proposition 1 where $\hat{f}(\cdot)$ is a cross-validation-based estimator being iteratively minimized by BO. Let $\boldsymbol{x}^*_t$ the solution reported in round t, and $\bar{r}_t$ defined in Eq. (4.7) be the simple regret bound computed at each iteration t. Let the variance $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}^*_t)$ of the estimator $\hat{f}(\cdot)$ be approximated according to Eq. (4.8). Then, BO is terminated once:*

$$\bar{r}_t < \sqrt{\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}^*_t)}. \tag{4.9}$$

Intuitively, the termination is triggered once the maximum plausible improvement becomes less than the standard deviation of the estimate. This variance-based termination condition adapts to different algorithms or datasets and its computation comes with

negligible computational cost on top of cross-validation. The pseudo-code for the criterion is summarised in Algorithm 5. If cross-validation cannot be used or is computationally prohibitive, the user can define the right-hand side of the termination condition. In this case, the upper bound on the left-hand side still has an intuitive interpretation: the user can set the threshold based on their desired solution accuracy. This case is demonstrated in Figure 4.2, with an example of automatic termination for tuning an MLP.

4.1.3 Experiments

The main challenge of any termination criterion for HPO is to balance between reducing runtime and performance degradation. We thus study in experiments how the speed-up gained from different termination criteria affects the final test performance. To this end, we define two new metrics that account for the trade-off between resources saved and performance drop and provide a list of reasonable baselines. The code to reproduce the experimental results is publicly available.[1]

Baselines. Since automatic BO terminating is a rather under-explored topic, we consider the following baselines that are, to the best of our knowledge, the only ones directly related to our method:

- Näive convergence test controlled by a parameter i (referred as Conv-i): stopping BO if the *best* observed validation metric remains unchanged for i consecutive iterations. It is challenging to define a suitable i suitable across different benchmarks since i is a fixed, predetermined choice that does not consider the observed data (in contrast to our method that refines the regret estimation). We consider common in practice values $i = \{10, 30, 50\}$ and study other values in Appendix, see Section D.2.
- Threshold for Expected improvement (EI; [Ngu+17]): stopping BO once EI drops below a pre-defined threshold. Choosing a threshold crucially depends on the problem at hand, e.g., values studied by the original paper result in too aggressive stopping across a range of our experiments. We thus extend it with a finer-grained grid resulting in $\{10^{-9}, 10^{-13}, 10^{-17}\}$.
- Threshold for Probability of improvement (PI, [Lor+16]): stopping BO once PI drops below a pre-defined threshold. Similar to EI baseline, we tune the threshold and use $\{10^{-5}, 10^{-9}, 10^{-13}\}$.

Metrics. We measure the effectiveness of a termination criterion via two metrics quantifying (i) the change in test error on a held-out dataset and (ii) the time saved. Given a T BO iterations budget, we compare the test error y_T after T iterations to the test error y_{es} after early stopping is triggered. For each experiment, we compute the *relative test error change* RYC as

$$\text{RYC} = \frac{y_T - y_{es}}{\max(y_T, y_{es})}. \tag{4.10}$$

[1] https://github.com/avidereta/automatic-termination-in-Bayesian-optimization

This allows us to aggregate the results over different algorithms and datasets, as RYC $\in [-1, 1]$ and can be interpreted as follows: A positive RYC represents an improvement in the test error when applying early stopping, while a negative RYC indicates the opposite. Similarly, let the total training time for a predefined budget T be t_T and the total training time when early stopping is triggered be t_{es}. Then the *relative time change* RTC is defined as

$$\text{RTC} = \frac{t_T - t_{es}}{t_T} \tag{4.11}$$

indicates a reduction in total training time, RTC $\in [0, 1]$. While reducing training time is desirable, it should be noted that this can be achieved through any simple stopping criterion (e.g., consider interrupting HPO with a fixed probability after every iteration). In other words, the RTC is not a meaningful metric when decoupled from the RYC, and, thus, both need to be considered in tandem.

Selecting the data for the bound estimate. Since we are only interested in the upper bound of the simple regret, we conjecture that using only the top-performing hyperparameter evaluations may improve the estimation quality. To validate this, we use results of BORE ([Tia+21]) on the `naval` dataset from HPO-Bench ([KH19]) where we can quantify the true regret (see subsection 4.1.3). We compute the upper bound by Eq. (4.7) using three options: 100%, top 50% or top 20% of the hyperparameters evaluated so far and measure the distance to the true regret (see Figure 4.3). From Figure 4.3, fitting a surrogate model with all the hyperparameter evaluations poses a challenge for estimating the upper bound of the regret, which is aligned with recent findings on more efficient BO with the local probabilistic model, especially for high-dimensional problems ([Eri+19]). Using the top 20% evaluations gives the best upper bound estimation quality in the median, at the cost of the most under-estimations of the true regret (2553). Our method would stop too early due to the under-estimation, thus negatively impacting the RYC score, as shown in Figure 4.3. *As a result, we use the top 50% hyperparameters evaluations for the upper bound estimation throughout this paper.*

BO for hyperparameter tuning with cross-validation

We tune XGBoost (XGB; 9 hyperparameters) and Random Forest (RF; 3 hyperparameters) on 19 small tabular datasets, where we optimize the error rate for classification and root mean square error for regression, computed via 10-fold cross-validation.

Methods setup. We use a Matérn 5/2 kernel for the GP, and its hyperparameters are found based on type II maximum likelihood estimation (see Section D.1 for more details). The termination is triggered only after the first 20 iterations to ensure a robust fit of GP. We use Eq. (4.8) to compute our stopping threshold, i.e., $\mathbb{V}\text{ar}\hat{f}(\gamma) \approx 0.21 s^2_{\text{cv}}(\gamma)$. We additionally use empirical scaling $0.5 s^2_{\text{cv}}(\gamma)$ to study the effect of varying the magnitude.

Results. We present the RYC-RTC results aggregated across the datasets in Figure 4.4, Table 4.3 and Figures 4.4 and 4.3 (Appendix Section D.1). The main takeaway from Figure 4.4 is as follows: more aggressive early stopping might indeed speed up, but it leads to worse test performance in terms of average and standard deviation over the datasets.

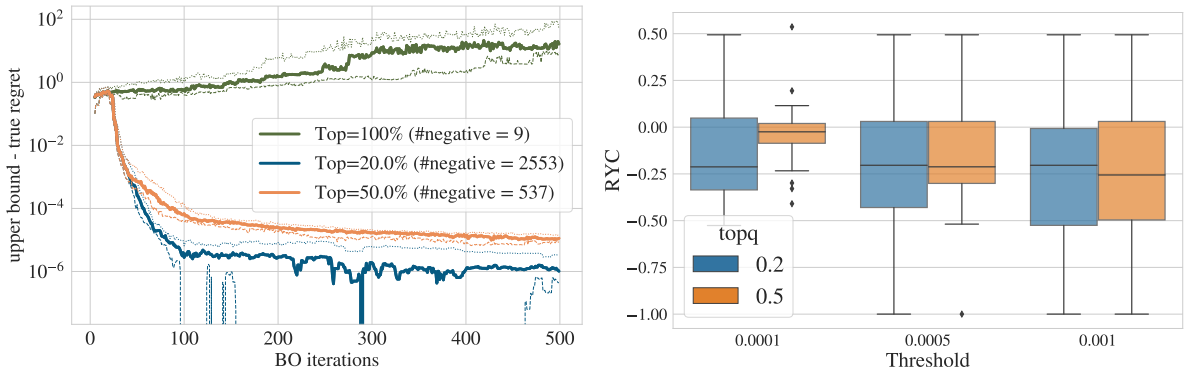


Figure 4.3: The upper bound estimation quality is affected by the set of hyperparameters evaluations used in the surrogate model training. **Left**: Bound quality for using all, top 50% and top 20% hyperparameter evaluations, measured by the difference between upper bound and true regret. The solid line represents the median over 50 replicates, the dashed is 20'th quantiles, and the dotted line is 80'th quantiles. The legend also shows the number of negative differences (the upper bound is smaller than the true regret). **Right**: Box plots of RYC scores when using the top 50% and top 20% hyperparameter evaluations under common thresholds.

In contrast, the desired behavior is the trade-off between RYC and RTC, where lower RYC error bars are prioritized over lower RTC error bars. In other words, the methods that adaptively stop BO, i.e., stops when necessary and does not stop when not, are preferable. Figure 4.4 indicates that our method successfully maintains a high solution quality across a wide range of scenarios (lower RYC variance) via adapting the termination to the particular problem (higher variance RTC). The results also show an anticipated RYC-RTC trade-off for i in Conv-i and thresholds in EI and PI baselines, where the solution quality improves as the thresholds increase and, consequently, the speed-up drops. The average RYC results in Table 4.3, however, are dominated by our method.

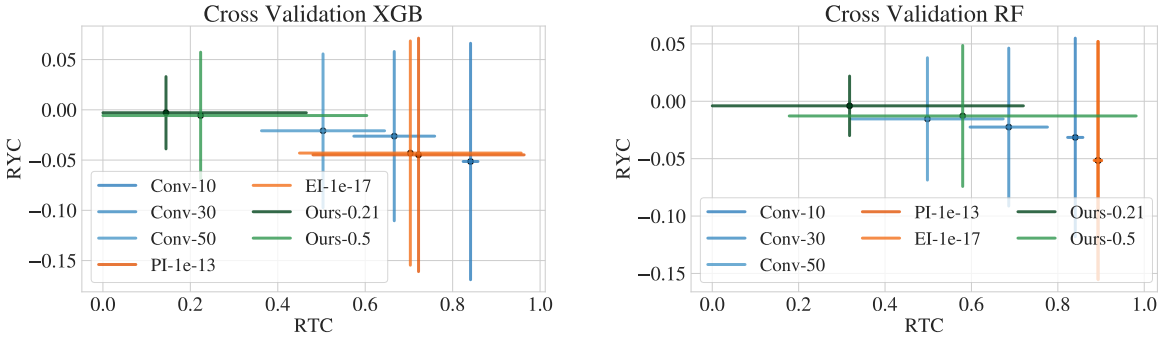


Figure 4.4: The mean and standard deviation of RYC and RTC scores for the compared automatic termination methods when using cross-validation in the hyperparameter evaluation when tuning XGB (left) and RF (right). The mean value is shown as the large dot, and the standard deviation is shown as an error bar in both dimensions.

In these experiments, we study out termination criterion beyond the BO scope, showing its main advantage of being applicable to any iterative HPO method. In addition, here we also show how to use our method if cross-validation is unavailable. To demonstrate this, we apply it to several state-of-the-art methods: TPE ([Ber+11a]), BORE ([Tia+21]), GP-BO ([SLA12b]) as well as random search (RS) ([BB12]).

Benchmarks. We consider two popular tabular benchmark suites: NAS-HPO-Bench ([KH19]), which mimics the hyperparameter and neural architecture search of multi-layer perceptrons on tabular regression datasets, and NAS-Bench-201 ([DY20]) for neural architecture search on image classification datasets. Notice that for NAS-Bench-201, we used *validation* metrics to compute RYC instead of test metrics; thus, no positive RYC scores are observed. We refer to the original paper for a detailed description of these benchmarks.

Methods setup. We consider the following thresholds on the final regret $\{0.0001, 0.001, 0.01\}$, corresponding to a loss of performance of 0.01%, 0.1% and 1%, respectively. Note that it is easier to set the threshold for our method because it is a threshold on the regret in the metric space that users aim to optimize, and it gives more explicit control of this trade-off, thus making it more interpretable. We perform 50 independent runs with a different seed for each method and dataset.

Results. Figure 4.5 and Figure 4.6 show results for BORE, and the rest can be found in the Appendix, Chapter D. While *no method* Pareto dominates the others, our termination criterion shows a similar trend as in Section 4.1.3 by prioritizing accuracy over speed. Users choose the threshold based on their preference regarding the speed-accuracy trade-off, i.e., a higher threshold saves more wall-clock time but potentially leads to a higher drop in performance.We further show the distribution of true regrets at the stopping iteration triggered by our method with the considered thresholds on HPO-Bench in Figure 4.6.

From Figure 4.6, with a high threshold of 0.01, all the experiments (4 datasets with 50 replicates) are early stopped by our method, and 41 (20%) experiments end up with true regret being higher than the threshold. With a low threshold of 0.0001, 112 experiments are stopped, and 12 (10.7%) experiments end up with true regret above the threshold. In short, our method achieves 80% to 90% success rate where the true regret is within the user-defined tolerance.

For every method, we aggregated the scores over datasets with other HPO optimizers in Figure D.1 (see Section D.2). We can see that the speed up of the convergence check baseline is affected very mildly by the optimizers while the RYC scores largely depend on the optimizer: RYC scores with the random search are worse than with BORE. In contrast, the RYC scores for our termination criterion are similar across optimizers, especially for smaller thresholds. On the other hand, the speed up for a given threshold tends to vary. This can be explained by the difference in the optimizer's performance for example random search is not as efficient as BORE, and hence the regret is mostly above the stopping threshold. In summary, while convergence check baselines are by design robust in terms of time saved, our method is more robust in maintaining the solution quality.

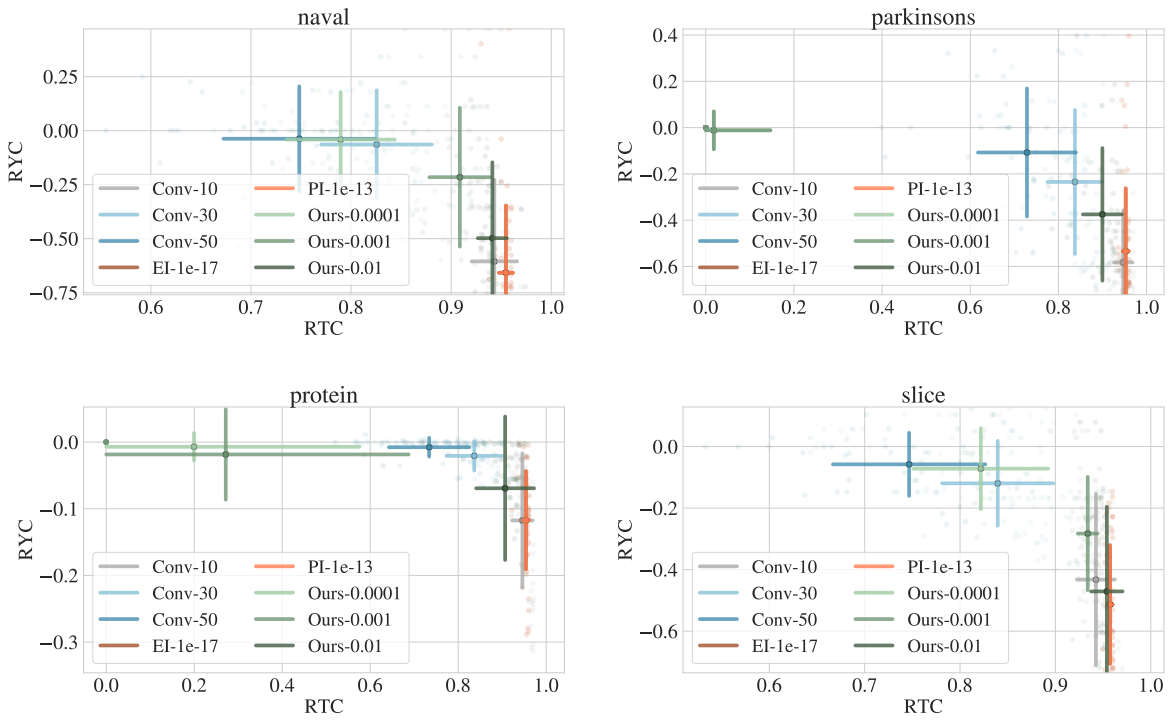


Figure 4.5: Mean (large dot) and standard deviation (error bar) of RYC and RTC scores for all methods for HPO-Bench datasets.

Overfitting in BO for Hyperparameter Optimization

Proposition 1 emphasizes an important problem of BO-based HPO: while focusing (and minimizing) the validation error, we cannot fully reduce the discrepancy between the validation and test errors. Empirically, we show this might happen when the correlation between the test and validation errors is low, thus, improvement in validation performance does not lead to better test results. A particular example of such low correlation in the *small* error region is presented in Figure 4.5 (Section D.2) when tuning XGB and Random Forest on tst-census dataset.

A positive RYC score indicates overfitting in our experiments, showing that the test error at the terminated iteration is lower than in the final round. We observe positive RYC scores in both Figure 4.4 and Figure 4.5, one with cross-validation on small datasets and one with medium-sized datasets. Hence, we would like to raise attention to the possible overfitting issue that occurs in HPO, for which our method can be used as a plugin to mitigate overfitting.

4.1.4 Conclusion

Despite the usefulness of hyperparameter optimizations (HPO), setting a budget in advance remains a challenging problem. In this work, we propose an automatic termination criterion that can be plugged into many common HPO methods. The criterion uses an intuitive and interpretable upper bound of simple regret, allowing users explicitly

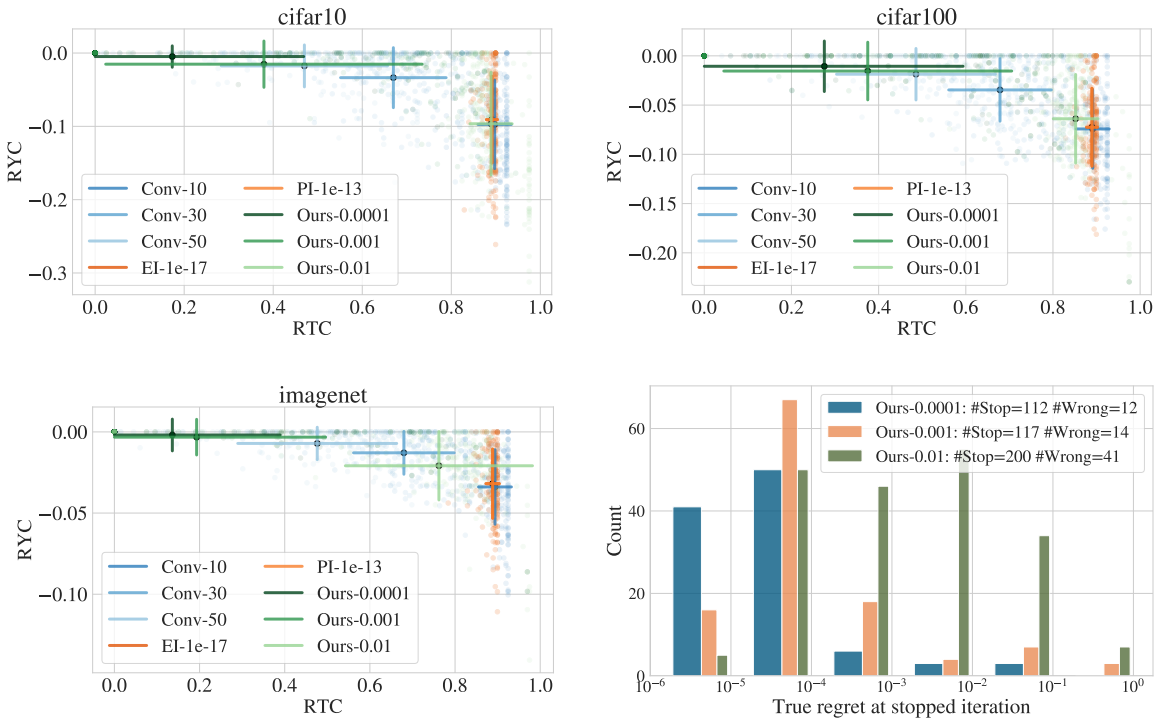


Figure 4.6: First three figures from the left show the mean and standard deviation of RYC and RTC scores for all methods on NAS-Bench-201. Since *validation metrics* are used in these experiments, no positive RYC scores are observed. The mean value is shown in the big dot, and the standard deviation is shown as an error bar in both dimensions. The Figure on the right shows a distribution of true regrets at the stopping iteration triggered by our method with different thresholds for HPO-Bench.

control the accuracy loss. In addition, when cross-validation is used in the evaluations of hyperparameters, we propose to use an analytical threshold rooted in the variance of cross-validation results. The experimental results suggest that our method can be robustly used across many HPO optimizers. Depending on the user-defined thresholds, with 80% to 90% chance, our method achieves true regret within that threshold, saving unnecessary computation and reducing energy consumption. We also observe that overfitting exists in HPO even when cross-validation is used. We hope our work will draw the attention of the HPO community to the practical questions of how to set a budget in advance and mitigate overfitting when tuning hyperparameters in machine learning.

4.1.5 Discussion

The termination criterion we introduce in this work demonstrates promising results in practice, and we explore principled ways to enhance it. Specifically, (i) we first briefly discuss its potential beyond HPO, in more general BO applications, and (ii) we examine ways to improve the statistical characteristics and computational efficiency of the threshold in the termination criterion.

(i) Applicability of the termination criterion beyond the HPO problem

The automatic termination concept based on Eq. (4.7) can also be applied beyond

HPO, in cases where BO point evaluations are impaired not only by random noise but also by some *adversarial corruptions* ([BKS20]). In the case of HPO, such an adversarial corruption is characterized by the discrepancy between the true objective and the computable target that leads to the Eq. (4.6). Similarly, in a general case, such an adversarially-corrupted setup relies on the objective f, its estimator $\hat{f}$, and the bounded statistical error of the estimator $\|f - \hat{f}\| \leq \epsilon_{st}$ for some $\epsilon_{st} \geq 0$. The only HPO-specific part is the well-studied estimation of ϵ_{st} for cross-validation, and the estimation of ϵ_{st} for a particular BO application would make the proposed termination condition applicable beyond HPO.

(ii) Statistical characteristics and the cost of the threshold

Why termination does not guarantee to converge to an optimal solution. This work, and particularly [Mak+21a], addresses the challenge of estimating the statistical error ϵ_{st} by using the variance (a component of the statistical error) in Eq. (4.8) and further using this variance at the *best-observed point* $\boldsymbol{x}_t^*$ for the termination. The latter decision to choose the variance values at $\boldsymbol{x}_t^*$ is empirically motivated and demonstrates promising practical results. That, however, relies on the two noisy values: (i) The variance $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}_t^*)$ is computed from a single cross-validation-based sample variance $s^2_{\mathrm{cv}}(\boldsymbol{x}_t^*)$ that is noisy; (ii) The best-observed point $\boldsymbol{x}_t^* = \underset{\boldsymbol{x}_i \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}}{\arg\min}\ y_i$ is chosen based on the realization of the noisy evaluation y_i. As a result, once the termination criterion is triggered, it does not guarantee to converge to the optimal solution. For example, consider an iteration t and assume that the observed variance of interest $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}_t^*)$ is large (due to noise) and consequently triggered the termination criterion, i.e., $\bar{r}_t < \sqrt{\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}_t^*)}$. However, at the next iteration, it might happen that $\bar{r}_{t+1} > \sqrt{\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}_{t+1}^*)}$, leading to the precipitate termination.

Ways to get a better variance estimation. One way to obtain a more reliable estimation of the variance $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}_t^*)$ is a *repeated experiment setup*, that is collecting several noisy evaluations $\{y_i(\boldsymbol{x})\}_{i=1}^N$ with $y_i(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) + \varepsilon_i$ by training and testing an ML model N times, i.e., N *cross-validation rounds*. In practice, it might be too expensive, and instead, one can use the sample variance evaluations collected during BO to get a better estimation under the assumptions homoscedasticity or heteroscedasticity of noise:

Idea 1: Under *homoscedastic* variance assumption, when all observations y_t come from a distribution with the same (unknown) variance ρ^2, sample variance evaluations $s^2_{\mathrm{cv}}(\boldsymbol{x}_1), \cdots, s^2_{\mathrm{cv}}(\boldsymbol{x}_t)$ can be used to estimate ρ^2. Confidence intervals can then be constructed for ρ^2, and the post-correction technique in Eq. (4.8) can be applied to obtain a threshold variance, as discussed in Section 5.1.

Idea 2: Under *heteroscedastic* variance assumption, when the variance of the observations y_t depends on the input $\boldsymbol{x}$, the unknown function $\rho^2(\boldsymbol{x})$ represents this dependency. Empirical studies support this assumption for tuning ML models such as [Cow+21]. In the context of risk-averse BO, some hyperparameters result in noisier output than others, as discussed in Chapter 3. Similarly to the homoscedastic case, sample variance evaluations $\{s^2_{\mathrm{cv}}(\boldsymbol{x}_1), \cdots, s^2_{\mathrm{cv}}(\boldsymbol{x}_t)\}$ can be used to estimate $\rho^2(\boldsymbol{x})$ and assuming $\rho^2(\boldsymbol{x})$ belongs to some RKHS, GP-based confidence intervals can be constructed and the post-correction

technique in Eq. (4.8) can be applied to obtain the threshold variance. This approach is an avenue for future research.

Adaptation of the BO objective. The proposed automatic termination criterion in the BO-based automated machine learning (AutoML) is a workaround for the inherent discrepancy between the two objectives: the true generalization performance of a model and its empirical estimator. By terminating optimization when further progress is inefficient in terms of computational resources, we acknowledge the fundamental gap between these two objectives. BO setups, where an adversary can additively perturb the observed rewards and corruption-robust strategies, were considered in [BK21; BKS20]. As an alternative to optimizing the empirical estimator, we can optimize the *generalization performance directly* by properly constructing confidence bounds for it. To this end, we can use Proposition 1 and the variance model (Idea 2) to build a conservative estimate of the generalization error and use it to guide the Bayesian optimization process instead of the validation error. This approach, which involves modeling both the validation error function $\hat{f}(\boldsymbol{x})$ and the validation variance $\rho(\boldsymbol{x})$, has been used to choose a risk-averse solution in Bayesian optimization [Mak+21b] and Chapter 3. In Section 5.1, we study improving the reliability of variance estimation.

Budget-adaptive variance estimation. In the case of cross-validation being unaffordable during the whole run of BO, one can make the variance estimate budget-adaptive by determining the number of cross-validation folds on the fly of BO. That is in the spirit of *multi-fidelity* Bayesian optimization studied in more detail in Section 4.2.2. At each BO iteration, we can choose the fidelity, e.g., ten folds with higher accuracy or cheaper three folds with lower accuracy, that might provide a cost-accuracy balance. Note that this approach would also require the homoscedasticity or heteroscedasticity assumptions on the noise variance discussed above.

By this, we conclude the first part of this chapter and follow with the second part focusing on the multi-fidelity approach.

4.2 Multi-fidelity Bayesian optimization

In the standard setting of Bayesian optimization, the objective function is assumed to be expensive to evaluate. This motivates for incorporation of cheaper albeit less accurate objective approximations into the optimization process, so-called multi-fidelity optimization. Such approximations are available for a wide range of BO applications where one can control the scale of the evaluation experiment. For example, trading off simulations and physical experiments in RL [Mar+17], controlling the number of iterations in ML models training in hyperparameter optimization (considered in Section 4.1), or trading off the number of repetitive evaluations in drug design or in other risk-averse setup studied in Chapter 3. In this section, we consider a particular application example for multi-fidelity optimization – calibrating transport system simulators. On the one hand, it involves a complex, iterative, and stochastic nature of the multi-agent simulation process,

on the other hand, allows for multiple fidelities via adjusting the number of agents or iterations. The methodological contribution of this section is thus driven and examined on this specific domain that we introduce along with our approach.

Application example Transport system simulators have become popular tools for studying large urban transport networks and analyzing the complex interactions between travelers ([Chi+11]). These interactions are modeled as a competition among agents for limited network resources to satisfy travel needs. Accurate representation of demand (e.g., route choice behavior), supply (e.g., link capacities), and their interaction is crucial for these models to function effectively. This representation involves a large number of parameters that need to be carefully calibrated to represent real-world traffic scenarios in a meaningful way. Joint calibration techniques are becoming increasingly important for accurately representing real-world traffic scenarios due to their ability to capture the correlation between demand and supply parameters ([ABK07; BBK07]). Calibrating a transport system simulator involves minimizing the mismatch between the simulator's output and real-world measurements. This optimization problem is challenging due to the high computational cost of running a transport system simulation, and the complex, iterative, and stochastic nature of the simulation process. Therefore, we cannot derive the gradient or Hessian of the objective function analytically with respect to the simulator's parameters.

Bayesian optimization (BO) is a suitable method for this task because of its ability to handle black-box functions, global search capabilities, and sample efficiency (see Figure 4.7). However, standard BO is inherently sequential, meaning it must wait for the current evaluation to be completed before choosing a new set of parameters to query. This can be wasteful with modern parallel computing infrastructures. Batch Bayesian Optimization (BBO) ([Gon+16; DKB14]) addresses this issue by allowing for parallel evaluations. Moreover, standard BO assumes that we can only query directly the target function. However, for example, in transport system simulations we have easy access to cheaper albeit less accurate function evaluation, e.g., for a low number of iterations or with a reduced population. Multi-Fidelity Bayesian Optimization (MFBO) ([Mar+17; Kan+17]) studies how to incorporate these different sources of information with different costs and different accuracies to quickly rule out search parameter configurations that are not promising.

Our Contributions. In this work, we present Multi-Fidelity Batch Min-value Entropy Search (MF-BMES), a novel algorithm that brings together ideas from batch Bayesian optimization and multi-fidelity Bayesian optimization for the efficient joint calibration of transport system simulations. Similarly to [DKB14], it encourages diversity within a batch of parameters to query in parallel by adding temporary observations to the data set for the parameters already chosen for the current batch. Inspired by [Mar+17], it learns the mismatch in accuracy between different fidelity levels and chooses parameter configurations to query based on information gained per unit of cost criterion. Finally, in order to make the computation more efficient and robust in this multi-fidelity context, it substitutes the entropy search criterion of [HS12] used by [Mar+17] with the max-value entropy search of [WJ17]. We show the effectiveness of our method on both small-scale and larger-scale scenarios used

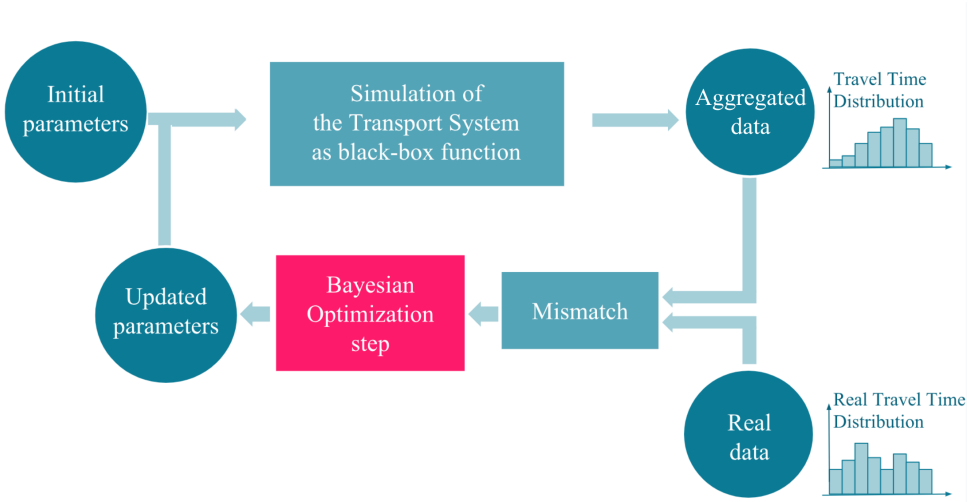


Figure 4.7: This figure illustrates the iterative calibration procedure for a transport system simulation using Bayesian optimization.

in transport planning. Importantly, MF-BMES is the first Bayesian optimization method that has been applied to traffic simulator calibration, and its complementarity to previous work in the field motivates it for being widely applied to this class of problems. For example, a practitioner can set a great number of parameters using gradient-based calibration methods and fine-tune the critical ones with BO. We release the code[2] of our algorithm that provides a flexible interface for calibrating different transport system simulations.

The section is organized as follows. We define the calibration as an optimization problem in Section 4.2.1. Section 4.2.2 provides background for batch BO and multi-fidelity BO that are key for our method. In Section 4.2.3 we present our MF-BMES algorithm. The main application of our method is the calibration of transport systems simulators: we discuss the relevant methods in Section 4.2.4, and in Section 4.2.5 we empirically compare MF-MES with them.

4.2.1 Problem Formulation

In this section, we formally introduce the calibration problem for transport system simulators. Moreover, we analyze the characteristics of the problem that make many of the commonly used optimization methods unsuitable for finding a solution.

Let $\mathbf{r} \in \mathcal{R}$ denote observations of aggregate traffic measures for a traffic network. For example, daily traffic counts at some measurement stations, hourly travel time distributions across the population, daily travel transport mode shares, and others. A simulator g maps input parameters of demand and supply to a list of atomic traffic events representing agent activities during a simulated period. Let h map a list of atomic traffic events to an aggregate traffic statistic. Finally, the mapping from input parameters to the aggregate traffic measure is $q = h \circ g : \mathcal{X} \to \mathcal{R}$. We aim to minimize the distance between the outcome of the simulation and the available observations, i.e., given a distance metric

[2] https://gitlab.ethz.ch/ivt-vpl/oc-matsim

$d_{\mathcal{R}} : \mathcal{R} \times \mathcal{R} \to \mathbb{R}^+$ we want to fins a minimizer of the expected distance value:

$$f(\boldsymbol{x}) \triangleq \mathbb{E}_g d_{\mathcal{R}}(\mathbf{r}, q(\boldsymbol{x})) \qquad \boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}). \tag{4.12}$$

This optimization problem is difficult to solve because the mapping g results from a complex iterative procedure involving expensive network loading algorithms as subroutines and does not have an analytical form. Moreover, the evaluation is computationally demanding as it usually requires multiple simulations of whole-day activities for every agent in the network until some notion of network equilibrium is achieved. This prevents us from using standard first or second-order gradient-based methods and optimization methods that require many function evaluations. Therefore, data-demanding optimization methods such as random or grid search, direct search methods ([KLT03]), simulated annealing ([KGV83]), genetic algorithms ([Whi94]) or algorithms aiming at estimating gradients, such as SPSA ([Lu+15]), might not be well-suited for such problems. Calibrating transport system simulators is a possibly non-convex optimization problem with unavailable derivative information and expensive function evaluations. That motivates for application of Bayesian optimization.

4.2.2 Background

To address the problem, we again rely on the Bayesian inference (see Section 2.1) and use Gaussian Processes ($\mathcal{GP}$s, see Section 2.2.1) as a flexible non-parametric model to describe probability distributions in function spaces. In contrast to previous chapters of this dissertation, here we actively rely on the parallel evaluation of the objective. Moreover, we incorporate cheaper but less accurate evaluations of the objective into the optimization process. To this end, we use *batch Bayesian optimization* and *multi-fidelity Bayesian optimization*, which we formally introduce below.

We assume $f \sim \mathcal{GP}(0, \kappa)$, and given the data $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^t$, where $y_i = f(\boldsymbol{x}_i) + \xi_i$, with noise $\xi_i \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma_\xi^2)$, the posterior distribution over the function value at $\boldsymbol{x} \in \mathcal{X}$ is a Gaussian distribution with posterior mean $\mu_t(\boldsymbol{x})$, variance $\sigma_t^2(\boldsymbol{x})$ and covariance $\kappa_t(\boldsymbol{x}, \boldsymbol{x}')$ defined in (2.8) and (2.9). The common acquisition functions are discussed in Section 2.5.2 and include approaches such as improvement-based (PI and EI), optimism-based (GP-UCB), and information-theoretic (entropy search, predictive entropy search, and max-value entropy search). The works [Mar+17; Kle+16; MOR17] use predictive entropy search and argue that information-based acquisition functions are suitable for taking into account the cost of evaluations because the information gained per unit of cost is an intuitive quantity to optimize. Max-value entropy search (MES) introduced later in ([WJ17]) is easier to compute than other information-based methods and leads to equal or better optimization results. In this work, we focus on MES, which in case of minimization problem Eq. (4.12) can be computed as follows:

$$\alpha^{MES}(\boldsymbol{x}) \approx \frac{1}{M} \sum_{y^*} \left[\frac{\gamma_{y^*}(\boldsymbol{x})\psi(\gamma_{y^*}(\boldsymbol{x}))}{2\big(1 - \Psi(\gamma_{y^*}(\boldsymbol{x}))\big)} + \log(1 - \Psi(\gamma_{y^*}(\boldsymbol{x}))) \right], \tag{4.13}$$

where ψ and Ψ indicate the probability density function and the cumulative density

function of the standard Gaussian distribution respectively, and $\gamma_{y^*}(\boldsymbol{x}) = \frac{y^* - \mu_t(\boldsymbol{x})}{\sigma_t(\boldsymbol{x})}$.

Batch Bayesian optimization

The Bayesian optimization algorithm, referred to as Algorithm 1, is inherently a sequential process that relies on the evaluation of the objective function at each iteration to determine the next step. This is because the new evaluation is necessary to update the posterior belief and determine the optimizer of the acquisition function. When the evaluation of the objective function is expensive, the sequential process can become a bottleneck. Batch Bayesian optimization aims to address this issue by identifying batches of points to query in parallel, with the goal of reducing the impact of the evaluation time. However, this requires modifying the acquisition function to take into account inputs that are already part of the current batch but for which function evaluations are not yet available. Figure 4.8 illustrates the differences between sequential and batch BO. In batch BO, the goal at each time step t is to select a batch of B points $\{\boldsymbol{x}_{t,k}\}_{k=1}^{B}$ based on the data $\mathcal{D}_{t-1}$ observed prior to the time step t, i.e.,

$$\mathcal{D}_{t-1} = \{(\boldsymbol{x}_{1,1}, y_{1,1}), \ldots\ldots (\boldsymbol{x}_{t-1,B}, y_{t-1,B})\}.$$

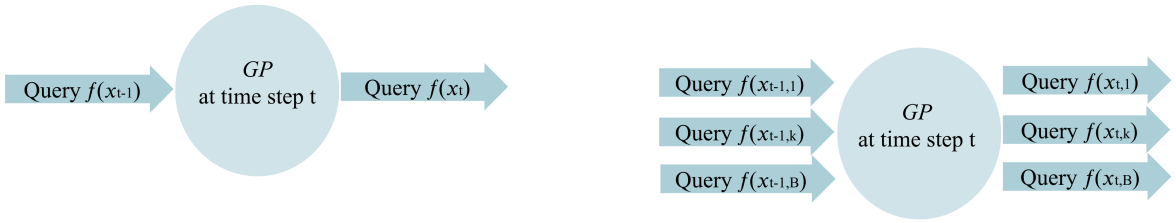


Figure 4.8: Scheme of sequential BO (left) and Batch BO (right)

Algorithm 3 Batch BO by [DKB14]

Input: Domain $\mathcal{X}$, Kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, Acquisition function $\alpha : \mathcal{X} \to \mathbb{R}$, Batch size B
Initialize data $\mathcal{D}_0 \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  $\tilde{\mathcal{D}}_{t,0} \leftarrow \mathcal{D}_t$
  for $k = 1, 2 \ldots, B$ **do**
    $\boldsymbol{x}_{t,k} \leftarrow \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}}\ \alpha\big(\boldsymbol{x}; p(f|\tilde{\mathcal{D}}_{t,k-1})\big)$
    $\tilde{\mathcal{D}}_{t,k} \leftarrow \tilde{\mathcal{D}}_{t,k-1} \cup \{(\boldsymbol{x}_{t,k}, \mu_{t-1}(\boldsymbol{x}_{t,k}))\}$
  end for
  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \bigcup_{k=1}^{B} \{(\boldsymbol{x}_{t,k}, f(\boldsymbol{x}_{t,k}) + \xi_{t,k})\}$
end for

Our goal is to select the B points that constitute the t^{th} batch. While it is straightforward to choose the first point, $\boldsymbol{x}_{t,1}$, by maximizing any of the acquisition functions introduced in Section 4.1.1, the same is not possible for the remaining $B-1$ points. Particularly, failing to update our posterior belief about the objective with new observations will result

in the maximizer of the acquisition function remaining constant for the entire batch. Many solutions have been proposed to address this issue, such as [AFF10; Gon+16; DKB14]. In this work, we focus on the approach suggested by [DKB14] as it has fewer hyperparameters and is more intuitive. This method relies on the fact that the posterior covariance over the objective only depends on the input $\boldsymbol{x}$ and not on the value $f(\boldsymbol{x})$, as can be seen in Eq. (2.9). Therefore, once a point has been chosen for the current batch, we can update the posterior covariance and leave the mean unchanged. When we maximize the acquisition function based on this updated covariance, the maximizer will be different, ensuring diversity within the batch. Obtaining an observation equal to $\mu_t(\boldsymbol{x})$ at input $\boldsymbol{x}$ updates the posterior belief, leaving the mean unchanged and modifying the covariance according to Eq. (2.10). To encourage diversity within a batch, [DKB14] propose temporarily inserting a fictitious observation with a value equal to the posterior mean into the data set. Within a batch, the k^{th} input is chosen by maximizing the acquisition function based on a fictitious data set, $\tilde{\mathcal{D}}_{t,k-1}$:

$$\tilde{\mathcal{D}}_{t,k-1} = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_{t,1}, \mu_{t-1}(\boldsymbol{x}_{t,1})), \cdots (\boldsymbol{x}_{t,k-1}, \mu_{t-1}(\boldsymbol{x}_{t,k-1}))\}. \tag{4.14}$$

While this fictitious data set is created temporarily to encourage diversity within a batch, we update the true data set with the values that we actually obtain from function evaluations once they become available, and we drop the fictitious observations. The algorithm using a generic acquisition function is summarized in Algorithm 3. In the work of [DKB14] the acquisition function is GP-UCB and, thus, their algorithm is called Gaussian Process Batch Bayesian optimization with UCB (GP-BUCB). However, the principle they introduce to find batches of points to query in parallel can be applied to other acquisition functions as well (albeit without the same theoretical guarantees).

Multi-fidelity Bayesian optimization

In this section, we show how the standard BO framework can be adapted using multiple cheaper but noisier approximations of the objective function. The goal is to use the cheaper approximations to quickly eliminate sub-optimal regions of the search space, and gradually increase the accuracy of the approximations as the search progresses to identify a high-quality solution. More formally, we assume we have access to approximations $f^{(0)}(\boldsymbol{x}), \ldots, f^{(L-1)}(\boldsymbol{x})$ of the objective $f(\boldsymbol{x})$. These approximations are ordered by their accuracy or, in other words, *fidelity*, with $f^{(0)}(\boldsymbol{x})$ being the least accurate and with $f^{(L-1)}(\boldsymbol{x})$ being the most accurate. We denote the objective $f(\boldsymbol{x})$ with $f^{(L)}(\boldsymbol{x}) \triangleq f(\boldsymbol{x})$ and refer to it as the *full fidelity*. Each approximation $f^{(l)}(\boldsymbol{x})$ has its evaluation cost $c^{(l)}(\boldsymbol{x})$, e.g., evaluation time or storage space, that is assumed to be known and ordered according to $c^{(0)}(\boldsymbol{x}) \leq c^{(1)}(\boldsymbol{x}) \cdots \leq c^{L}(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. The evaluations are perturbed by some noise $y^{(l)}(\boldsymbol{x}) = f^{(l)}(\boldsymbol{x}) + \xi^{(l)}$ with $\xi^{(l)} \overset{i.i.d}{\sim} \mathcal{N}(0, \sigma_\xi^2)$. We aim to *trade off the cost and information gain* of the evaluation to do fewer evaluations on the expensive objective when it is not crucial. While in the standard BO, we only select the input $\boldsymbol{x} \in \mathcal{X}$ for the next evaluation, in multi-fidelity BO, we also choose the function $f^{(l)}(\boldsymbol{x})$, $l \in [m]$. For a graphical intuition of the multi-fidelity setting see Figure 4.9.

Multi-fidelity extension of BO has been widely studied. For example, extensions to

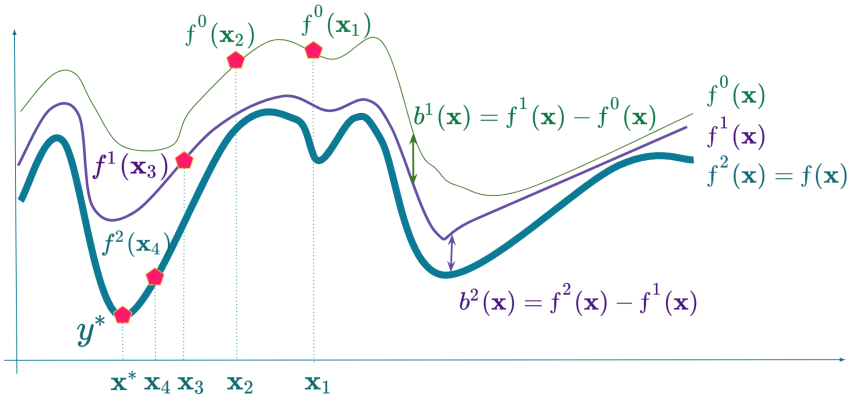


Figure 4.9: Schematic illustration of the objective $f(\boldsymbol{x})$, its approximations $f^{(0)}(\boldsymbol{x}), f^1(\boldsymbol{x})$ and the multi-fidelity optimization procedure. Cheaper evaluations the funcitons $f^{(0)}(\boldsymbol{x})$ and $f^(1)(\boldsymbol{x})$ at inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ guide the convergence to the optimum $y^* = \min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. At each step, we pick both the input $\boldsymbol{x} \in \mathcal{X}$ and fidelity $l = [0, 1, 2]$ to evaluate at $\boldsymbol{x}$. To balance the cost and the information gain, we model the functions $b^{(l)}(\boldsymbol{x})$ representing the bias between fidelity levels.

the standard EI were proposed in [Pic+13]. The EI acquisition function, as discussed in Section 5.2.3 is rather myopic, and other solutions to use the cheaper approximations in GP-UCB ([Sri+10]) have been proposed in [Kan+16; Kan+17]. Information-based acquisition function has been also studied in the multi-fidelity setting, such as ES-based [Kle+20] and PES-based [MOR17]. The advantage of these methods is that they can directly measure the utility of the evaluation for the global optimizer, but they are computationally expensive which makes them impractical in high-dimensional spaces. A more practical method that transforms the tedious computation into a one-dimensional problem is max-value entropy search (MES, see Eq. (2.25)). Therefore, in this work, we propose an MES-based multi-fidelity approach. Independently to this work, another multi-fidelity MES was proposed in [Tak+20].

The correlation between different approximations adds an extra decision variable to incorporate into the statistical model and the acquisition function, which we implement as follows. Assume the lowest-fidelity function is a $\mathcal{GP}$ sample, i.e. $f^{(0)}(\boldsymbol{x}) \sim \mathcal{GP}\big(\mu^{(0)}(\boldsymbol{x}), \kappa^{(0)}(\boldsymbol{x}, \boldsymbol{x}')\big)$. We consider the difference between consecutive fidelity levels, denoted as $b^{(l)}(\boldsymbol{x}) \triangleq f^{(l)}(\boldsymbol{x}) - f^{(l-1)}(\boldsymbol{x})$, $l \in [L]_+$, and assume them being $\mathcal{GP}$ samples too, i.e., $b^{(l)}(\boldsymbol{x}) \sim \mathcal{GP}\big(\mu^{(l)}(\boldsymbol{x}), \kappa^{(l)}(\boldsymbol{x}, \boldsymbol{x}')\big)$. It implies that every function $f^{(l)}$, $l \in [L]$, is also a GP sample. Moreover, the function $\tilde{f} : \mathcal{X} \times \{0, \cdots, L\} \to \mathbb{R}$ such that $\tilde{f}(\boldsymbol{x}, l) = f^{(l)}(\boldsymbol{x})$, i.e., extended to the domain of the fidelity level, also follows $\tilde{f} \sim \mathcal{GP}(\tilde{\mu}, \tilde{\kappa})$ that we define below. Let $\tilde{\mu}(\boldsymbol{x}, 0) \triangleq \mu^{(0)}(\boldsymbol{x})$ and then $\forall l \in [L]_+$ the mean is defined as:

$$\tilde{\mu}(\boldsymbol{x}, l) = \mu^{(0)}(\boldsymbol{x}) + \sum_{i=1}^{l} \mu^{(i)}(\boldsymbol{x}). \tag{4.15}$$

The kernel should capture both how the function values co-vary across the domain $\mathcal{X}$ and

fidelities. Let $\boldsymbol{\delta}^{(l)} \in [0,1]^{L+1}$ be a binary vector with i^{th} component defined as:

$$\delta_i^{(l)} = \begin{cases} 1 & \text{if } i \leq l \\ 0 & \text{otherwise.} \end{cases} \tag{4.16}$$

Then we define the kernel $\tilde{\kappa}$ that models the objective or higher fidelity functions as being partly explained via the lower fidelity approximations plus some bias terms as follows:

$$\tilde{\kappa}\big((\boldsymbol{x}, l), (\boldsymbol{x}, l')\big) = \kappa^{(0)}(\boldsymbol{x}, \boldsymbol{x}') + \sum_{i \in [L]_+} \kappa_\delta(\delta_i^{(l)}, \delta_i^{(l')}) \cdot \kappa^{(l)}(\boldsymbol{x}, \boldsymbol{x}'), \tag{4.17}$$

where the kernel $\kappa_\delta(\delta_i^{(l)}, \delta_i^{(l')}) = \delta_i^{(l)} \delta_i^{(l')}$. Intuitively, the kernel $\tilde{\kappa}$ encodes that two evaluations $f^{(l)}(\boldsymbol{x})$ and $f^{(l)}(\boldsymbol{x}')$ of the same fidelity covary strongly as $\delta_i^{(l)} \delta_i^{l} = 1$. Contrariwise, if $\delta_i^{(l)} \delta_i^{(l')} = 0$ and $l < l'$, the only covariance between the two values of $f^{(l)}(\boldsymbol{x})$ and $f^{(l')}(\boldsymbol{x}')$ is captured by the kernel corresponding to the lower fidelity function. In the remainder of the work, we refer to the standard BO setting where no approximations of f are available as single-fidelity BO.

4.2.3 MF-BMES Algorithm

In this section, we present Multi-Fidelity Batch Min-value Entropy Search (MF-BMES), the multi-fidelity batch Bayesian optimization algorithm, and we start with its single-fidelity simplification, batch min-value entropy search (BMES).

Single-fidelity batch BO BMES is an extension of the standard min-value entropy search (MES) algorithm to the batch setting using the idea of temporary fictitious data sets as in Eq. (4.14) from [DKB14]. Specifically, BMES uses MES acquisition function in the batch framework from Algorithm 3 to select the next point $\boldsymbol{x}_{t,k}$ in the batch. In BMES the computation of the acquisition function is approximated according to Eq. (2.25). BMES is a simplified version of MF-BMES that is suitable for simple calibration problems where the overhead for estimating the gap between different fidelities outweighs the benefits of having access to multiple sources of information.

Multi-fidelity batch BO MF-BMES algorithm, i.e., the multi-fidelity batch min-value entropy search, extends the BMES approach incorporating available objective approximations and their evaluation costs $\{f^{(l)}, c^{(l)}\}_{l \in [L]}$ into the acquisition function. To this end, we define the acquisition function jointly over the parameter space $\mathcal{X}$ and the set of possible fidelities $l \in [L]$:

$$\boldsymbol{x}_t = \underset{\boldsymbol{x} \in \mathcal{X},\ l \in [L]}{\operatorname{argmax}}\, \alpha(\boldsymbol{x}, l).$$

The goal is to maximize the gain in mutual information between the true noiseless optimum $y^* = \min_{\boldsymbol{x} \in \mathcal{X}} f^{(L)}(\boldsymbol{x})$ and the candidate $\boldsymbol{x}$ and its evaluation at l^{th} fidelity $y^{(l)}(\boldsymbol{x})$. Taking the

evaluation cost into account results in the information gain per unit of cost as follows:

$$\alpha(\boldsymbol{x}, l) = \frac{I\big(\{\boldsymbol{x}, y^{(l)}\}; y^*\big)}{c^{(l)}} = \frac{H\big[p(y^{(l)}|\boldsymbol{x})\big] - \mathbb{E}\big[H\big[p(y^{(l)}|\boldsymbol{x}, y^*)\big]\big]}{c^{(l)}}, \tag{4.18}$$

with the expectation over $p(y^*|\mathcal{D})$. At the start of the optimization process, even inaccurate fidelities have a relatively high information gain. Therefore, crude but cheap approximations are favored in this phase. As the optimization continues, we get more data from inaccurate approximations and their information gain reduces while their costs stay the same. This causes the algorithm to progressively favor more accurate fidelities as we refine our knowledge about the objective.

Furthermore, at each iteration, we want to choose a batch of inputs to evaluate. To this end, we adapt the notation as follows: each bias function $b^{(l)}(\boldsymbol{x})$ captures the difference between consecutive fidelity levels $f^{(l)}(\boldsymbol{x})$ and $f^{(l-1)}(\boldsymbol{x})$ is modeled as a $\mathcal{GP}$:

$$b^{(l)}(\boldsymbol{x}) \sim \mathcal{GP}\left(\mu_0^{(l)}(\boldsymbol{x}), \kappa_0^{(l)}(\boldsymbol{x}, \boldsymbol{x}')\right), \qquad l \in [L]_+. \tag{4.19}$$

Following the batch notation in Algorithm 3, the upper subscript denotes the fidelity level and the lower – iteration and batch. Thus, the k^{th} component has the posterior $\mu_{t,k}^{(l)}(\boldsymbol{x})$ and $\kappa_{t,k}^{(l)}(\boldsymbol{x}, \boldsymbol{x}')$:

$$b^{(l)}(\boldsymbol{x}) \sim \mathcal{GP}\left(\mu_{t,k}^{(l)}(\boldsymbol{x}), \kappa_{t,k}^{(l)}(\boldsymbol{x}, \boldsymbol{x}')\right), \qquad l \in [L]_+,\ k \in [B]. \tag{4.20}$$

Given this statistical model, we want to compute Eq. (4.18) for each fidelity. The first term in this equation is the entropy of a Gaussian distribution $p(y^{(l)}|\boldsymbol{x})$, which can be computed in a closed form. The second term is the entropy of a more complicated distribution $p(y^{(l)}|\boldsymbol{x}, y^*)$ that cannot be computed in a closed form. Particularly, conditioned on the given minimum y^*, $y^{(l)}$ is a random variable that results from the algebraic sum of a random variable $y^{(l-1)}$, distributed as a truncated Gaussian, $p(y^{(l-1)}|\boldsymbol{x}, y^*)$ and a bias variable distributed as $p(-b^{(l')}|\boldsymbol{x}, y^*)$, $l' \in \{L, L-1, \cdots, l+1\}$. For instance, $p(y^{(l-1)}|\boldsymbol{x}, y^*) = p(y^{(l)} - b^{(l)}|\boldsymbol{x}, y^*)$. We approximate this distribution with a truncated Gaussian, i.e., $y^{(l)}(\boldsymbol{x}) \geq y^*$, which can be computed in the closed form given the mean and standard deviation of the non-truncated distribution in (4.15) and (4.17) and the truncation interval $[y^*, \infty]$. Finally, Eq. (4.18) is approximated using Monte Carlo estimation by sampling a set of M function minima y^*:

$$\alpha_{t,k}(\boldsymbol{x}, l) \approx \frac{1}{c^{(l)}} \frac{1}{M} \sum_{y^*} \left[\frac{\gamma_{y^*}^{(l)}(\boldsymbol{x}) \psi(\gamma_{y^*}^{(l)}(\boldsymbol{x}))}{2\big(1 - \Psi(\gamma_{y^*}^{(l)}(\boldsymbol{x}))\big)} + \log\big(1 - \Psi(\gamma_{y^*}^{(l)}(\boldsymbol{x}))\big) \right], \tag{4.21}$$

where the notation repeats Eq. (4.13) with a correction that $\gamma_{y^*}^{(l)}(\boldsymbol{x}) = \frac{y^* - \tilde{\mu}_{t,k}(\boldsymbol{x}, l)}{\tilde{\sigma}_{t,k}(\boldsymbol{x}, l)}$ is a standard Gaussian distribution defined for each fidelity $l \in [L]$. Here, $\tilde{\mu}_{t,k}(\boldsymbol{x}, l)$ and $\tilde{\sigma}_{t,k}(\boldsymbol{x}, l)$ are the mean and standard deviation of the corresponding function $f^{(l)}(\boldsymbol{x})$. The minima y^* are sampled following the strategy from [WJ17] (see Eq. (2.25)). Algorithm 4 summarizes MF-BMES approach.

Algorithm 4 Multi-fidelity Batch Min-value Entropy Search (MF-BMES)

Input: Batch size B, Prior $\mu_0^{(0)}(\boldsymbol{x}) = \cdots = \mu_0^{(L)}(\boldsymbol{x}) = 0$, Kernel κ, Costs $\{c^{(l)}\}_{l\in[L]}$
Initialize the fictitious dataset $\mathcal{D}_0 \leftarrow \emptyset$
for $t = 1, 2 \ldots, T$ **do**
  $\tilde{\mathcal{D}}_{t,0} \leftarrow \mathcal{D}_t$
  for $k = 1, 2 \ldots, B$ **do**
    Compute $\tilde{\mu}_{t,k}(\boldsymbol{x}, l)$ and $\tilde{\sigma}_{t,k}(\boldsymbol{x}, l)$ of the posterior $p(f^{(l)}|\tilde{\mathcal{D}}_{t,k-1})$ as in (2.8) and (2.9)
    Select $(\boldsymbol{x}_{t,k}, l_{t,k}) \in \underset{\boldsymbol{x}\in\mathcal{X},\ l\in[L]}{\operatorname{argmax}}\ \alpha_{t,k}(\boldsymbol{x}, l)$ computed via Eq. (4.21)
    Update the fictitious dataset $\tilde{\mathcal{D}}_{t,k} \leftarrow \tilde{\mathcal{D}}_{t,k-1} \cup \{(\boldsymbol{x}_{t,k}, \tilde{\mu}_{t,k}(\boldsymbol{x}_{t,k}, l_{t,k}))\}$
  end for
  Evaluate $y_{t,k} = f^{(l_{t,k})}(\boldsymbol{x}_{t,k}) + \xi_{t,k}, \quad \forall k \in [B]_+$
  Update the true dataset $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(\boldsymbol{x}_{t,k}, y_{t,k})\}_{k=1}^{B}$
end for

Scaling BO to high dimensional problems BO relies on acquisition function optimization which can be difficult to optimize over high-dimensional domains. In order to make it tractable, a line BO suggested by [Kir+19] is successfully used in practice. The idea is to solve one-dimensional BO by choosing iteratively the line in the optimization space and fixing the rest of the high-dimensional calibrating vector. Though the method is simple and intuitive, the algorithm enjoys convergence guarantees while obtaining competitive performance. Analogously to [Kir+19], to perform BO, we iteratively choose a tractable subspace of the domain instead of a line, and thus we call it subdomain BO. In our high-dimensional experiments, we use the block coordinate descent method to define the sub-problems. While this allows us to calibrate more parameters, BO methods are still not able to handle problems with hundreds of thousands of parameters.

4.2.4 BO in the context of transport systems calibration

Calibration methods for transport system simulations have traditionally relied on optimization techniques such as Simultaneous Perturbation Stochastic optimization (SPSA; [Spa98a; Spa98b]). SPSA and its modern modifications are gradient-based methods, which can get stuck in local minima and require multiple evaluations of the simulator to accurately estimate gradients. Moreover, estimating the gradient accurately from noisy measurements can be challenging and require multiple evaluations of the simulator around the same input, which can be computationally expensive. For a more in-depth discussion of the advantages and shortcomings of SPSA-based methods, see the works in [Dju14; Ant+15; TKJ15; Tym18]. Recently, the Opdyts algorithm [Flö17] was introduced to exploit the iterative structure of transport simulations to improve the efficiency of the optimization procedure. At its core, Opdyts manages the computational resources by cutting the simulation time of sub-optimal parameter configurations. However, Opdyts does not really focus on *how* to choose the next candidates to evaluate. Exploiting the domain experts' knowledge to propose new candidates is studied in [OB13; Oso19]. The idea is to substitute the simulator with a cheap-to-evaluate analytical model designed by domain experts.

These methods form two orthogonal families of calibrating approaches: *adaptive config-*

uration evaluation, e.g., Opdyts, and *adaptive configuration selection*, e.g., SPSA or work by [OB13] and its adherents. The methods with smart configuration evaluation allocate resources to examine orders-of-magnitude more configurations than needed for evaluating the objective to completion, and then gradually eliminate poor launches. Though these methods were shown to be flexible and scalable to higher dimensional spaces, they do not learn generally from configurations previously sampled.

BO is a smart configuration selection method that, in contrast to the methods above, incorporates prior knowledge and proposes parameters based on the probabilistic model capturing the density of good configurations. This can be beneficial in two cases: (1) when the resources are limited, BO can avoid launching poor configurations, and (2) when the tuning problem is solved repeatedly, the BO probabilistic model can be used for transfer learning, i.e., reused as an informative prior in a new setting. For example, once calibrated for a particular city, the probabilistic model can be used for speeding up the calibration for other cities. However, BO is not as effective for high-dimensional problems and is the best fit for small-scale problems or when tuning key parameters in large models. Examples of transport model applications that require calibrating a small number of parameters can be found in [18; ZKN19; BCA15]. Finally, the two calibration families, i.e., adaptive configuration evaluation and selection, are not mutually exclusive and the successful use cases are different. The idea of combining them was successfully applied to tuning machine learning models ([FKH18]).

4.2.5 Experiments

In this section, we empirically examine the methods for calibrating MATSim – the agent- and activity-based transport simulation framework. We first introduce the simulation, the input data, and two calibration problems. Second, we compare the proposed BO-based methods and baselines tuning the key MATSim parameters. Finally, we show how BO adapts to higher dimensions.

Transport systems simulator: a closer look

We use eqasim[3], an extension to the transport simulation framework MATSim ([HNA16]). Eqasim combines the queue-based network simulation component of MATSim with discrete mode choice models ([HBA19c]). In comparison with the evolutionary choice-making algorithm of standard MATSim, eqasim has smoother simulation dynamics from iteration to iteration that is beneficial for calibration algorithms that rely on iterative structure, e.g., [Flö17].

In the simulation, a large number of agents have daily plans consisting of multiple activities to be done at certain locations in a capacitated road network. The activities are connected by trips taken using certain *modes of transport* (private car, public transport, biking, and walking). The plans are simulated in a queue-based network that allows for spillback effects to occur. The output of the simulation includes all realized travel plans of the agents, including congestion, and measures travel times on each road network link.

[3] http://www.eqasim.org

These travel times are then fed back into the discrete mode choice model provided by eqasim, which makes transport mode decisions for each agent based on the traffic conditions of the previous network simulation. By running this loop between the network simulation and choice models, the system dynamics eventually stabilize, with small fluctuations in mode decisions but generally stable mode share when observed on a population average.

The transport mode choice is modeled by a multinomial logit ([Tra09]) that is used to predict the probability of choosing a mode m given the expected utility u_m gained by choosing mode m, i.e.:

$$P(m) = \frac{\exp(u_m)}{\sum_{m'} \exp(u_{m'})}. \tag{4.22}$$

The idea is that each mode has a certain utility that is calculated using a linear combination of mode-specific attributes. For example, the utility function for a public transport trip can be written as a linear combination of attributes such as the number of transfers required, the time spent in the vehicle, and the cost of the trip:

$$u_{\text{pt}}(z) = \beta_{\text{ASC,pt}} + \beta_{\text{nmbOfTransfers}} z_{\text{nmbOfTransfers}} \tag{4.23}$$

$$+ \beta_{\text{inVehicleTime}} z_{\text{inVehicleTime}} + \beta_{\text{cost}} \left(\frac{z_{\text{crowflyDistance}}}{\theta_{\text{averageDistance}}} \right)^{\lambda} z_{\text{cost,pt}} + \dots, \tag{4.24}$$

where β denotes sensitivities to attributes of trips and travelers or alternative-specific constants (ASCs). The fixed model constants are denoted as θ and z are attributes measured and derived from the simulation. The mode choice model is used within the context of a tour-based approach, which means that the choice set is constrained by additional conditions such as the need to bring a private vehicle back home. For more details about the mode choice, please refer to [HBA19c].

Calibrated parameters: Zurich case study

We apply BO for calibrating the MATSim parameters for the canton of Zurich ([Hör+19]). The model is based on detailed census data and the Swiss nationwide household travel survey ([Fed17]), and has been used in studies on topics such as automated vehicles ([HBA19b; Hör+19; HBA19a]), car-sharing ([Bal+19]), and Urban Air Mobility ([Bal+18; BRH19]). We use the eqasim extension to simulate transport mode decisions in the model.

The true values of the model parameters are obtained from a survey conducted in Zurich. In practice, such surveys and corresponding models may not be available when setting up MATSim. Alternatively, we can impose an existing discrete choice model structure and use an algorithm to calibrate the parameters so that certain metrics are replicated in the simulation. In this work, we focus on two settings for calibration: first, we calibrate only the alternative specific constants in Eq. (4.24), and second, we calibrate both the alternative specific constants and the value of time per mode. Specifically, we aim to recover the true values of parameters such as $\beta_{\text{ASC,car}}, \beta_{\text{ASC,pt}}, \beta_{\text{ASC,bike}}$ and $\beta_{\text{ASC,walk}}$ while keeping other parameters fixed at their true values. For consistency with the rest of the dissertation, we denote β with as x, and $\boldsymbol{x} = (x_{car}, x_{pt}, x_{bike})$.

Data aggregation

We calibrate the simulator according to two metrics: (1) mode share distribution by the number of trips, and (2) travel time distribution for car trips. We construct the mode share distribution by the percentage of trips with this mode to all performed trips:

$$q_m = \frac{n_m}{\sum_{m' \in \mathcal{M}} n_{m'}}, \qquad \forall m \in \mathcal{M} = \{\text{car}, \text{pt}, \text{bike}, \text{walk}\}, \tag{4.25}$$

Similarly, for travel time distribution, we focus on trips taken using the car mode and record their travel time t. We use reference data to determine the upper boundaries of four travel time bins B, with the 20th, 40th, 60th, and 80th percentiles serving as the quantiles. Any travel times greater than 80% are placed in an additional bin, resulting in a total of five bins. For each bin B, we count the number of simulation n_B and divide it by the total number of trips to calculate the distribution:

$$q_B = \frac{n_B}{\sum_{B' \in \{1,...5\}} n_{B'}}, \qquad \forall B \in \{1, ..., 5\}. \tag{4.26}$$

Objective functions

It is important to carefully choose a suitable objective function because the quality of the calibration process depends on it. The objective function should accurately reflect the desired level of agreement between the simulation and real-world data.

Given the parameters $\boldsymbol{x}$, a simulation outputs measurements $Q(\boldsymbol{x}) = \{q_i(\boldsymbol{x})\}$ representing modes' share distributions or travel time distributions. The true observations are denoted as $R = \{r_i\}$ binned in the same way as $Q(\boldsymbol{x})$. Here, the true distributions are obtained from a full-fidelity with 40 MATSim iterations and the ground truth parameters as presented in subsection 4.2.5. We consider two types of objectives below.

L2 Error for mode share distribution In our experiments, the mode share distribution $Q(\boldsymbol{x})$ is represented by four bins for the modes {car, pt, bike, walk}. The L_2 error is defined as follows:

$$f(\boldsymbol{x}) = \|Q(\boldsymbol{x}) - R\|_2 = \sqrt{\sum_m (q_m(\boldsymbol{x}) - r_m)^2}. \tag{4.27}$$

Hellinger Distance for travel time distribution The Hellinger distance is a measure of similarity between two probability distributions that is often used in statistics. It is defined as the Euclidean norm of the difference between the square roots of the elements in the vectors representing the distributions. This metric is popular due to its ability to capture the difference between the distributions while still being invariant to monotonic

transformations.

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n} (\sqrt{q_i(\boldsymbol{x})} - \sqrt{r_i})^2}. \tag{4.28}$$

Fidelity levels

Our multi-fidelity optimization framework involves approximations of the objective function at different levels of fidelity, which are defined based on the number of iterations or population fraction of a MATSim simulation. These approximations allow us to balance the computational cost of evaluating the objective function with the accuracy of the simulation results. We define the low-fidelity approximations $\{f^l(\boldsymbol{x})\}_{l=0}^{L-1}$ for the objective functions $f(\boldsymbol{x})$ from (4.27) and (4.28) based on the mismatch between the reference observations R and the output of a low-fidelity simulation with parameters $\boldsymbol{x}$, $Q^l(\boldsymbol{x})$. In particular, $Q^l(\boldsymbol{x})$ contains the aggregate measurements from a MATSim simulation for a lower number of iterations $n^l \in \{n^0, \dots n^{L-1}\}$ or smaller population fraction $s^l \in \{s^0, \dots s^{L-1}\}$:

$$f^l(\boldsymbol{x}) = f(Q^l(\boldsymbol{x}), R), \qquad \forall l \in \{0, \dots, L-1\}, \tag{4.29}$$

$$f(\boldsymbol{x}) = f(Q(\boldsymbol{x}), R). \tag{4.30}$$

This choice of approximations is motivated by the numerical experiments shown in Figures 4.10 and 4.11. Figure 4.10 depicts how the simulation run time changes when altering population fraction and number of iterations. Figure 4.11 presents a high correlation in behavior between the evaluations of the objective and its approximations both for a simple objective such as the mode share distribution and a more complex objective such as the travel time distribution.

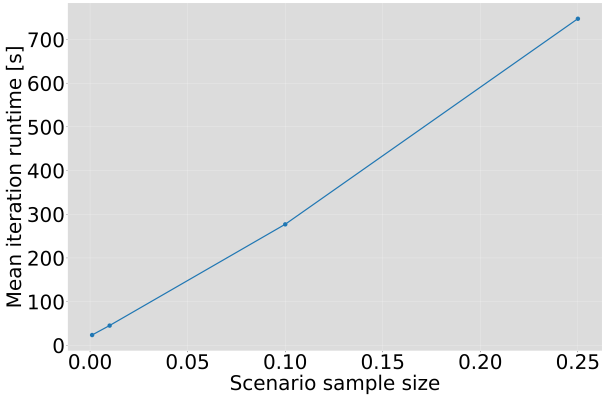
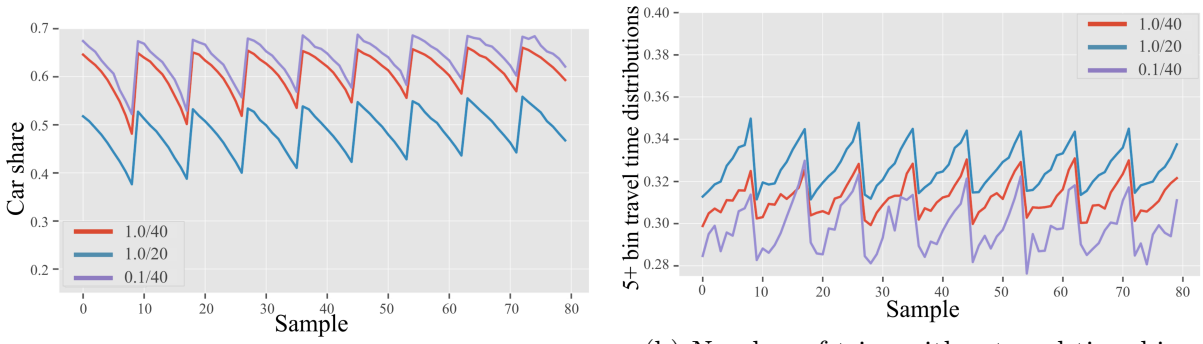


Figure 4.10: Mean timing for one MATSim iteration for different population fractions in a simulation with 32 threads with 100G memory. One can notice that acquiring a full-fidelity (40 iterations) evaluation for 10% population takes more than 3 hours.

Model selection for GPs

In practice, one needs to specify the form of the function prior and its parameters, also known as model hyperparameters. In this work, we use a constant mean function:



(a) Car share value for different samples

(b) Number of trips with a travel time bin $[5, 10)$

Figure 4.11: Examples of correlation between objective function evaluations (in red) and its approximations for Matsim. The full-fidelity (red) is taken such that the population fraction equals to 1.0 and the number of iterations equals 40. Blue lines represent the output for a lower number of iterations such that the population fraction equals to 1.0 and the number of iterations equals 20. Violet lines represent the output for a smaller population fraction such that the population fraction equals 0.1 and the number of iterations equals to 40. Each sample represents one evaluated parameter setting: they come from a uniform three-dimensional grid over the domain of variables $x_{car} \in [-1, 1], x_{bike} \in [-1, 1], x_{walk} \in [0, 2]$ that contains 9 points per dimension. One can notice seesaw behavior caused by the fact that every 9 points same value of the x_{car} is evaluated while changing the values for the x_{bike} and x_{walk}.

$\mu(\boldsymbol{x}) = m$ that we treat as a hyperparameter of our model. The covariance function expresses our belief about how function values for different inputs covary with each other and, thus, it encodes our belief about the smoothness of the function f. In Figure 4.12 we show which kind of functions are likely under different priors by plotting four samples from four different $\mathcal{GP}$s with covariance functions expressing different degrees of smoothness and $\mu(\boldsymbol{x}) = 0$. In particular, in our experiments, we use Matern kernel 3/2 for the lower fidelities and Matern kernel 5/2 for modeling the bias function which represents the most realistic assumptions on the smoothness of our function f. These kernels are defined as follows:

$$\kappa_{5/2}(\boldsymbol{x}, \boldsymbol{x}') = \eta_0^2\big(1 + \sqrt{5r^2(\boldsymbol{x}, \boldsymbol{x}')} + \frac{5}{3}r^2(\boldsymbol{x}, \boldsymbol{x}')\big) \exp\big\{ - \sqrt{5r^2(\boldsymbol{x}, \boldsymbol{x}')}\big\}, \tag{4.31}$$

$$\kappa_{3/2}(\boldsymbol{x}, \boldsymbol{x}') = \eta_0^2\big(1 + \sqrt{3r^2(\boldsymbol{x}, \boldsymbol{x}')}\big) \exp\big\{ - \sqrt{3r^2(\boldsymbol{x}, \boldsymbol{x}')}\big\}, \tag{4.32}$$

$$r^2(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D}(x_d - x'_d)/\eta_d^2, \qquad \boldsymbol{x} = (x_1, \ldots, x_D), \tag{4.33}$$

and are parametrized by a vector of kernel hyperparameters $\boldsymbol{\eta}_k = \{\eta_0, \eta_1, \ldots, \eta_D\}$, where η_0 stands for standard deviation of f and η_d is characteristic lengthscales for each dimension d. Intuitively, the characteristic lengthscales define how far apart the input values $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be for the corresponding evaluations y_i and y_j to become uncorrelated. These lengthscales $\{\eta_d\}_{d=1}^{D}$ can be equal for all the dimensions $d = \{1, \ldots, D\}$ or, as it is done in our experiments, can be separately defined by automatic relevance determination, ARD ([Nea96]), that implicitly defines the "relevance" of each dimension.

Once the form of the prior is fixed, we have to determine the hyperparameters $\boldsymbol{\eta} = (m, \sigma_\varepsilon, \boldsymbol{\eta}_k)$ based on the observed data. The most commonly advocated approach for

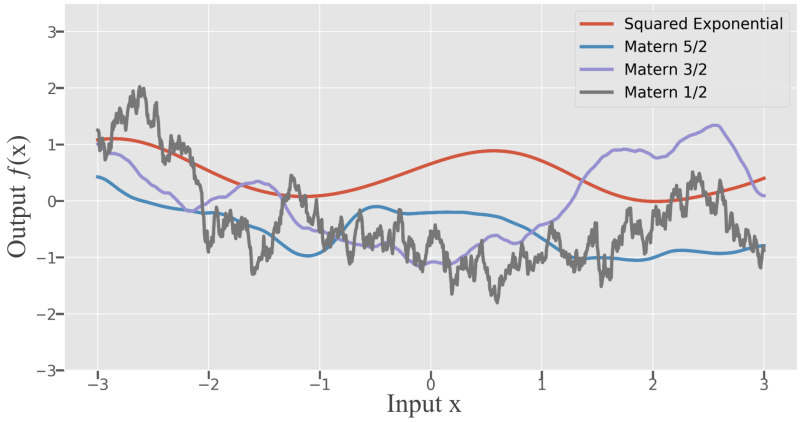


Figure 4.12: Visualization of different kernel functions and their effect on sample functions. The top row shows functions likely under a prior with zero mean and Squared Exponential kernel, while the bottom row shows functions likely under a prior with zero mean and three variations of the Matern kernel (5/2, 3/2, and 1/2). The sample functions with the Squared Exponential kernel appear to be overly smooth, while those with the Matern kernel appear to be excessively noisy.

choosing the unknown parameters $\boldsymbol{\eta}$ is to use a point estimate of $\boldsymbol{\eta}$. Particularly, $\boldsymbol{\eta}$ is chosen such that it maximizes either the likelihood (ML) or the posterior (MAP) of the observed data:

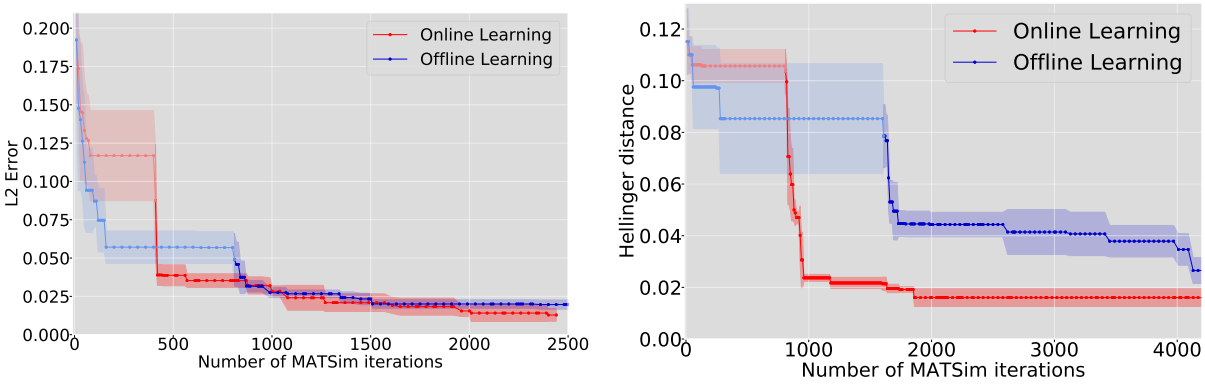
$$\boldsymbol{\eta}^{ML} = \arg\max_{\boldsymbol{\eta}} \ \log\left(p(\boldsymbol{y}|\ \{\boldsymbol{x}_i\}_{i=1}^t, \boldsymbol{\eta})\right), \tag{4.34}$$

$$\boldsymbol{\eta}^{MAP} = \arg\max_{\boldsymbol{\eta}} \ \log\left(p(\boldsymbol{y}|\ \{\boldsymbol{x}_i\}_{i=1}^t, \boldsymbol{\eta})p(\boldsymbol{\eta})\right), \tag{4.35}$$

where $\boldsymbol{y} = \{y_1, \ldots, y_t\}$ are observed evaluations at points $\{\boldsymbol{x}_i\}_{i=1}^t$ respectively.
In our experiments, we use MAP estimates proved to be good in practice. One should notice that MAP estimate requires setting a prior $p(\boldsymbol{\eta})$ over the learning parameters, which we set to be an independent log-normal distribution (Lizotte [Liz08]).

Online and Offline learning

There are two approaches to choosing the hyperparameters in our framework: offline learning and online learning. In the offline learning approach, we first collect a fixed number of function evaluations at random points in the domain, and then use these evaluations to compute the maximum a posteriori (MAP) estimate for the hyperparameters. This MAP estimate is then fixed throughout the optimization process. In contrast, the online learning approach starts the optimization with a smaller number of samples and continuously updates the MAP estimate based on the observations obtained from the optimization queries. Generally, offline learning provides a better prior for the objective function before the optimization starts, leading to a faster convergence rate. However, the time and cost overhead of collecting the necessary function evaluations beforehand can be significant, especially in the multi-fidelity setting which requires estimating additional hyperparameters for the different fidelity levels. An example of this overhead is shown in Figure 4.13, where online learning requires twice less data than offline learning.



(a) MF-BMES for Mode share

(b) MF-BMES for Travel Time

Figure 4.13: Comparison of computation times for offline and online learning approaches for MF-BMES. The high-fidelity function consists of 40 MATSim iterations and the low-fidelity function consists of 10 MATSim iterations. The results are averaged over 5 runs for the 0.1% Zurich scenario, with (a) mode share output and L2 error as the objective function and (b) travel time distribution output and Hellinger distance as the objective function. Each point on the lines represents one BO evaluation, with the x-axis representing the cost of the evaluation in terms of the number of internal MATSim iterations. The light sections of the curves represent model selection, while the dark sections represent actual Bayesian optimization. For offline learning in (a), 16 evaluations of high-fidelity and 16 evaluations of low-fidelity are performed on the same input points. For online learning in (a), half of the evaluations are randomly chosen to be performed at high-fidelity, with corresponding evaluations at low-fidelity. In (b), 32 evaluations of high-fidelity and 32 evaluations of low-fidelity are performed on the same input points for offline learning, while half of the evaluations are randomly chosen to be performed at high-fidelity for online learning.

Results

We present the results for three different scenarios: (1) calibrating the alternative specific constants for 0.1% of the population, (2) calibrating the alternative specific constants for 10% of the population and greater box constraints, and, finally, (3) calibrating not only alternative specific constants but also the value of time per mode and others.

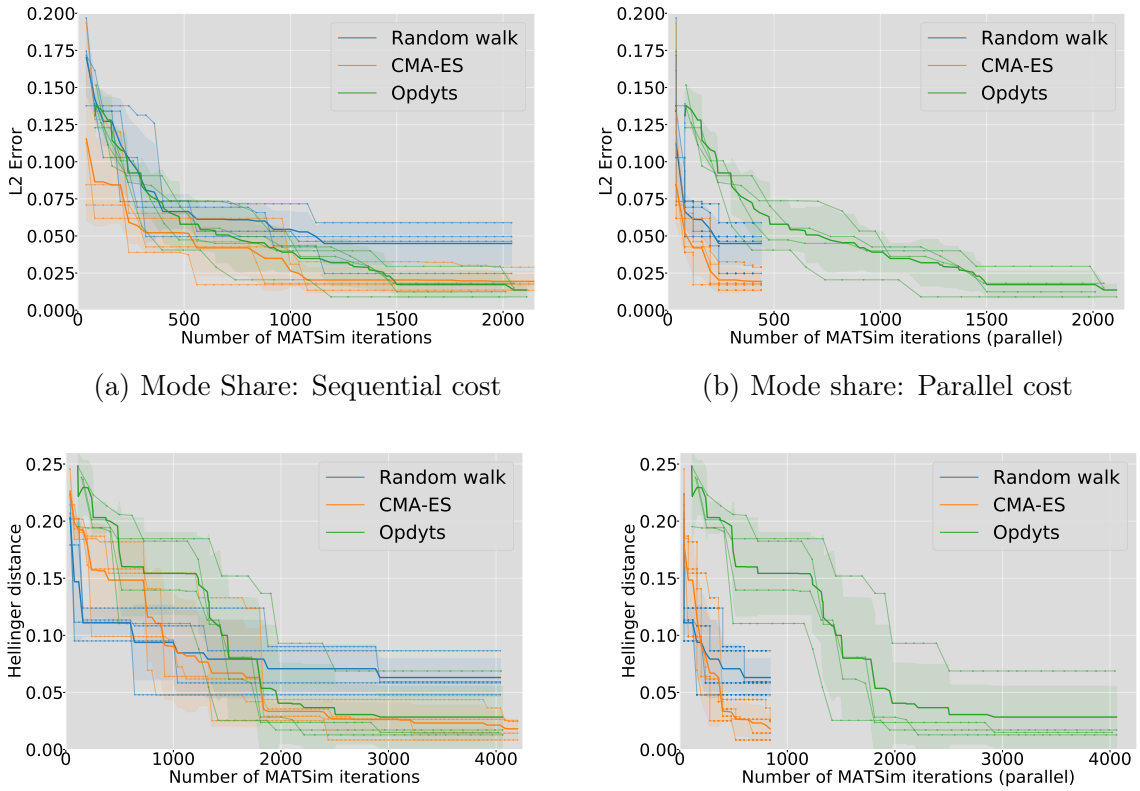
We calibrate MATSim for the Zurich scenario with 0.1% population size. The full-fidelity corresponds to 40 MATSim iterations and the low-fidelity is 10 iterations. The ground truth parameter values to recover are $x_{car} = 0.827$, $x_{pt} = 0.0$, $x_{bike} = 0.344$, and $x_{walk} = 1.3$. The reference data is obtained from simulations with true parameters and full-fidelity. We compare BO-based methods to three approaches: random walk, CMA-ES algorithm ([Han06]) and Opdyts ([Flö17]) described below.

Random walk. The candidate points are chosen uniformly at random over the box-constrained domain using plausible ranges: $x_{car} \in [-1, 1]$, $x_{bike} \in [-1, 1]$, $x_{walk} \in [0, 2]$. We run 4 parallel candidates which is a practical limit for the computing infrastructure at hand. Each evaluation is run for 40 iterations which correspond to high fidelity.

CMA-ES. For CMA-ES[4] we use a candidate set size of 4 parallel evaluations and an initial step size of 0.3. The MATSim simulations run for 40 iterations.

Opdyts. Opdyts is a method that uses the iterative structure of an objective function to find good configurations of parameters. It works by examining the intermediate steps of

[4]We use a commonly used implementation from https://en.wikipedia.org/wiki/CMA-ES, version 29 July 2019.



(a) Mode Share: Sequential cost

(b) Mode share: Parallel cost

(c) Travel time: Sequential cost

(d) Travel time: Parallel cost

Figure 4.14: Comparison between the baselines Random, CMA-ES and Opdyts. Each thin line of the corresponding color represents a run of an algorithm with a different random seed. In bold, we report the mean of these runs. In the left (a,c) plots, we present the flattened-out number of MATSim iterations used by the algorithms as if all evaluations of the algorithms were made sequentially. In the plots on the right (b,d), we take into account the ability of the methods to perform parallel evaluations. This means, for example, that for a random walk the cost in the parallel mode will be 4 times smaller than for sequential. The number of parallel evaluations is set to 4 which is defined by the real capacities for model calibrations. For CMA-ES, we run a batch of 4 parallel MATSim simulations with 40 iterations each, which also shrinks the cost for parallel run by 4 as for random walk. However, in Opdyts only one MATSim simulation at a time is pushed forward for a number of iterations while all others are idle. Only the first transition for each simulation run (which amounts to five MATSim iterations) can be run in parallel. The following transitions are counted as serial.

a simulation (called transitions), and deciding which set of parameters to pursue further. In these experiments, we use a candidate set of size 4, with 5 iterations of the simulation per transition and 5 total transitions per chain (totaling 40 iterations). The candidates are initially sampled from a Gaussian distribution with a standard deviation of 0.1, centered at the intervals of $x_{car} \in [-1, 1]$, $x_{bike} \in [-1, 1]$, $x_{walk} \in [0, 2]$. Opdyts uses an adaptation weight of 0.9 as defined in ([Flö17]), and samples new candidates from a similar Gaussian distribution, with means based on the best parameters found so far.

Note that ([Flö17]) does not focus on generating new promising candidates to evaluate by Opdyts, but rather speeds up the optimization by detecting unpromising candidates early on. The performance of Opdyts depends on the size of the candidate set and we study this dependence in the MATSim case for the sizes $4, 8, 16, 32$ in Figure 4.15. The

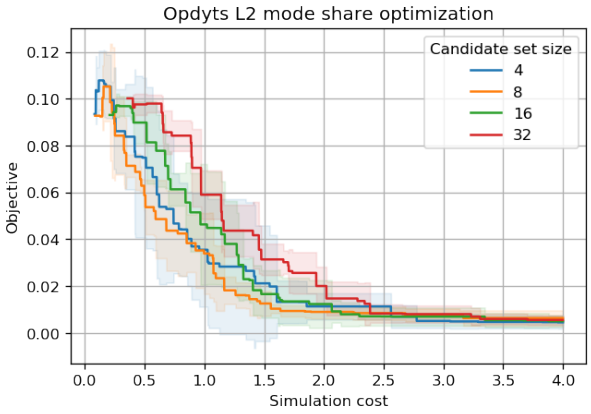


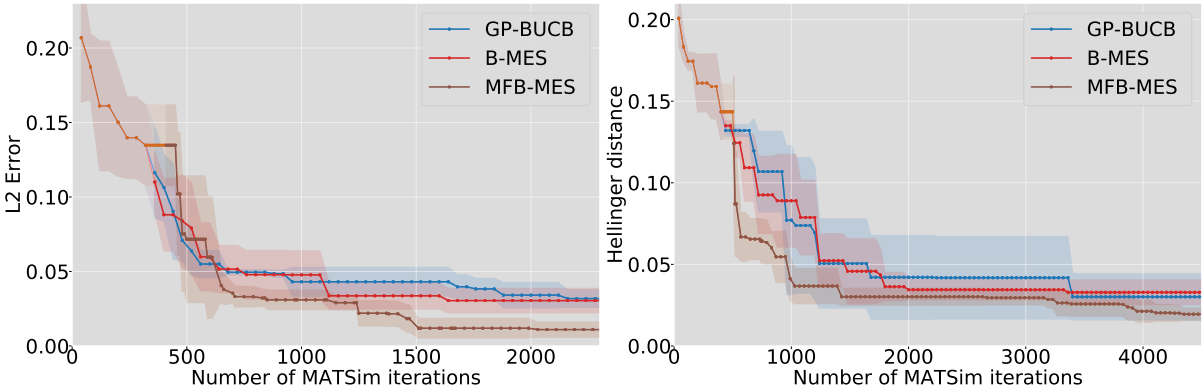
Figure 4.15: Convergence of Opdyts with the different candidate set sizes. The cost corresponds to the simulation cost if the experiments were launched sequentially and it is shown for shown Figure 4.17. Improved convergence is observed with larger candidate set sizes, which can be exploited if greater resource capacities are available. Differences between Opdyts and BO, including the smart use of resources in Opdyts and the smart choice of points for evaluation in BO, are discussed in Section 4.2.4 and Section 4.2.6.

inclusion of Opdyts in our comparison is therefore indicative as even larger candidate sets and different candidate proposal strategies may drastically change the obtained results. Rather, we want to demonstrate that Opdyts can be used in a calibration framework along with Bayesian optimization. In fact, BO for candidates proposals and Opdyts for efficient evaluation can be jointly used in practice. While the BO loop would be able to globally find parameters that are interesting to evaluate, Opdyts would speed up those evaluations by not further evolving candidates that are clearly sub-optimal.

Baselines. In the experiments presented in Figure 4.14, we repeat calibration 4 times for each algorithm with different random seeds and report the mean and standard deviation of the best value of the objective found by each of the methods as a function of the number of MATSim iterations. We also examine the ability of the methods to perform parallel evaluations and depict the results in terms of the number of parallel internal MATSim iterations. It can be seen that for the mode share distribution, the algorithms require fewer MATSim iterations to converge than for travel time distributions. We observe the same ranges of convergence time for other experiments that confirm the assumption of mode share problem being easier than travel time distribution.

Empirical comparison. First, we compare proposed MF-BMES with BO baselines algorithms: B-MES and GP-BUCB described in Section 4.2.3 and Section 4.2.2. As before, we take the full-fidelity to be 40 iterations and lower-fidelity to be 10 iterations with a fixed 0.1% sample of the population. The optimization domain is defined by the box constrains $x_{car} \in [-1, 1]$, $x_{bike} \in [-1, 1]$, $x_{walk} \in [0, 2]$. The results in Figure 4.16 show the comparison in terms of the objective function and number of required internal MATSim iterations. In can be seen that MFB-MES outperforms B-MES and GP-BUCB as it uses cheaper evaluations at the beginning to explore the domain and then, based on the obtained information, acquires high-fidelity evaluations.

Second, we compare MF-BMES to the baselines Opdyts, CMA-ES, and Random walk in Figure 4.17. For the baselines, we report the means and standard deviations



(a) Mode Share (b) Travel Time Distribution

Figure 4.16: Comparing the performance of GP-BUCB, B-MES, and MF-BMES in the Zurich scenario with 0.1% population size. Results are shown for the mean and standard deviation over 4 runs with different random seeds. Each point on the curves represents one function evaluation. For model selection, shown in thin, all algorithms use the same number of high-fidelity evaluations, resulting in overlap at the beginning of the learning curves. MFB-MES also uses low-fidelity evaluations for model selection, leading to a longer stagnation period at the start. For the mode share 8 high-fidelity and 8 low-fidelity evaluations were used, while for the travel time objective, 10 of each were used. Low-fidelity evaluations do not improve the objective function and therefore MFB-MES's learning curve stagnates while exploring the domain and acquiring cheaper evaluations.

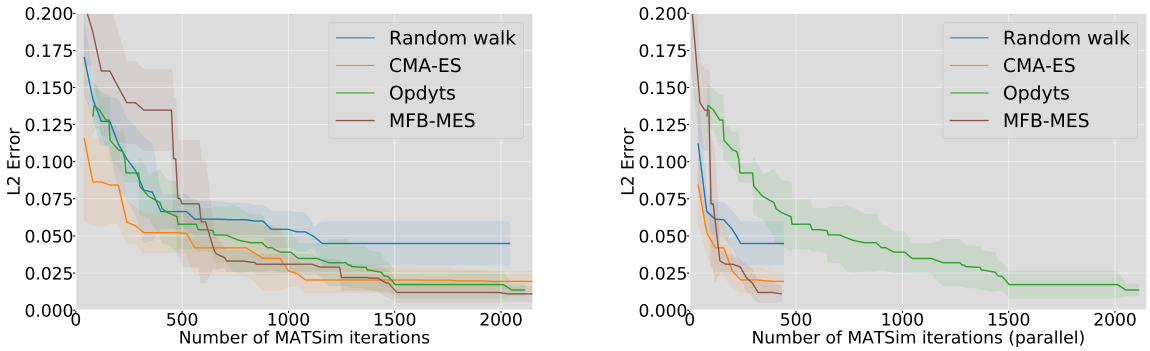
obtained in Figure 4.14. For a fair comparison, we consider the model selection part in MFB-MES as being a part of the optimizations process, thus making the MFB-MES curves in Figure 4.17(a, c) identical to curves in Figure 4.16(a, b) respectively. For the parallel setting in Figure 4.17(b, d), we take into account a batch size of 4 for MFB-MES which is equal to the number of parallel runs of other algorithms.

Mode share is an easier problem and in Figure 4.17 the baselines start to converge faster from the very beginning, while MFB-MES still performs model selection. However, once the model is selected, MFB-MES converges rapidly and achieves the state-of-the-art result. From the plot for travel time distribution, one can conclude that when the calibration problem is hard enough, i.e., not an easy problem as mode share, even with overhead produced by model training, MFB-MES has an advantage over the competing methods. The ability of MFB-MES to be parallelized gains additional profit in performance, especially in comparison with Opdyts.

We visualize intermediate optimization results, such as mode share and travel time distributions and their correspondences to the objective value, in Figure 4.19 for MFB-MES, Opdyts, and CMA-ES. Note that different parameter configurations may lead to similar mode shares/travel time distributions and thus error in the parameter space is not a representative of the quality of the solution to the calibration procedure.

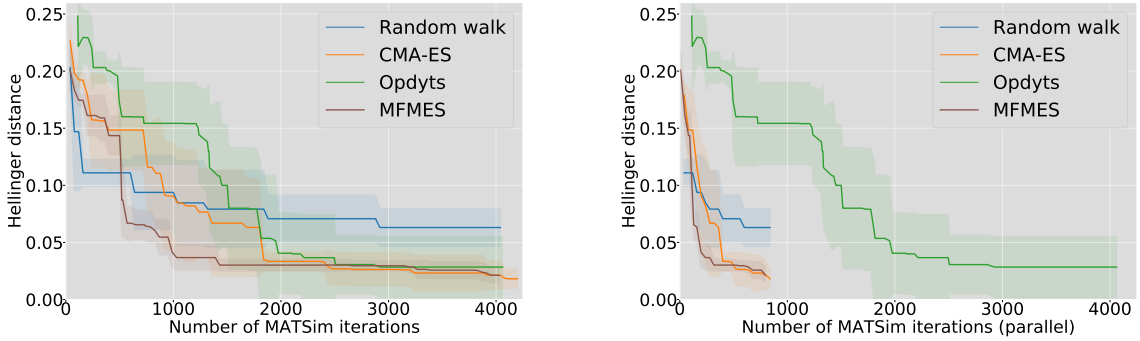
MF-BMES for 10% population of Zurich scenario

In this experiment with 10% population fraction, we explore the behavior of the proposed MF-BMES method on more difficult setting that a practitioner would have in reality: 1) twice bigger box-constrained domain for ACS being $[-2, 2]$ for each mode, 2) the reference



(a) Mode Share: Sequential

(b) Mode Share: Parallel



(c) Travel times: Sequential

(d) Travel times: Parallel

Figure 4.17: Comparison between MFMES, CMA-ES, and Opdyts for the 0.1% population Zurich scenario. We report mean and standard deviations based on 4 runs of the algorithms with different random seeds. In the right plots, we take into account the ability of the methods to perform parallel evaluations. For MFB-MES this means dividing the computational time by the batch size. For the baselines, the curves for the parallel version repeat ones in Figure 4.14.

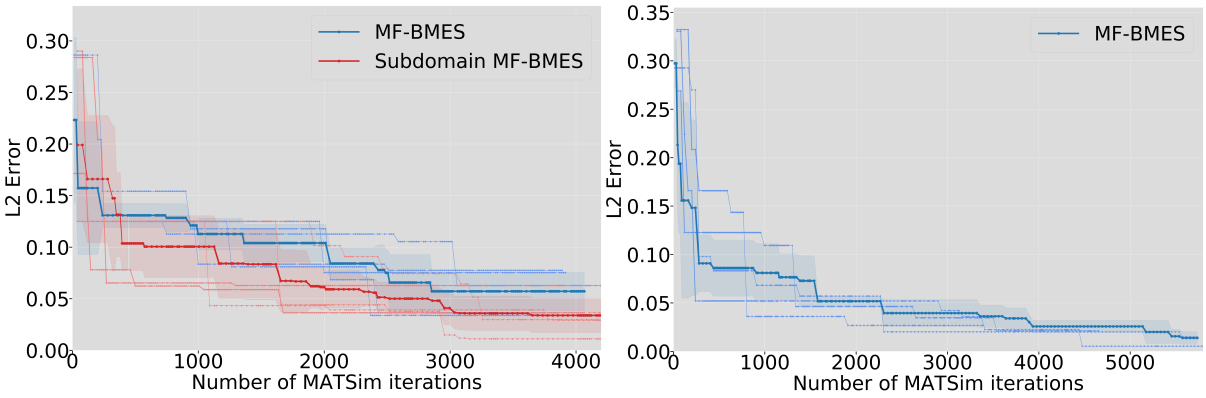
data corresponds to simulation with 25% of the population that reflects the true real data the most. Acquiring each full-fidelity evaluation takes more than 3 hours while low-fidelity evaluation is 4 times cheaper. The results presented in Figure 4.18 (b) depict mean and variance over 4 runs. As before, the value for the objective is updated when a better value for high-fidelity point is acquired. One can see from the plot, that this problem is indeed more difficult and requires more evaluations.

Subdomain MF-BMES for higher dimension problems

In this experiment, we show the behavior of MF-BMES and its subdomain modification described in Section 4.2.3 on 10 parameters. Here, we consider alternative specific constants and the values of time per mode. For the ASC we define a twice bigger box-constrained domain being $[-2, 2]$ and for others, we set the plausible ranges to be $[-1, 1]$. The reference data corresponds to the simulation with 25% of the population.

For the subdomain BO, as an oracle, we use block coordinate descent proposed in [Kir+19], so the sub-problems are defined by the coordinate-aligned directions. We choose the size of the subdomain to be 3, i.e., in each iteration we fix all other parameters to

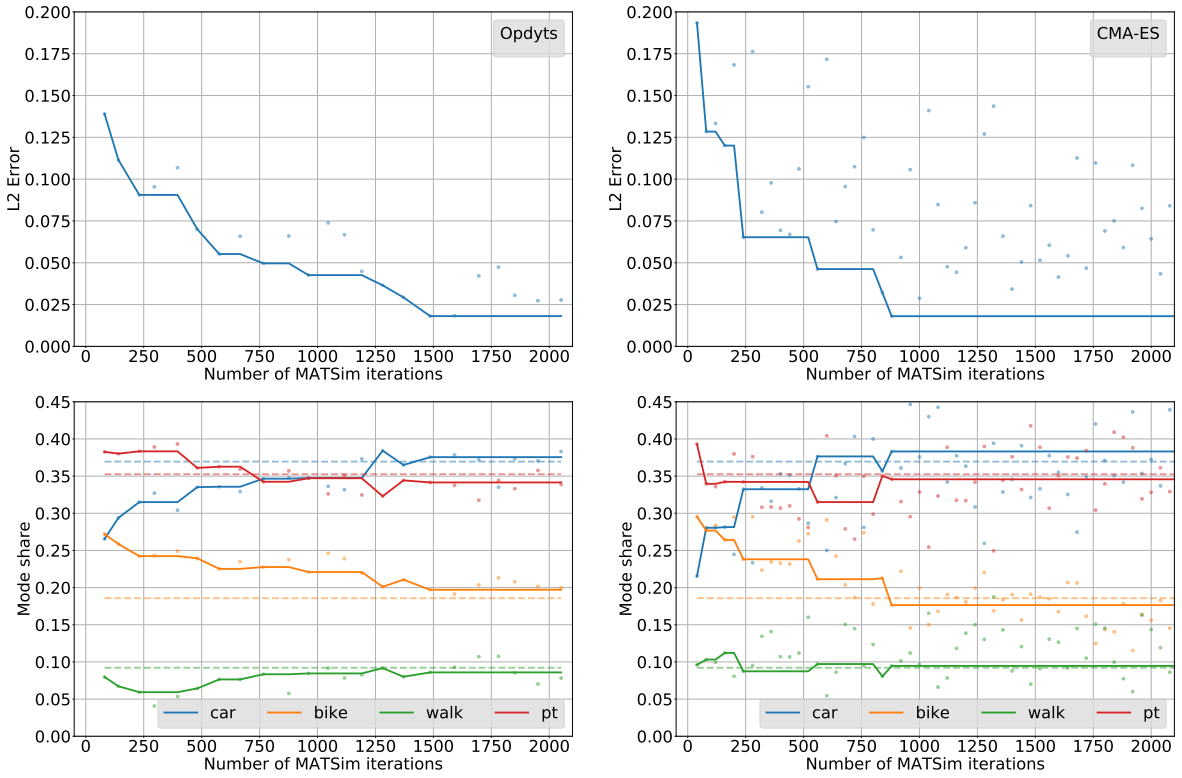
the best-known values so far while optimizing these three. We use 8 low-fidelity and 8 high-fidelity initial points to define the model and then use online learning as described before. The results presented in Figure 4.18 (a) depict mean and variance over 4 runs.



(a) 10 dimensional

(b) 10% population

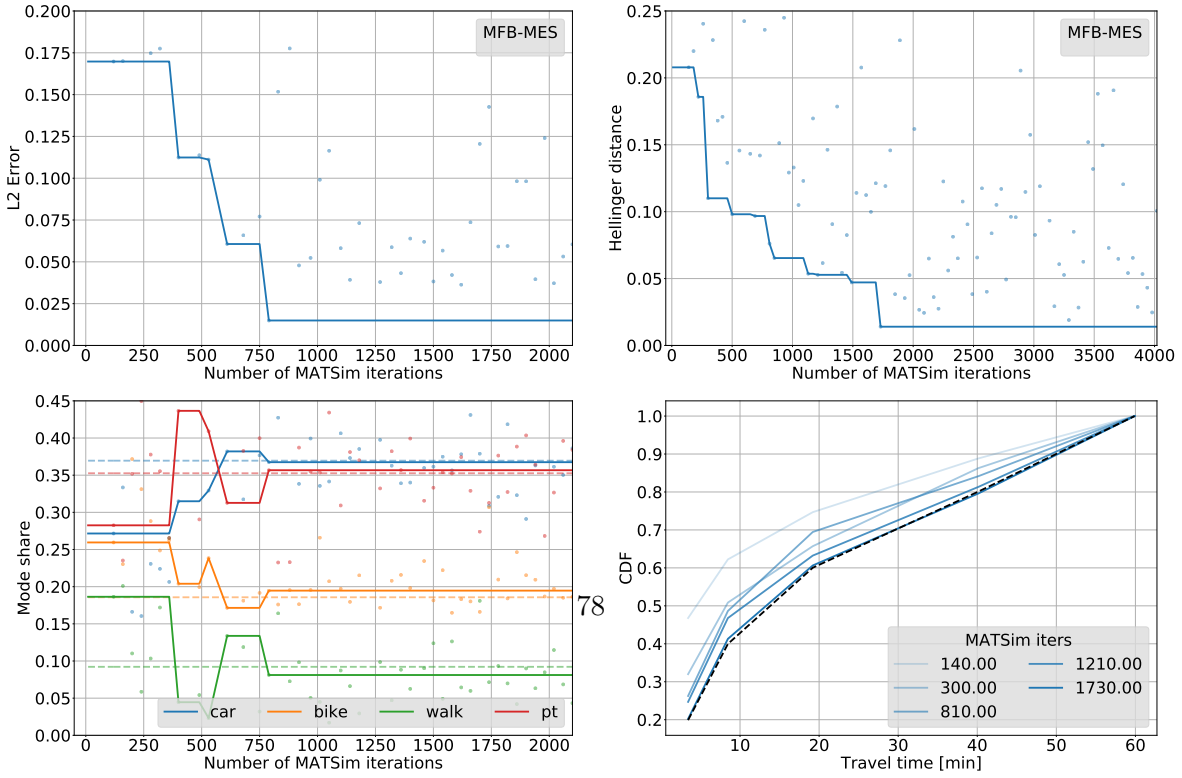
Figure 4.18: Results for the higher scale experiments: (a) for calibrating 10 parameters (Section 4.2.5) and (b) for calibrating on a greater box-constrained domain and 10% population (Section 4.2.5). Both plots depict results for mode share problem; bold curves and shaded areas represent the mean and standard deviation obtained from the thin curves of corresponding colors. Though each experiment uses a batch size of 4 for parallel runs, in the plot we show sequential cost as defined in Figure 4.17, thus in practice, this timing is reduced by 4 times.



(a) Opdyts: Mode share

(b) CMA-ES: Mode share

Figure 4.19: Convergence comparison of MFB-MES, Opdyts, and CMA-ES. The curves are obtained for the best results from Figures 4.14 and 4.16. In each set of two plots, the upper plots show the convergence of the objective function (L2 error and Hellinger distance), which is identical to the corresponding curves in Figure 4.14. The lower plots show the actual convergence of the distributions produced by the calibrated model to target distributions. For mode share (a, b, c), in the lower plot, dashed lines represent the values of target shares and solid lines represent the corresponding changing shares produced by MATSim during the model calibration. For travel time (d, e, f), the lower plot visualizes the cumulative distribution functions (CDF) of target travel time distributions (dashed lines) and CDF of distributions produced by MATSim during the model calibration (solid lines). The different curves in CDF Visualization are obtained after the number of corresponding MATSim iterations (specified in the legend) was used.



4.2.6 Discussion

These results give the first evidence that Bayesian Optimization is suitable for calibrating transport system simulators. One main practical advantage is the automated nature of the process, where the whole calibration routine is treated as a black box and where both the simulator parameters and the hyperparameters of the optimizers are tuned from data without requiring any expertise in these domains from the practitioners.

In order to make MF-MES method useful in practice, one needs to know both its strengths and its weaknesses. In particular, BO becomes problematic when applied in high-dimensional domains due to two reasons: (i) the optimization of the acquisition function becomes itself a hard optimization problem; (ii) performing global exploration requires prohibitively many evaluations since the volume of the search space grows exponentially with the dimensionality of the domain. In the experiments, we show that the adjustment proposed in [Kir+19] covers all the parameters in MATSim and can be used in practice. While the adoption of these techniques allows for calibrating more parameters, BO-based methods are still unable to tackle problems with hundreds of thousands of parameters. Therefore, MF-MES and the existing gradient-based methods are not mutually exclusive rather they should be applied in different circumstances. In particular, whenever the number of calibration variables is small our BO-based methods should be preferred over other black-box optimization methods studied in the transport simulators calibration literature due to their sample efficiency and the global nature of its search. On the other hand, when the number of calibration variables is high, existing gradient-based methods should be preferred as they improve the calibration objective more quickly since they do not make exploratory evaluations. In conclusion, BO-based methods should be used to tune a few, critical parameters that have the largest impact on the calibration procedure while standard, gradient-based methods should be used to tune a large amount of parameters whose individual contribution toward calibration accuracy is low to moderate.

Also, it should be pointed out that the relatively long calibration times, as they are presented in this work, are not given for a specified convergence criterion. As can be seen, the objective values become rather small after a few iterations. Defining adequate stopping criteria is subject to future work. While the paper at hand establishes the theoretical and methodological foundation for applying Bayesian Optimization to the calibration of a transport system simulator it will be up to the transport planning community to decide when the calibration process reaches the desired fit.

To that end, the framework has been published as open source for the community to be tested. A major part of future work will be to streamline the implementation and to make the approach accessible to a larger variety of simulators and calibration objectives, which may include count calibration, calibration of modes shares by distance, and others.

Moreover, we highlight the fact that, while the approximation to the second term of the MF-BMES acquisition function introduced in this work performs well in practice, in future research it would be interesting to investigate the impact of different approximations for such a term.

Furthermore, it remains to show that the approach is applicable to real-world planning

scenarios, which often are one magnitude larger than the examples presented in this work. For instance, in [SBF18], the authors calibrate a full-fidelity scenario for Switzerland with around 8 million agents. While the manual calibration has taken several months to finish, it will be interesting to assess how well the BO framework is able to support calibration endeavors of such scale.

Another point that needs to be emphasized is that transport problems do not only rely on one specific objective ([SBF18]). Rather, there are multiple criteria such as the total passenger distance traveled or the number of passengers per station that need to be calibrated jointly as is in the case in the mentioned Switzerland scenario. Therefore, in future work, we intend to apply multi-objective Bayesian optimization in combination with the existing multi-fidelity batch BO presented here.

Finally, we would like to mention that an interesting direction for future research is to investigate how some of the advancements that were recently proposed in the calibration domain for other optimizers could benefit BO-based methods. In particular, we think that the use of prior knowledge suggested by [OB13] and the smart resource management suggested by [Flö17] could be readily applied to BO-based methods and greatly improve their efficiency.

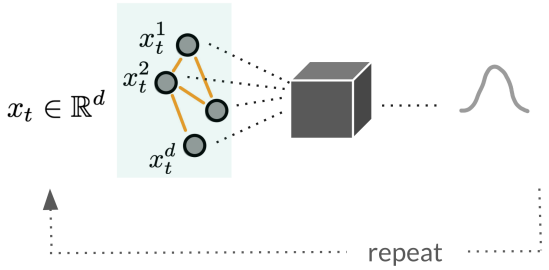
4.2.7 Conclusion

In this chapter, we propose an automatic low-dimensional calibration procedure for transport system simulators based on batch multi-fidelity Bayesian optimization. The calibration problem is formulated as an optimization problem where the mismatch between data obtained from the simulation and real data is minimized with respect to the simulator parameters being calibrated. To tackle this non-convex optimization problem that has no analytic form and computationally expensive evaluations of its objective function, we introduce the algorithms Multi-Fidelity Batch Min-value Entropy Search (MF-BMES) and Batch Min-value Entropy Search (B-MES).

We show the effectiveness of MF-BMES and BMES methods for mode share and travel time distributions as calibration targets on the Zurich scenario that is used in transport planning. While the experiments were set up for a specific transport system simulator, the proposed framework can be used to calibrate any (agent-based) transport simulator that has long turnover times for the simulation runs, but has the possibility to use cheaper approximations of the full simulation. We show how to adapt MF-BMES to higher dimensional problems and propose a way to combine them with existing calibration routines. Therefore, as part of our contribution, we release an open-source implementation of our method. The work shows the first, in our knowledge, evidence that Bayesian optimization is suitable for the calibration of transport system simulators.

CHAPTER 5

Bayesian optimization over structured domains



Encoding the inductive biases, such as the problem's structure, into the probabilistic models allows for more efficient algorithms. Most of the existing Bayesian optimization literature focuses on objectives that have purely *continuous* domains, such as those arising in the tuning of continuous hyperparameters of machine learning algorithms, recommendation systems, and preference learning [Sha+16]. More recently, problems with purely *discrete* domains, such as food safety control and model-sparsification in multi-component systems [BP18] have been considered. However, many real-world optimization problems in science and engineering are of *mixed-variable* nature, involving *both* continuous and discrete input variables, and exhibit complex constraints. For example, tuning the hyperparameters of a convolutional neural network involves both continuous variables, e.g., learning rate and momentum, and discrete ones, e.g., kernel size, stride, and padding. Also, these hyperparameters impose validity constraints, as some combinations of kernel size, stride and padding define invalid networks. Further examples of mixed-variable, potentially constrained optimization problems include sensor placement [KSG08], drug discovery [NFP11], optimizer configuration [HHL11], to name a few. Nonetheless, only a few BO methods can address the unconstrained version of such a problem, and no existing method can handle the constrained one. This work introduces the first algorithm that can efficiently optimize mixed-variable functions subject to known constraints with provable convergence guarantees.

In the chapter, we introduce MiVaBo, a novel BO algorithm for the efficient optimization of mixed-variable functions combining a linear surrogate model based on expressive feature representations with Thompson sampling. We propose an effective method to op-

timize its acquisition function, a challenging problem for mixed-variable domains, making MiVaBo the first BO method that can handle complex constraints over discrete variables. Moreover, we provide the first convergence analysis of a mixed-variable BO algorithm. Finally, we show that MiVaBo is significantly more sample efficient than state-of-the-art mixed-variable BO algorithms on several hyperparameter tuning tasks, including the tuning of deep generative models. This chapter is based on our paper "Mixed-Variable Bayesian Optimization" [Dax+20].

Our Contributions. We introduce MiVaBo, the first BO algorithm for efficiently optimizing mixed-variable functions subject to known linear and quadratic integer constraints, encompassing many of the constraints present in real-world domains (e.g. cardinality, budget and hierarchical constraints). It relies on a linear surrogate model that decouples the continuous, discrete and mixed components of the function using an expressive feature expansion (Sec. 5.2.1). We exploit the ability of this model to efficiently draw samples from the posterior over the objective (Sec. 5.2.2) by combining it with Thompson sampling, and show how to optimize the resulting constrained acquisition function (Sec. 5.2.3). While in continuous BO, optimizing the acquisition function is difficult but has well-established solutions, this is not true for mixed-variable spaces and doing this efficiently and accurately is a key challenge that hugely impacts the algorithm's performance. We also provide the first convergence analysis of a mixed-variable BO algorithm (Sec. 5.2.5). Finally, we demonstrate the effectiveness of MiVaBo on a set of complex hyperparameter tuning tasks, where it outperforms state-of-the-art methods and is competitive with human experts (Sec. 5.3).

Related Work. Extending *continuous* BO methods [Sha+16] to mixed inputs requires ad-hoc relaxation methods to map the problem to a fully continuous one and rounding methods to map the solution back. This ignores the original domain structure, makes the solution quality dependent on the relaxation and rounding methods, and makes it hard to handle discrete constraints. Extending *discrete* BO methods [BP18; Oh+19] to mixed inputs requires a discretization of the continuous domain part, the granularity of which is crucial: If it is too small, the domain becomes prohibitively large; if it is too large, the domain may only contain poorly performing values of the continuous inputs. Few BO methods address the mixed-variable setting. SMAC [HHL11] uses a random forest surrogate model. However, its frequentist uncertainty estimates may be too inaccurate to steer the sampling. TPE [Ber+11b] uses kernel density estimation to find inputs that will likely improve upon and unlikely perform worse than the incumbent solution. While SMAC and TPE can handle hierarchical constraints, they cannot handle more general constraints over the discrete variables, e.g., cardinality constraints. They also lack convergence guarantees. Hyperband (HB) [Li+18] uses cheap but less accurate approximations of the objective to dynamically allocate resources for function evaluations. BOHB [FKH18] is the model-based counterpart of HB, based on TPE. They thus extend existing mixed-variable methods to the multi-fidelity setting rather than proposing new ones, which is complementary to our approach, rather than in competition with it. Surrogate modeling over a mixed-variable domain is addressed in [BH10]; however, the authors omit acquisition function optimization, which is a challenging problem, especially

in mixed-variable domains. Moreover, the authors leave the important part of choosing the model features to its user. [GH18] propose a Gaussian process kernel to model discrete inputs without rounding bias. Their method lacks guarantees and cannot handle discrete constraints. We instead use discrete optimizers for the acquisition function, which avoid bias by only making integer evaluations. Finally, while [Her+15; Gar+14; Sui+15] extend continuous BO methods to handle unknown constraints, no method can handle known discrete constraints in a mixed-variable domain.

5.1 Problem Formulation

We consider the problem of optimizing an unknown, costly-to-evaluate function defined over a mixed-variable domain, accessible through noisy evaluations and subject to known linear and quadratic constraints. Formally, we aim to solve

$$\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \tag{5.1}$$

$$\text{subject to } g^c(\boldsymbol{x}) \geq 0, \tag{5.2}$$

$$g^d(\boldsymbol{x}) \geq 0, \tag{5.3}$$

where $\mathcal{X} \subseteq \mathcal{X}^c \times \mathcal{X}^d$ with continuous subspace $\mathcal{X}^c$ and discrete subspace $\mathcal{X}^d$. Both constraints $g^c(\boldsymbol{x}) \geq 0$ over $\mathcal{X}^c$ and $g^d(\boldsymbol{x}) \geq 0$ over $\mathcal{X}^d$ are known, and specifically $g^d(\boldsymbol{x})$ are linear or quadratic. We assume that the domain of the continuous inputs is box-constrained and can thus, w.l.o.g., be scaled to the unit hypercube, $\mathcal{X}^c = [0, 1]^{D_c}$. We further assume, w.l.o.g., that the discrete inputs are binary, i.e., vectors $\boldsymbol{x}^d \in \mathcal{X}^d = \{0, 1\}^{D_d}$ are vertices of the unit hypercube. This representation can effectively capture the domain of any discrete function. For example, a vector $\boldsymbol{x}^d = [x_i^d]_{i=1}^{D_d} \in \mathcal{X}^d$ can encode a subset A of a ground set of D_d elements, such that $x_i^d = 1 \Leftrightarrow a_i \in A$ and $x_i^d = 0 \Leftrightarrow a_i \notin A$, yielding a set function. Alternatively, $\boldsymbol{x}^d \in \mathcal{X}^d$ can be a binary encoding of integer variables, yielding a function defined over integers. We sequentially interact with the unknown objective and observe noise-perturbed evaluations $y_t \triangleq f(\boldsymbol{x}_t) + \varepsilon$ with $\varepsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \beta^{-1})$, $\beta > 0$. To this end, at each iteration, we optimize the acquisition function that provides the next input to evaluate. However, in our case, this optimization problem involves mixed variables and exhibits linear and quadratic constraints and is thus still challenging. We now present MiVaBo, an algorithm to efficiently solve the optimization problem in Eq. (5.1).

5.2 MiVaBo Algorithm

We first introduce the linear model used to represent the objective in Section 5.2.1 and describe how to do inference with it in Section 5.2.2. We then show how to use Thompson sampling to query informative inputs in Section 5.2.3 and, finally, provide a bound on the regret incurred by MiVaBo in Section 5.2.5.

5.2.1 Model

We propose a surrogate model that accounts for both discrete and continuous variables in a principled way, while balancing two conflicting goals: Model expressiveness versus feasibility of Bayesian inference and of the constrained optimization of the mixed-variable acquisition function. Linear models defined over non-linear feature mappings, $f(\boldsymbol{x}) = \omega^\top \boldsymbol{\phi}(\boldsymbol{x})$, are a class of flexible parametric models that strike a good trade-off between model capacity, interpretability and ease of use through the definition of features $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^M$. While the complexity of the model is controlled by the number of features, M, its capacity depends on their definition. Therefore, to make the design of a set of expressive features more intuitive, we treat separately the contribution to the objective f from the discrete part of the domain, from the continuous part of the domain, and from the interaction of the two,

$$f(\boldsymbol{x}) = \textstyle\sum_{j \in \{d,c,m\}} {\omega^j}^\top \boldsymbol{\phi}^j(\boldsymbol{x}^j) \tag{5.4}$$

where, for $j \in \{d, c, m\}$, $\boldsymbol{\phi}^j(\boldsymbol{x}^j) = [\phi_i^j(\boldsymbol{x}^j)]_{i=1}^{M_j} \in \mathbb{R}^{M_j}$ and $\omega^j \in \mathbb{R}^{M_j}$ are the feature and weight vector for the *d*iscrete, *c*ontinuous and *m*ixed function component, respectively.

In many real-world domains, a large set of features can be discarded *a priori* to simplify the design space. It is common practice in high-dimensional BO to assume that only low-order interactions between the variables contribute significantly to the objective, which was shown for many practical problems [Rol+18; MK18], including deep neural network hyper-parameter tuning [HKY17]. Similarly, we focus on features defined over small subsets of the inputs. Formally, we consider $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_k(\boldsymbol{x}_k)]_{k=1}^M$, where $\boldsymbol{x}_k$ is a subvector of $\boldsymbol{x}$ containing exclusively continuous or discrete variables or a mix of both. Thus, the objective $f(\boldsymbol{x})$ can be decomposed into a sum of low-dimensional functions $f_k(\boldsymbol{x}_k) \triangleq w_k \phi_k(\boldsymbol{x}_k)$ defined over subspaces $\mathcal{X}_k \subseteq \mathcal{X}$ with $\dim(\mathcal{X}_k) \ll \dim(\mathcal{X})$. This defines a *generalized additive model* [Rol+18; Has17], where the same variable can be included in multiple subvectors/features. The complexity of this model is controlled by the *effective dimensionality* (ED) of the subspaces, which is crucial under limited computational resources. In particular, let $\bar{D}_d \triangleq \max_{k \in [M]} \dim(\mathcal{X}_k^d)$ denote the ED of the discrete component in Eq. (5.4), i.e. the dimensionality of the largest subspace that exclusively contains discrete variables. Analogously, $\bar{D}_c$ and $\bar{D}_m$ denote the EDs of the continuous and mixed component, respectively. Intuitively, the ED corresponds to the maximum order of the variable interactions present in f. Then, the number of features $M \in \mathcal{O}\big(D_d^{\bar{D}_d} + D_c^{\bar{D}_c} + (D_d + D_c)^{\bar{D}_m}\big)$ scales exponentially in the EDs only (as modeling up to L-th order interactions of N inputs requires $\sum_{l=0}^{L} \binom{N}{l} \in \mathcal{O}(N^L)$ terms), which are usually small, even if the true dimensionality is large.

Discrete Features $\boldsymbol{\phi}^d$. We aim to define features ϕ^d that can effectively represent the discrete component of Eq. (5.4) as a linear function, which should generally be able to capture arbitrary interactions between the discrete variables. To this end, we consider all subsets S of the discrete variables in $\mathcal{X}^d$ (or, equivalently, all elements S of the powerset $2^{\mathcal{X}_d}$ of $\mathcal{X}_d$) and define a monomial $\prod_{j \in S} x_j^d$ for each subset S (where for $S = \emptyset$, $\prod_{j \in \emptyset} x_j^d = 1$). We then form a weighted sum of all monomials to yield the multi-linear polynomial ${\omega^d}^\top \boldsymbol{\phi}^d(\boldsymbol{x}^d) = \sum_{S \in 2^{\mathcal{X}_d}} w_S \prod_{j \in S} x_j^d$. This functional representation corresponds

to the Fourier expansion of a *pseudo-Boolean function* (PBF) [BH02]. In practice, an exponential number of features can be prohibitively expensive and may lead to high-variance estimators as in BO one typically does not have access to enough data to robustly fit a large model. Alternatively, [BP18; HKY17] empirically found that a second-order polynomial in the Fourier basis provides a practical balance between expressiveness and efficiency, even when the true function is of higher order. In our model, we also consider quadratic PBFs, ${\omega^d}^\top \boldsymbol{\phi}^d(\boldsymbol{x}^d) = w_\emptyset + \sum_{i=1}^n w_{\{i\}} x_i^d + \sum_{1 \le i < j \le n} w_{\{i,j\}} x_i^d x_j^d$, which induces the discrete feature representation $\boldsymbol{\phi}^d(\boldsymbol{x}^d) \triangleq [1, \{x_i^d\}_{i=1}^{D_d}, \{x_i^d x_j^d\}_{1 \le i < j \le D_d}]^\top$ and reduces the number of model weights to $M_d \in \mathcal{O}(D_d^2)$.

Continuous Features $\boldsymbol{\phi}^c$. In BO over continuous spaces, most approaches are based on Gaussian process (GP) models [WR06] due to their flexibility and ability to capture large classes of continuous functions. To fit our linear model formulation, we leverage GPs' expressiveness by modeling the continuous part of our model in Eq. (5.4) using feature expansions $\boldsymbol{\phi}^c(\boldsymbol{x}^c)$ that result in a finite linear approximation of a GP. One simple, yet theoretically sound, choice is the class of Random Fourier Features (RFFs) [RR08], which use Monte Carlo integration for a randomized approximation of a GP. Alternatively, one can use Quadrature Fourier Features [MK18], which instead use numerical integration for a deterministic approximation, which is particularly effective for problems with low effective dimensionality. Both feature classes were successfully used in BO [Jen+17; MK18]. In our experiments, we use RFFs approximating a GP with a squared exponential kernel, which we found to best trade off complexity vs. accuracy in practice.

Mixed Features $\boldsymbol{\phi}^m$. The mixed term should capture as rich and realistic interactions between the discrete and continuous variables as possible while keeping model inference and acquisition function optimization efficient. To this end, we stack products of all pairwise combinations of features of the two variable types, i.e. $\boldsymbol{\phi}^m(\boldsymbol{x}^d, \boldsymbol{x}^c) \triangleq [\phi_i^d(\boldsymbol{x}^d) \cdot \phi_j^c(\boldsymbol{x}^c)]_{1 \le i \le M_d, 1 \le j \le M_c}^\top$. This formulation provides a good trade-off between modeling accuracy and computational complexity. In particular, it allows us to reduce $\boldsymbol{\phi}^m$ to the discrete feature representation $\boldsymbol{\phi}^d$ when conditioned on a fixed assignment of continuous variables $\boldsymbol{\phi}^c$ (and vice versa). This property is crucial for optimizing the acquisition function, as it allows us to optimize the mixed term of our model by leveraging the tools for optimizing the discrete and continuous parts individually. The proposed representation contains $M_d M_c$ features, resulting in a total of $M = M_d + M_c + M_d M_c$. To reduce model complexity, prior knowledge about the problem can be incorporated into the construction of the mixed features. In particular, one may consider the following approaches. Firstly, one can exploit a known interaction structure between variables, e.g., in form of a dependency graph, and ignore the features that are known to be irrelevant. Secondly, one can start by including all of the proposed pairwise feature combinations and progressively discard not-promising ones. Finally, for high-dimensional problems, one can do the opposite and progressively add pairwise feature combinations, starting from the empty set.

5.2.2 Model Inference

Let $\mathbf{X}_{1:t} \in \mathbb{R}^{t \times D}$ be the matrix whose i^{th} row contains the input $\boldsymbol{x}_i \in \mathcal{X}$ queried at iteration i, $\dim \mathcal{X} = D$, and let $\mathbf{y}_{1:t} = [y_1, \ldots, y_t]^\top \in \mathbb{R}^t$ be the array of the corresponding noisy function observations. Also, let $\boldsymbol{\Phi}_{1:t} \in \mathbb{R}^{t \times M}$ be the matrix whose i^{th} row contains the featurized input $\boldsymbol{\phi}(\boldsymbol{x}_i) \in \mathbb{R}^M$. The formulation of f in Eq. (5.4) and the noisy observation model induce the Gaussian likelihood $p(\mathbf{y}_{1:t}|\mathbf{X}_{1:t}, \omega) = \mathcal{N}(\boldsymbol{\Phi}_{1:t}\omega, \beta^{-1}\mathbf{I})$. To reflect our *a priori* belief about the weight vector ω and thus f, we specify a prior distribution over ω. A natural choice for this is a zero-mean isotropic Gaussian prior $p(\omega|\alpha) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$, with precision $\alpha > 0$, which encourages ω to be uniformly small, so that the final predictor is a sum of all features, each giving a small, non-zero contribution. Given the likelihood and prior, we infer the posterior $p(\omega|\mathbf{X}_{1:t}, \mathbf{y}_{1:t}, \alpha, \beta) \propto p(\mathbf{y}_{1:t}|\mathbf{X}_{1:t}, \omega, \beta)p(\omega|\alpha)$, which due to conjugacy is Gaussian, $p(\omega|\mathbf{X}_{1:t}, \mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{m}, \mathbf{S}^{-1})$, with mean $\mathbf{m} = \beta\mathbf{S}^{-1}\boldsymbol{\Phi}_{1:t}^\top \mathbf{y}_{1:t} \in \mathbb{R}^M$ and precision $\mathbf{S} = \alpha\mathbf{I} + \beta\boldsymbol{\Phi}_{1:t}^\top\boldsymbol{\Phi}_{1:t} \in \mathbb{R}^{M \times M}$ [WR06]. This simple analytical treatment of the posterior distribution over ω is a main benefit of this model, which can be viewed as a GP with a linear kernel in feature space.

Sparse Prior. While the number and degree of the features used in the model is a design choice, in practice it is typically unknown which variable interactions matter and thus which features to choose. To discard irrelevant features, one may impose a sparsity-encouraging prior over the weight vector ω [BP18]. However, due to non-conjugacy to the Gaussian likelihood, exact Bayesian inference of the resulting posterior distribution is in general intractable, imposing the need for approximate inference methods. One choice for such a prior is the Laplace distribution, for which approximate inference techniques based on expectation propagation [Min01] and variational inference [WJ+08] were developed in [See08; SN08; SN11]. Alternatively, one can use a horseshoe prior and use Gibbs sampling to sample from the posterior over weights [BP18]. However, this comes with a significantly larger computational burden, which is a well-known issue for sampling based inference techniques [Bis06]. Lastly, one may consider a spike-and-slab prior with expectation propagation for approximate posterior inference [HHD13; HHS15].

5.2.3 Acquisition Function

We propose to use Thompson sampling (TS) [Tho33], which samples weights $\widetilde{\omega} \sim p(\omega|\mathbf{X}_{1:t}, \mathbf{y}_{1:t}, \alpha, \beta)$ from the posterior and chooses the next input by solving $\widehat{\boldsymbol{x}} \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} \widetilde{\omega}^\top \boldsymbol{\phi}(\boldsymbol{x})$. TS intuitively focuses on inputs that are plausibly optimal and has previously been successfully applied in discrete and continuous domains [BP18; MK18].

TS requires solving $\widehat{\boldsymbol{x}} \in \arg\min_{\boldsymbol{x} \in \mathcal{X}} \widetilde{\omega}_t^\top \boldsymbol{\phi}(\boldsymbol{x})$, which is a challenging mixed-variable optimization problem. However, as $\widetilde{\omega}_t^\top \boldsymbol{\phi}(\boldsymbol{x})$ decomposes as in Eq. (5.4), we can naturally use an alternating optimization scheme which iterates between optimizing the discrete variables $\boldsymbol{x}^d$ conditioned on a particular setting of the continuous variables $\boldsymbol{x}^c$ and vice versa, until convergence to some local optimum. While this scheme provides no theoretical guarantees, it is simple and thus widely and effectively applied in many contexts where the objective is hard to optimize. In particular, we iteratively solve $\widehat{\boldsymbol{x}}^d \in \arg\min_{\boldsymbol{x}^d \in \mathcal{X}^d} \left(\widetilde{\omega}^{d\top} \boldsymbol{\phi}^d(\boldsymbol{x}^d) + \widetilde{\omega}^{m\top} \boldsymbol{\phi}^m(\boldsymbol{x}^d, \boldsymbol{x}^c = \widehat{\boldsymbol{x}}^c)\right)$, $\widehat{\boldsymbol{x}}^c \in \arg\min_{\boldsymbol{x}^c \in \mathcal{X}^c} \left(\widetilde{\omega}^{c\top} \boldsymbol{\phi}^c(\boldsymbol{x}^c) +\right.$

$\widetilde{\omega}^{m\top}\boldsymbol{\phi}^m(\boldsymbol{x}^d = \widehat{\boldsymbol{x}}^d, \boldsymbol{x}^c)\big)$. Importantly, using the mixed features proposed in Sec. 5.2.1, these problems can be optimized by purely discrete and continuous optimizers, respectively. This also holds in the presence of mixed constraints $g^m(\boldsymbol{x}) \geq 0$ if those decompose accordingly into discrete and continuous constraints.

This scheme leverages independent subroutines for discrete and continuous optimization: For the discrete part, we exploit the fact that optimizing a second-order pseudo-Boolean function is equivalent to a binary integer quadratic program (IQP) [BH02], allowing us to exploit commonly-used efficient and robust solvers such as Gurobi or CPLEX. While solving general binary IQPs is NP-hard [BH02], these optimizers are in practice very efficient for the dimensionalities we consider (i.e., $D_d < 100$). This approach allows us to use any functionality offered by these tools, such as the ability to optimize objectives subject to linear constraints $\mathbf{A}\boldsymbol{x}^d \leq \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{K \times D_d}, \mathbf{b} \in \mathbb{R}^K$ or quadratic constraints $\boldsymbol{x}^{d\top}\mathbf{Q}\boldsymbol{x}^d + \mathbf{q}^\top \boldsymbol{x}^d \leq b$, $\mathbf{Q} \in \mathbb{R}^{D_d \times D_d}, \mathbf{q} \in \mathbb{R}^{D_d}, b \in \mathbb{R}$. For the continuous part, one can use optimizers commonly used in continuous BO, such as L-BFGS or DIRECT. In our experiments, we use Gurobi as the discrete and L-BFGS as the continuous solver within the alternating optimization scheme, which we always run until convergence.

5.2.4 Model Discussion

BO algorithms are comprised of three major design choices: the surrogate model to estimate the objective, the acquisition function to measure informativeness of the inputs and the acquisition function optimizer to select queries. Due to the widespread availability of general-purpose optimizers for continuous functions, continuous BO is mostly concerned with the first two design dimensions. However, this is different for mixed-variable constrained problems. We show in Sec. 5.3 that using a heuristic optimizer for the acquisition function optimization leads to poor queries and, therefore, poor performance of the BO algorithm. Therefore, the tractability of the acquisition function optimization influences and couples the other design dimensions. In particular, the following considerations make the choice of a linear model and TS the ideal combination of surrogate and acquisition function for our problem. Firstly, the linear model is preferable to a GP with a mixed-variable kernel as the latter would complicate the acquisition function optimization for two reasons: (i) the posterior samples would be arbitrary nonlinear functions of the discrete variables and (ii) it would be non-trivial to evaluate them at arbitrary points in the domain. In contrast, our explicit feature expansion solves both problems, while second order interactions provide a valid discrete function representation [BP18; HKY17] and lead to tractable quadratic MIPs with capacity for complex discrete constraints. Moreover, Random Fourier Features approximate common GP kernels arbitrarily well, and inference in MiVaBo scales linearly with the number of data points, making it applicable in cases where GP inference, which scales cubically with the number of data points, would be prohibitive. Secondly, TS induces a simple relation between the surrogate and the resulting optimization problem for the acquisition function, allowing to trade off model expressiveness and optimization tractability, which is a key challenge in mixed-variable domains. Finally, the combination of TS and the linear surrogate facilitates the convergence analysis described in Sec. 5.2.5, making MiVaBo the first mixed-variable BO method

with theoretical guarantees.

Optimization with theoretical guarantees. Alternatively, one can minimize Eq. (5.4) using dual decomposition, which is a powerful approach based on Lagrangian optimization and has been successfully used for many problems [KPT11; SGJ11; RC12]. Its well-studied theoretical properties facilitate the convergence analysis of MiVaBo, making it particularly useful for settings where optimization accuracy is of crucial importance. Due to lack of space, we refer the reader to Section 7.4 for details on the dual decomposition and its derivation for our problem.

5.2.5 Convergence Analysis

Using a linear model and Thompson sampling, we can leverage convergence analysis from linearly parameterized multi-armed bandits, a well-studied class of methods for solving structured decision making problems [AL+17]. These also assume the objective to be linear in features $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^M$ with a fixed but unknown weight vector $\omega \in \mathbb{R}^M$, i.e. $\mathbb{E}[f(\boldsymbol{x})|\boldsymbol{\phi}(\boldsymbol{x})] = \omega^\top \boldsymbol{\phi}(\boldsymbol{x})$, and aim to minimize the *total regret* up to time T: $\mathcal{R}(T) = \sum_{t=1}^{T}(f(\boldsymbol{x}_*) - f(\boldsymbol{x}_t))$. We obtain the following regret bound for MiVaBo:

Proposition 1. *Consider the stochastic linear model $y(\boldsymbol{x}) = f(\boldsymbol{x}) + \xi$, i.e., $f(\boldsymbol{x}) = \omega^\top \boldsymbol{x}$, with a fixed but unknown parameter ω, perturbed by zero-mean sub-Gaussian noise ξ with variance-proxy σ^2. Let $\mathcal{X} \subset \mathbb{R}^M$ be a (finite or infinite) compact set and $\boldsymbol{x}_* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. Let μ_t and Σ_t be the mean and variance, respectively, of the posterior distribution of ω Eq. (2.3). Further assume running* MiVaBo *Eq. (2.26) with the following conditions being maintained:*

1. $\widetilde{\omega}_t \sim \mathcal{N}(\mu_t, 24M \ln T \ln \frac{1}{\delta}\Sigma_t^{-1})$, *i.e. with scaled variance.*
2. $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x}} \widetilde{\omega}_t{}^\top \boldsymbol{x}$ *is selected exactly.*[1]
3. $\|\widetilde{\omega}_t\|_2 \leq c, \|\boldsymbol{x}_t\|_2 \leq c, \|f(\boldsymbol{x}_*) - f(\boldsymbol{x}_t)\|_2 \leq c$, $c \in \mathbb{R}_+$

Then, the following holds with probability at least $1-\delta$ for any $T \geq 1$:

$$R(T) \leq \tilde{\mathcal{O}}\left(M^{3/2}\sqrt{T}\ln\frac{1}{\delta}\right).$$

Prop. 1 follows from Theorem 1 in [AL+17] and works for infinite arms $\boldsymbol{x} \in \mathcal{X}, |\mathcal{X}| = \infty$. In our setting, both the discrete and continuous Fourier features (and, thus, the mixed features) satisfy the standard boundedness assumption, such that the proof indeed holds. Prop. 1 implies *no-regret*, $\lim_{T\to\infty} \mathcal{R}(T)/T = 0$, i.e., convergence to the global minimum, since the minimum found after T iterations is no further away from $f(\boldsymbol{x}_*)$ than the mean regret $\mathcal{R}(T)/T$. To our knowledge, MiVaBo is the first mixed-variable BO algorithm for which such a guarantee is known to hold.

[1] To this end, one can use more expensive but theoretically backed optimization methods instead of the alternating one, such as the powerful and popular *dual decomposition* [SGJ11].

5.3 Experiments

We present experimental results on tuning the hyperparameters of two machine learning algorithms, namely gradient boosting and a deep generative model, on multiple datasets.

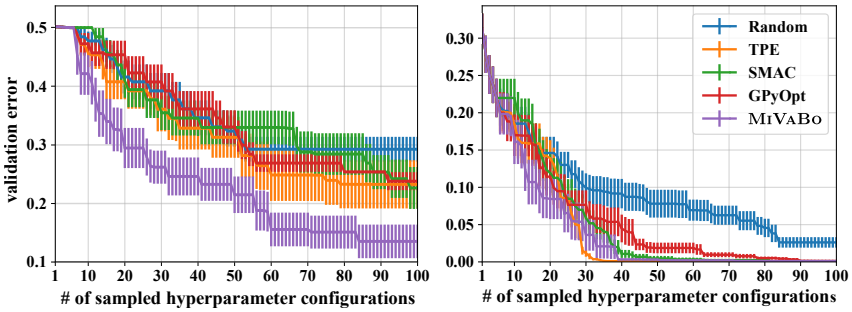


Figure 5.2: XGBoost hyperparameter tuning on `monks-problem-1` (left) and `steel- plates-fault` (right). Mean ± one std. of the validation error over 16 random seeds. MiVaBo significantly outperforms the baselines on the first dataset, and is competitive on the second.

Figure 5.3: Randomly chosen `MNIST` test images (left column) and their reconstructions by the best VAE models found by MiVaBo, random search, GPyOpt, TPE and SMAC (left to right), thus ordered by NLL values, which seem to capture visual quality.

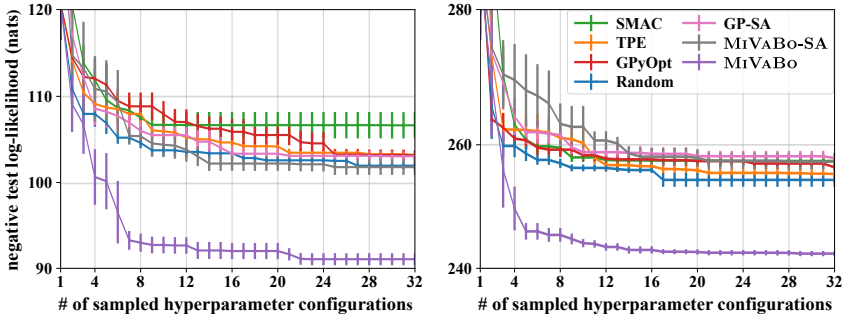


Figure 5.4: VAE hyperparameter tuning on `MNIST` (left) and `FashionMNIST` (right). Mean ± one std. of the NLL in nats, estimated using 32 importance samples, over 8 random seeds. Every model was trained for 32 epochs. MiVaBo significantly outperforms the state-of-the-art baselines, demonstrating its ability to handle the complex constrained nature of the VAE's parameter space.

Method	Time	NLL
SMAC	0.32s	99.09
TPE	0.12s	97.05
GPyOpt	0.65s	97.33
Random	0.01s	93.74
MiVaBo	7.39s	**84.25**

Figure 5.5: Mean wall-clock time of one iteration (excluding function evaluation time) and mean negative log-likelihood (NLL) in nats, estimated with 5000 importance samples, of the best VAEs found after 32 BO iterations (as in Figure 5.6), when trained for 3280 epochs. Human expert baseline for even deeper models is 82-83 nats.

Experimental Setup. For MiVaBo[2], we set the prior variance α, observation noise variance β, and kernel bandwidth σ to 1.0, and scale the variance as stated in Prop. 1. We compare against SMAC, TPE, random search, and the popular GPyOpt BO package. GPyOpt uses a GP model with the upper confidence bound acquisition function [Sri+10], and accounts for mixed variables by relaxing discrete variables to be continuous and later

[2]We provide a `Python` implementation of MiVaBo at `https://github.com/edaxberger/mixed_variable_bo`.

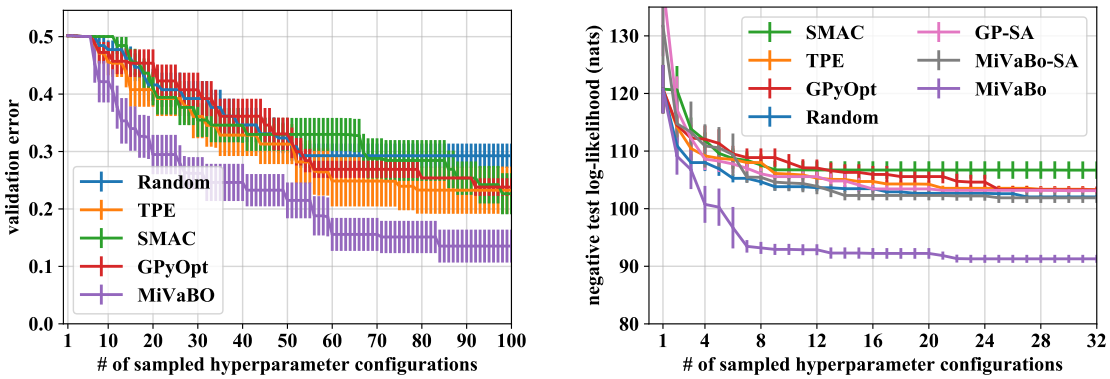


Figure 5.6: (Left) XGBoost hyperparameter tuning results on the `monks-problem-1` dataset. Mean plus/minus one standard deviation of the validation error over 16 random initializations. MiVaBo significantly outperforms the competing state-of-the-art methods. (Right) VAE hyperparameter tuning results on the `MNIST` dataset. Mean plus/minus one standard deviation of the negative test log-likelihood (NLL) in nats, estimated using 32 importance samples, over 8 random initializations. Every model was trained for 32 epochs. MiVaBo significantly outperforms the competing state-of-the-art methods, demonstrating its ability to handle the complex constrained nature of the VAE's parameter space.

Table 5.1: For the best models found by each algorithm in Figure 5.6 (left) after 32 BO iterations, we report the negative log-likelihood (NLL), estimated with 5000 importance samples, after training for 3280 epochs. We also report the NLL achieved after 32 training epochs, as in Figure 5.6 (left)

Algorithm	NLL (nats)	
	32 epochs	3280 ep.
SMAC	106.69 ± 1.51	99.09
TPE	103.28 ± 0.49	97.05
GPyOpt	103.25 ± 0.58	97.33
Random	102.04 ± 0.41	93.74
MiVaBo	$\mathbf{91.29 \pm 0.69}$	**84.25**

Table 5.2: Randomly chosen `MNIST` test images (left column) and their reconstructions by the best VAE models found by MiVaBo, random search, GPyOpt, TPE and SMAC (from left to right), thus ordered by NLL values, which seem to capture quality of visual appearance.

rounding them to the nearest discrete neighbor. To separate the influence of model choice and acquisition function optimization, we also consider the MiVaBo model optimized by simulated annealing (SA) (MiVaBo-SA) and the GP approach optimized by SA (GP-SA). We compare against the SA-based variants only in constrained settings, using more principled methods in unconstrained ones. To handle constraints, SA assigns high energy values to invalid inputs, making the probability of moving there negligible. We use SMAC, TPE and GPyOpt and SA with their respective default settings.

We assess three different surrogate models for this setting:

1. An agnostic model which doesn't know the feature representation of the true objective, i.e., where we use the full discrete feature vector of size $M_d \in \mathcal{O}(D_d^2)$ and the full mixed feature vector of size $M_d M_c$. We place a Gaussian prior over ω. This simulates a realistic setting where the objective function is unknown.

2. The model as in 1., but with a Laplacian prior over ω. This is to assess the effect of

imposing a sparse prior on the weights in a setting where the true objective actually is sparse (due to the feature subsampling).

3. A model which knows the feature design, i.e., which uses the same features as the underlying objective. We again place a Gaussian prior over ω. The goal of the algorithm now is to recover the fixed (i.e., randomly sampled) but unknown weight vector ω. This simulates the setting where we know that our model class captures the true objective (which is e.g. also assumed for our regret bounds to hold).

5.3.1 Gradient Boosting Tuning

The OpenML database [Van+14] contains evaluations for various machine learning methods trained on several datasets with many hyperparameter settings. We consider extreme gradient boosting (XGBoost) [CG16], one of the most popular OpenML benchmarks, and tune its ten hyperparameters – three are discrete and seven continuous – to minimize the classification error on a held-out test set (without any constraints). We use two datasets, each containing more than 45000 hyperparameter settings. To evaluate hyperparameter settings for which no data is available, we use a surrogate modeling approach based on nearest neighbor [Egg+15], meaning that the objective returns the error of the closest (w.r.t. Euclidean distance) setting available in the dataset. Figure 5.2 shows that MiVaBo achieves performance which is either significantly stronger than (left dataset) or competitive with (right dataset) the state-of-the-art mixed-variable BO algorithms on this challenging task. GPyOpt performs poorly, likely because it cannot account for discrete variables in a principled way. As compared to TPE and SMAC, MiVaBo seems to benefit from more sophisticated uncertainty estimation.

5.3.2 Deep Generative Model (DGM) Tuning

DGMs recently received considerable attention in the machine learning community. Despite their popularity and importance, effectively tuning their hyperparameters is a major challenge. We consider tuning the hyperparameters of a variational autoencoder (VAE) [KW14] composed of a convolutional encoder and a deconvolutional decoder [SKW15]. The VAEs are evaluated on stochastically binarized `MNIST`, as in [BGS16], and `FashionMNIST`. They are trained on 60000 images for 32 epochs, using Adam with a mini-batch size of 128. We report the negative log-likelihood (NLL; in nats) achieved by the VAEs on a held-out test set of 10000 images, as estimated via importance sampling using 32 samples per test point. To our knowledge, no other BO paper considered DGM tuning.

VAE tuning is difficult due to the high-dimensional and structured nature of its hyperparameter space, and, in particular, due to constraints arising from dependencies between some of its parameters. We tune 25 discrete parameters defining the model architecture, e.g. the number of convolutional layers, their stride, padding and filter size, the number and width of fully-connected layers, and the latent space dimensionality. We further tune three continuous parameters for the optimizer and regularization. Crucially, mutual dependencies between the discrete parameters result in complex constraints, as certain combinations of stride, padding and filter size lead to invalid architectures.

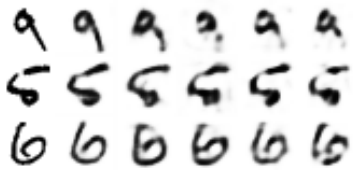


Figure 5.7: Randomly chosen `MNIST` test images (left column) and their reconstructions by the best VAE models found by MiVaBo, random search, GPyOpt, TPE and SMAC (left to right), thus ordered by NLL values, which seem to capture visual appearance quality.

Particularly, for the encoder, the shapes of all layers must be integral, and for the decoder, the output shape must match the input data shape, i.e., one channel of size 28×28 for `{Fashion}MNIST`. The latter constraint is especially challenging, as only a small number of decoder configurations yield the required output shape. Thus, even for rather simple datasets such as `{Fashion}MNIST`, tuning such a VAE is significantly more challenging than, say, tuning a convolutional neural network for classification.

While MiVaBo can conveniently capture these restrictions via linear and quadratic constraints, the competing methods cannot. To enable a comparison that is as fair as possible, we thus use the following sensible heuristic to incorporate the knowledge about the constraints into the baselines: If a method tries to evaluate an invalid parameter configuration, we return a penalty error value, which will discourage a model-based method to sample this (or a similar) setting again. However, for fairness, we only report valid observations and ignore all configurations that violated a constraint. We set the penalty value to 500 nats, which is the error incurred by a uniformly random generator. We investigated the impact of the penalty value (e.g., we also tried 250 and 125 nats) and found that it does *not* qualitatively affect the results.

Figure 5.6 shows that MiVaBo significantly outperforms the competing methods on this task, both on `MNIST` (left) and `FashionMNIST` (right). This is because MiVaBo can naturally encode the constraints and thus directly optimize over the feasible region in parameter space, while TPE, SMAC and GPyOpt need to learn the constraints from data. They fail to do so and get stuck in bad local optima early on. The model-based approaches likely struggle due to sharp discontinuities in hyperparameter space induced by the constraint violation penalties (i.e., as invalid configurations may lie close to well-performing configurations). In contrast, random search is agnostic to these discontinuities, and thus notably outperforms the model-based methods. Lastly, GP-SA and MiVaBo-SA struggle as well, suggesting that while SA can avoid invalid inputs, the effective optimization of complex constrained objectives crucially requires more principled approaches for acquisition function optimization, such as the one we propose. This shows that all model choices for MiVaBo (as discussed in Sec. 5.2.4) are necessary to achieve such strong results. false

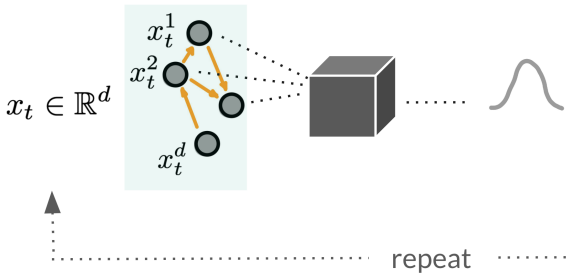
Although log-likelihood scores allow for a quantitative comparison, they are hard to interpret for humans. Thus, for a qualitative comparison, Section 5.3.2 visualizes the reconstruction quality achieved on `MNIST` by the best VAE configuration found by all methods after 32 BO iterations. The VAEs were trained for 32 epochs each, as in Figure 5.6. The likelihoods seem to correlate with the quality of appearance, and the model found by

MiVaBo arguably produces the visually most appealing reconstructions among all models. Note that while MiVaBo requires more time than the baselines (see Table 5.1), this is still negligible compared to the cost of a function evaluation, which involves training a deep generative model. Finally, the best VAE found by MiVaBo achieves 84.25 nats on `MNIST` when trained for 3280 epochs and using 5000 importance samples for log-likelihood estimation, i.e. the setting used in [BGS16] (see Table 5.1). This is comparable to the performance of 82-83 nats achieved by human expert tuned models, e.g. as reported in [SKW15] (which use even more convolutional layers and a more sophisticated inference method), highlighting MiVaBo's effectiveness in tuning complex deep neural network architectures.

5.4 Conclusion

We propose MiVaBo, the first method for efficiently optimizing expensive mixed-variable black-box functions subject to linear and quadratic discrete constraints. MiVaBo combines a linear model of expressive features with Thompson sampling, making it simple yet effective. Moreover, it is highly flexible due to the modularity of its components, i.e., the mixed-variable features, and the optimization oracles for the acquisition procedure. This allows practitioners to tailor MiVaBo to specific objectives, e.g. by incorporating prior knowledge in the feature design or by leveraging optimizers handling specific types of constraints. We show that MiVaBo enjoys theoretical convergence guarantees that competing methods lack. Finally, we empirically demonstrate that MiVaBo significantly improves optimization performance as compared to state-of-the-art methods for mixed-variable optimization on complex hyperparameter tuning tasks.

5.5 Discussion



Following the growing development of Bayesian optimization, a number of recent works have considered ways of extending it into discrete [Bor+21; Oh+19] and mixed domains [OGW21], non-Euclidean domains [Hut+21], and other structured assumptions such as symmetry. Exploiting the structure of the underlying problem helps to improve both the computational efficiency and the quality of the founds solutions. In this work, we first consider all possible interactions between the variables up to the reasonable polynomial order, and scaling to higher orders might not be feasible. In contrast, a known structural (causal) graph allows reasoning about the effective dimensionality of the problem and applying BO in cases where the standard approaches would scale exponentially with

space dimensionality. We consider in [SMK23] a particular case of exploiting structural knowledge in the form of a known causal graph (see Section 5.5).

In the future, it is interesting thus further investigate the challenging extension of *learning the dependency graph* from data. In addition to the dependency, a problem of finding the causal relationships (directed influence of one variable to another) might be a part of the optimization process, e.g., maximizing the crop yield while learning the causal dependence in soil fumigants and insects level. In particular, recent advances in casual global optimization that aim at objective optimization when there is a casual structure over the input variables [Agl+20; Agl+21; SMK23] open a promising direction and applying Bayesian optimization to the problems that are out of the reach for the current approaches.

CHAPTER 6

Conclusion

In this thesis, we proposed and analyzed novel approaches to risk-averse, computationally-effective, and problem-adaptive Bayesian optimization methods and showed their advantages in various applications. In Chapter 3, we proposed a method that takes into account the input-dependent aleatoric uncertainty and learns it together with the objective on the fly of the optimization. The method is suitable for high-stakes applications requiring risk-averse decision-making and enjoys sublinear convergence guarantees. In Chapter 4, we contributed to the computational efficiency of Bayesian optimization from different angles, such as the multi-fidelity approach and automated termination. Motivated by impactful applications, such as the calibration of an agent-based transport system simulation and automated machine learning, the proposed methods show competitive results and provide new insights for future exciting advancements. In Chapter 5, we emphasized the importance of exploiting the structure and interconnections within the decision domain in Bayesian optimization and introduced the first method for mixed-variable optimization. Importantly, the proposed method is both practically viable and theoretically studied. Overall, this dissertation contributes to the field of sequential decision-making by introducing novel Bayesian optimization approaches that address real-world motivated challenges. These methods provide a balance between theoretical soundness and practical applicability, which, we believe, makes them a valuable resource for researchers and practitioners alike.

6.1 Future directions

Finally, we would like to discuss several exciting avenues for future work.

Multi-fidelity learning of the aleatoric uncertainty The aversion to risk clearly comes with an extra evaluation price. As can be seen from the convergence guarantees in Theorem 1 for RaHBO proposed in Chapter 3, the regret bound non-trivially depends on the number of the experiment repetitions k: the larger k increases sample complexity but also leads to better estimation of the noise model. There are several ways to improve this computational complexity, for example, the multi-fidelity approach where the number of the experiment repetitions k is a matter of choice at each evaluation step. Thus, it would be interesting to connect methods developed Section 4.2.2 and Chapter 3 to allow for better practical and theoretical results. Moreover, such multi-fidelity estimation of the

heteroscedastic noise variance might be directly useful for the termination condition in BO from Section 4.1 that would be in the same spirit of the approach under homoscedasticity assumption in Section 5.1.

Risk-averse causal optimization Causal Bayesian optimization (CBO) tackles the problem of optimal intervening on an unknown structural causal model to maximize a downstream variable of interest. CBO has important applications in medicine, ecology, and manufacturing that require decisions in high-stakes settings. Similar to the non-causal BO considered in Chapter 3, two optimal interventions have the same mean value, but different noise levels, i.e., heteroscedastic noise variances. Making risk-averse decisions in such settings is a rather under-explored area, and it would exciting to explore how the advances in causal decision-making can enrich the BO in high-stakes applications, and vice versa. In particular, on the one hand, there are still open questions on how to measure uncertainty in CBO [SMK23]. On the other hand, exploiting the structural dependency within the decision domain can improve the convergence guarantees for such a risk-averse optimization over the ones in Chapter 3. While in a general BO setup, we intervene on all variables breaking the dependency structure in variables, in causal Bayesian optimization [Agl+20] we can use do-calculus that allows predicting the effect of the intervention without its direct evaluation. Thus, under the assumption of a known graph structure, we could rely on the graph to infer aleatoric uncertainty with fewer evaluations.

Bayesian optimization under heavy-tailed noise variance On the one hand, the sub-Gaussian tail assumption on the observation likelihood is quite rich covering many distributions beyond the normal distribution. On the other hand, as we have seen in Section 5.1, the observation likelihood for the noise variance might be heavy-tailed, thus requiring more principled ways of dealing with it.

Appendices

APPENDIX A

Additional Background for Chapter 2

A.1 Preliminaries and definitions

I. Random variables and functions

Definition 3 (Sub-Gaussian)**.** *A zero-mean real-valued random variable* ξ *is* ρ–sub-Gaussian*, if there exists variance-proxy* ρ^2 *such that* $\forall \lambda \in \mathbb{R}, \quad \mathbb{E}[e^{\lambda \xi}] \leq e^{\frac{\lambda^2 \rho^2}{2}}$.

For a sub-Gaussian ξ, its variance $\mathbb{V}ar[\xi]$ lower bounds any valid variance-proxy ρ, i.e., $\mathbb{V}ar[\xi] \leq \rho^2$ (this can be proved using Tailor expansion). In case $\mathbb{V}ar[\xi] = \rho^2$ holds, ξ is said to be *strictly* ρ*–sub-Gaussian*. Besides zero-mean Gaussian random variables, most standard symmetric bounded random variables (e.g., Bernoulli, beta, uniform, binomial) are strictly sub-Gaussian (see [AMN19, Proposition 1.1]). Throughout the paper, we consider sub-Gaussian noise, and in Section 3.3.3, we specialize to the case of strictly sub-Gaussian noise.

II. Optimization problem Throughout this thesis, we consider optimization problems in the form

$$\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}), \tag{A.1}$$

where f is the objective function and $\mathcal{X} \in \mathbb{R}^d$ is some closed convex set. In some chapters, we particularly consider constrained optimization problems in the form of

$$\begin{aligned} &\min_{\boldsymbol{x} \in \mathcal{X}} f(\boldsymbol{x}) \\ \text{s.t.} \quad &f_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \end{aligned} \tag{A.2}$$

with m constraint functions and *s.t.* standing for *subject to*.

III. Lagrangian Duality The *Lagrangian* function $\mathcal{L} : \mathcal{X} \times \mathbb{R}^m \to \mathbb{R}$ for the optimization problem Eq. (A.2) is defined as a weighted sum of the objective and the constraints, i.e.:

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i f_i(\boldsymbol{x}), \tag{A.3}$$

The vector $\lambda \in \mathbb{R}^m$ is a vector of dual variable, also called *Lagrange multipliers*. The *dual function* $g : \mathbb{R}^m \to \mathbb{R}$ is defined as a pointwise minimum of the Lagrangian, i.e.:

$$g(\lambda) = \min_{\boldsymbol{x} \in \mathcal{X}} \mathcal{L}(\boldsymbol{x}, \lambda). \tag{A.4}$$

The *dual problem* is defined as the maximization of the concave dual function, i.e.:

$$\max_{\lambda \in \mathbb{R}^m_+} g(\lambda). \tag{A.5}$$

Let p^* be the optimal value of the primal problem Eq. (A.2). Then the convex problem Eq. (A.5) finds the tightest lower bound for p^*. Let d^* be the optimal value for the dual problem Eq. (A.5). The *weak duality*, i.e, when $d^* \leq p^*$, holds for convex and nonconvex primal problems and can be used to find nontrivial lower bounds when primal is difficult.

IV. Function spaces A *function space* $\mathcal{F}$ is a space whose elements are functions, e.g., $f \colon \mathbb{R}^d \to \mathbb{R}$. Throughout the thesis, we use the following notions important when talking about function spaces:

Definition 4 (Inner product)**.** *Inner product* $\langle \cdot, \cdot \rangle \colon \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ *is a function that for any* $f, g \in \mathcal{F}$ *and* $\alpha \in \mathbb{R}$ *satisfies: (1) symmetry* $\langle f, g \rangle = \langle g, f \rangle$*, (2) linearity* $\langle \alpha f, g \rangle = \alpha \langle f, g \rangle$*, and (3) pd* $\langle f, f \rangle \geq 0 \forall f$ *and* $\langle f, f \rangle = 0$ *iff* $f = 0$*.*

Definition 5 (Norm)**.** *Norm* $\| \cdot \| \colon \mathcal{F} \to \mathbb{R}$ *is a function that for all* $f, g \in \mathcal{F}$ *and* $\alpha \in \mathbb{R}$ *satisfies: (1) positivity* $\|f\| \geq 0$ *and* $\|f\| = 0$ *iff* $f = 0$*, (2) triangle inequality* $\|f + g\| \leq \|f\| + \|g\|$*, (3) linearity* $\|\alpha f\| = \alpha \|f\|$*.*

Definition 6 (Hilbert space)**.** *A Hilbert space* $\mathcal{H}$ *is a complete, (possibly) infinite-dimensional linear space endowed with an inner product.*

Definition 7 (Reproducing kernel)**.** *A function* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is a* reproducing kernel *of a Hilbert space* $\mathcal{H}$ *if* $\forall f \in \mathcal{H}, f(\boldsymbol{x}) = \langle \kappa(\boldsymbol{x}, \cdot, f(\cdot) \rangle$*.*

Definition 8 (Positive semidefinite matrix)**.** *The matrix* M *is called* positive-semidefinite *if* M *is symmetric and for any vector* $x \in \mathcal{X}$ *the following is satisfied:*

$$x^\top M x \geq 0. \tag{A.6}$$

Definition 9 (Positive semidefinite function)**.** *Function* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is called* positive semidefinite function *if for any* $x_1, \ldots, x_n \in \mathcal{X}$ *and any* $a_1, \ldots, a_n \in \mathbb{R}$ *the following is satisfied:*

$$\sum_{i=1}^{n} \sum_{i=1}^{n} a_i a_j \kappa(x_i, x_j) \geq 0 \tag{A.7}$$

There are two equivalent ways to define a reproducing kernel Hilbert space. The first one is to define it as a Hilbert space of functions with all evaluation functionals bounded and linear. Though this highlights continuity requirements on the pointwise evaluation functionals, it might be unclear what a reproducing kernel Hilbert space *induced by a*

kernel κ means. We, therefore, introduce another way to define it as a Hilbert space $\mathcal{H}$ with a reproducing kernel whose span is dense in $\mathcal{H}$.

Definition 10 (RKHS)**.** *Let* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *be a positive semi-definite kernel function and* $\mathcal{X}$ *is a compact set. The* reproducing kernel Hilbert space $\mathcal{H}_k$ *is the linear space defined as the span*

$$\mathcal{H}_\kappa = \left\{ f(\cdot) \triangleq \sum_{i=1}^{t} \alpha_i \kappa(x_i, \cdot) : \quad t \in \mathcal{N}, \alpha_i \in \mathbb{R}, \boldsymbol{x}_i \in \mathcal{X} \right\}, \tag{A.8}$$

with the inner product $\langle f, g \rangle = \sum_i \sum_j \alpha_i \beta_j \kappa(u_i, v_j)$ *for* $f, g \in \mathcal{H}_\kappa$ *be* $f = \sum_i \alpha_i \kappa(u_i, \cdot)$ *and* $g = \sum_i \beta_i \kappa(v_i, \cdot)$.

A.2 Recap on Hilbert spaces

Hilbert spaces. Let $\mathcal{H}$ be some separable Hilbert space, i.e., space with a countable orthonormal basis. Hilbert spaces allow us to apply finite-dimensional linear algebra for infinite-dimensional spaces of functions, e.g., completeness of the space guarantees convergence of some algorithms. Assuming only Hilbert spaces is however not enough to reason about the predictions of an unknown function because of whole domain. As an example, consider a space of square-integrable functions $L_2[a, b]$ on an interval $\mathcal{X} = [a, b]$ with the inner product defines as $\langle f, g \rangle = \int_a^b f(x)g(x)dx$ and norm induced by that inner product. Then for any function f we can come up with a function g that would differ from f at any finite number of points by an arbitrary real value but the norm would coincide $\|f - g\| = 0$ (since it is an integral). Reproducing kernel Hilbert spaces (RKHS) define a family of spaces that does not have such a problem.

Formal introduction of RKHS. An RKHS $\mathcal{H}$ is a Hilbert space with evaluation functional $F_{\boldsymbol{x}}[f] = f(\boldsymbol{x}), \quad F_{\boldsymbol{x}} \colon \mathcal{H} \to \mathbb{R}$, being bounded, i.e., there exists $M \neq M(x)$ such that $|F_{\boldsymbol{x}}[f]| = |f(\boldsymbol{x})| \leq M\|f\|_{\mathcal{H}}$. This condition allows to evaluate the function of interest f over the whole domain, i.e., $\forall \boldsymbol{x} \in \mathcal{X}$. Notably, this condition does not hold for the square-integrable functions $L_2(\mathcal{X})$ where functions can get arbitrary high values on some finite set of inputs $\boldsymbol{x}$.

RKHS and kernels. In RKHS $\mathcal{H}$, for any $\boldsymbol{x} \in \mathcal{X}$, there exists a function $\kappa_{\boldsymbol{x}} \in \mathcal{H}$ (the representer of $\mathcal{H}$) such that evaluating any $f \in \mathcal{H}$ at $\boldsymbol{x}$ corresponds to the inner product with the representer, i.e., the evaluation functional $F_{\boldsymbol{x}}[f] = \langle \kappa_{\boldsymbol{x}}, f \rangle = f(\boldsymbol{x})$. Thus an RKHS defines a reproducing kernel, i.e. symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $\kappa_{\boldsymbol{x}}(\boldsymbol{x}') = \langle \kappa_{\boldsymbol{x}}, \kappa_{\boldsymbol{x}'} \rangle = \kappa_{\boldsymbol{x}'}(\boldsymbol{x})$, and vice versa: a reproducing kernel defines a unique RKHS. For most kernel used in the thesis, any continuous function on any compact $\mathcal{X}$ can be approximated by functions from $\mathcal{H}_\kappa(\mathcal{X})$. RKHS $\mathcal{H}_\kappa(\mathcal{X})$ is a complete subspace of $L_2(\mathcal{X})$ obeying the reproducing property.

APPENDIX B

Proofs of Chapter 3

B.1 Details on Proposition 1

We first provide the proof of Proposition 1 for cumulative risk-averse regret Eq. (3.3) with known variance-proxy $\rho^2(\cdot)$ (see Definition 3) (Section B.1.1). We further provide data-independent bounds for β_T (Section B.1.2) and maximum information gain γ_T (Section B.1.3) that together conclude the proof for sub-linear on T regret guarantees for most of the popularly used kernels.

B.1.1 Proof Proposition 1

Proposition 1. *Consider any $f \in \mathcal{H}_\kappa$ with $\|f\|_\kappa \leq \mathcal{B}_f$ and sampling model from Eq. (3.1) with known variance-proxy $\rho^2(\boldsymbol{x})$. Let $\{\beta_t\}_{t=1}^T$ be set as in Lemma 3 with $\lambda = 1$. Then, with probability at least $1-\delta$,* RAHBO *attains cumulative risk-averse regret $R_T = \mathcal{O}\big(\beta_T\sqrt{T\gamma_T(\bar{\varrho}^2+1)}\big)$.*

Proof. The main steps of the proof are as follows: In *Step 1*, we derive the upper and the lower confidence bounds, $\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)$ and $\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)$, on $\mathrm{MV}(\boldsymbol{x}_t)$ at iteration t. In *Step 2*, we bound the instantaneous risk-averse regret $r(\boldsymbol{x}_t) := \mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_t)$. In *Step 3*, we derive mutual information $I(y_{1:T}, f_{1:T})$ in case of the heteroscedastic noise. In *Step 4*, we bound the sum of variances $\sum_{t=1}^T \sigma_{t-1}(\boldsymbol{x}_t)$ via mutual information $I(y_{1:T}, f_{1:T})$. In *Step 5*, we bound the cumulative regret $R_T = \sum_{t=1}^T r(\boldsymbol{x}_t)$ based on the previous steps.

Step 1: *On the confidence bounds for MV($\boldsymbol{x}$).*
In case of known variance-proxy $\rho^2(\boldsymbol{x})$, the confidence bounds for $\mathrm{MV}(\boldsymbol{x})$ at iteration t can be directly obtained based on the posterior $\mu_t(\boldsymbol{x})$ and $\sigma_t(\boldsymbol{x})$ for $f(\boldsymbol{x})$ defined in (3.5) and (3.6). Particularly, for $\beta_t = \beta_t(\delta)$ defined in Eq. (3.8), $\Pr\big\{\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}) \leq \mathrm{MV}(\boldsymbol{x}) \leq \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x})\ \forall x \in \mathcal{X}, \forall t \geq 0\big\} \geq 1-\delta$ with the confidence bounds:

$$\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}) := \mu_{t-1}(\boldsymbol{x}) - \beta_t\sigma_{t-1}(\boldsymbol{x}) - \alpha\rho^2(\boldsymbol{x}), \tag{B.1}$$
$$\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}) := \mu_{t-1}(\boldsymbol{x}) + \beta_t\sigma_{t-1}(\boldsymbol{x}) - \alpha\rho^2(\boldsymbol{x}). \tag{B.2}$$

Step 2: *On bounding the instantaneous risk-averse regret $r_t(\boldsymbol{x})$.* We have

$$r(\boldsymbol{x}_t) = \mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_t)$$

$$\leq \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}^*) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)$$
$$\leq \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}_t) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t) = 2\beta_t\sigma_{t-1}(\boldsymbol{x}_t),$$

where the first inequality is due to the definition of confidence bounds, the second is due to the acquisition strategy $x_t \in \arg\max_{x\in\mathcal{X}} \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x})$; and the equality further expands $\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x})$ and $\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x})$. Thus, the cumulative regret can be bounded as follows:

$$R_T = \sum_{t=1}^{T} r(\boldsymbol{x}_t) \leq \sum_{t=1}^{T} 2\beta_t\sigma_{t-1}(\boldsymbol{x}_t) \leq 2\beta_T \sum_{t=1}^{T} \sigma_{t-1}(\boldsymbol{x}_t), \tag{B.3}$$

where the last inequality holds since $\{\beta_t\}_{t=1}^T$ is a non-decreasing sequence.

Step 3: *On mutual information $I(y_{1:T}, f_{1:T})$ and maximum information gain γ_T.* Mutual information $I(y_{1:T}, f_{1:T})$ between the vector of evaluations $y_{1:T} \in \mathbb{R}^T$ at points $A = \{x_t\}_{t=1}^T$ and $f_{1:T} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_T)]^\top$ is defined by

$$I(y_{1:T}, f_{1:T}) = H(y_{1:T}) - H(y_{1:T}|f_{1:T}),$$

where $H(\cdot)$ denotes entropy. Under the modelling assumptions $f_{1:T} \sim \mathcal{N}(0, \lambda^{-1}K_T)$ and $\xi_{1:T} \sim \mathcal{N}(0, \Sigma_T)$ for the noise $\xi_{1:T} = [\xi(\boldsymbol{x}_1), \ldots, \xi(\boldsymbol{x}_T)]^\top$, the measurements are distributed as $y_{1:T} \sim \mathcal{N}(0, \lambda^{-1}K_T + \Sigma_T)$ and $y_t|y_{1:t-1} \sim \mathcal{N}(\mu_{t-1}(\boldsymbol{x}_t), \rho^2(\boldsymbol{x}_t) + \sigma_{t-1}^2(\boldsymbol{x}_t))$, where $\sigma_{t-1}^2(\cdot)$ is defined in Eq. (3.6). Hence, the entropy of each new measurement y_t conditioned on the previous history $y_{1:t-1}$ is:

$$\begin{aligned}
H(y_t|y_{1:t-1}) &= \frac{1}{2}\ln\left(2\pi e\left(\rho^2(\boldsymbol{x}_t) + \sigma_{t-1}^2(\boldsymbol{x}_t)\right)\right) \\
&= \frac{1}{2}\ln\left(2\pi e\rho^2(\boldsymbol{x}_t)\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right)\right) \\
&= \frac{1}{2}\ln\left(2\pi e\rho^2(\boldsymbol{x}_t)\right) + \frac{1}{2}\ln\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right), \\
H(y_{1:T}) &= \sum_{t=1}^{T} H(y_t|y_{1:t-1}) = \frac{1}{2}\sum_{t=1}^{T}\ln\left(2\pi e\rho^2(\boldsymbol{x}_t)\right) + \frac{1}{2}\sum_{t=1}^{T}\ln\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right), \\
H(y_{1:T}|f_{1:T}) &= \sum_{t=1}^{T} H(y_t|f_t) = \frac{1}{2}\sum_{t=1}^{T}\ln(2\pi e\rho^2(\boldsymbol{x}_t)).
\end{aligned}$$

Therefore, the information gain for $y_{1:T}$ is:

$$I(y_{1:T}, f_{1:T}) = H(y_{1:T}) - H(y_{1:T}|f_{1:T}) = \frac{1}{2}\sum_{t=1}^{T}\ln\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right). \tag{B.4}$$

Then, by definition of maximum information gain:

$$\gamma_T := \max_{A\subset\mathcal{X},\,|A|=T} I(y_{1:T}, f_{1:T}) \geq \frac{1}{2}\sum_{t=1}^{T}\ln\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right). \tag{B.5}$$

Step 4: *On bounding* $\sum_{t=1}^{T}\sigma_{t-1}(\boldsymbol{x}_t)$.

$$\begin{aligned}\sum_{t=1}^{T}\sigma_{t-1}(\boldsymbol{x}_t) &= \sum_{t=1}^{T}\frac{\rho(\boldsymbol{x}_t)}{\rho(\boldsymbol{x}_t)}\sigma_{t-1}(\boldsymbol{x}_t) \leq \sqrt{T\sum_{t=1}^{T}\rho^2(\boldsymbol{x}_t)\frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}} \\ &\leq \sqrt{T\sum_{t=1}^{T}\frac{1}{\ln(1+\rho^{-2}(\boldsymbol{x}_t))}\ln\left(1+\frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right)} \\ &\leq \sqrt{\frac{2T}{\ln(1+\bar{\varrho}^{-2})}\underbrace{\frac{1}{2}\sum_{t=1}^{T}\ln\left(1+\frac{\sigma_{t-1}^2(\boldsymbol{x}_t)}{\rho^2(\boldsymbol{x}_t)}\right)}_{\text{mutual information Eq. (B.4)}}},\end{aligned} \tag{B.6}$$

where the first inequality follows from the Cauchy-Schwarz inequality. The second one is due to the fact that for any $s^2 \in [0, \rho^{-2}(\boldsymbol{x}_t)]$ we can bound $s^2 \leq \frac{\rho^{-2}(\boldsymbol{x}_t)}{\ln(1+\rho^{-2}(\boldsymbol{x}_t))}\ln(1+s^2)$, that also holds for $s^2 := \rho^{-2}(\boldsymbol{x}_t)\sigma_{t-1}^2(\boldsymbol{x}_t)$ since $\rho^{-2}(\boldsymbol{x}_t)\sigma_{t-1}^2(\boldsymbol{x}_t) \leq \rho^{-2}(\boldsymbol{x}_t)\lambda^{-1}\kappa(\boldsymbol{x}_t, x_t) \leq \rho^{-2}(\boldsymbol{x}_t)$ for $\lambda \geq 1$. The third inequality is due to $\rho(\boldsymbol{x}) \in [\underline{\varrho}, \bar{\varrho}]$.

Step 5: *Bounding risk-averse cumulative regret* $R_T = \sum_{t=1}^{T} r(\boldsymbol{x}_t)$.
Combining the previous three steps together: Eq. (B.3), Eq. (B.5), and Eq. (B.6) we finally obtain:

$$R_T \leq \sum_{t=1}^{T} 2\beta_t\sigma_{t-1}(\boldsymbol{x}_t) \leq 2\beta_T\sum_{t=1}^{T}\sigma_{t-1}(\boldsymbol{x}_t) \leq 2\beta_T\sqrt{\frac{2T}{\ln(1+\bar{\varrho}^{-2})}\gamma_T}$$

Also, note that for any $\alpha \geq 0$ the bound $\ln(1+\alpha) \geq \frac{\alpha}{1+\alpha}$ holds, thus $\frac{1}{\ln(1+\bar{\varrho}^{-2})} \leq \frac{1+\bar{\varrho}^{-2}}{\bar{\varrho}^{-2}} = \bar{\varrho}^2 + 1$. Therefore, the cumulative regret can be also bounded as $R_T = O(\beta_T\sqrt{T\gamma_T(\bar{\varrho}^2+1)})$. ■

B.1.2 Bounds for β_T

We provide the bounds for the data-dependent β_T that appear in the regret bound (see Eq. (3.8)). Following our modelling assumptions $f_{1:T} \sim \mathcal{N}(0, \lambda^{-1}K_T)$ and $\xi_{1:T} \sim \mathcal{N}(0, \Sigma_T)$, the information gain $I(y_{1:T}, f_{1:T}) = H(y_{1:T}) - H(y_{1:T}|f_{1:T})$ is given as follows:

$$I(y_{1:T}, f_{1:T}) = \underbrace{\frac{1}{2}\ln\left(\det(2\pi e(\lambda^{-1}K_T + \Sigma_T))\right)}_{H(y_{1:T})} - \underbrace{\frac{1}{2}\ln\left(\det(2\pi e\Sigma_T)\right)}_{H(y_{1:T}|f_{1:T})} = \frac{1}{2}\ln\left(\frac{\det(K_T+\lambda\Sigma_T)}{\det(\lambda\Sigma_T)}\right). \tag{B.7}$$

By definition then $\gamma_T = \max_{A\subset\mathcal{X},\,|A|=T} I(y_{1:T}, f_{1:T}) \geq \frac{1}{2}\ln\left(\frac{\det(K_T+\lambda\Sigma_T)}{\det(\lambda\Sigma_T)}\right)$. On the other hand, β_T defined in Lemma 1 can be expanded in a data-independent manner as follows:

$$\beta_T := \sqrt{2\ln\left(\frac{\det(\lambda\Sigma_T + K_T)^{1/2}}{\delta\det(\lambda\Sigma_T)^{1/2}}\right)} + \sqrt{\lambda}\|f\|_\kappa$$

$$= \sqrt{2\ln\frac{1}{\delta} + \ln\left(\frac{\det(\lambda\Sigma_T + K_T)}{\det(\lambda\Sigma_T)}\right)} + \sqrt{\lambda}\|f\|_\kappa \leq \sqrt{2\ln\frac{1}{\delta} + \gamma_T} + \sqrt{\lambda}\mathcal{B}_f. \tag{B.8}$$

B.1.3 Bounds for γ_T

Here, we show the relation between the information gains under heteroscedastic and homoscedastic noise. Note that for the latter the upper bounds are widely known, e.g., [Sri+10]. To distinguish between the maximum information gain for heteroscedastic noise with variance-proxy $\rho^2(\boldsymbol{x})$ and the maximum information gain for homoscedastic noise with fixed variance-proxy σ^2, we denote them as $\gamma_T^{\rho_x}$ and γ_T^{σ} respectively. Recall that $\varrho^2(\cdot) \in [\underline{\varrho}^2, \bar{\varrho}^2]$ for some constant values $\bar{\varrho}^2 \geq \underline{\varrho}^2 > 0$.

Below, we show that $\gamma_T^{\rho_x} \leq \gamma_T^{\sigma}\frac{\bar{\varrho}^2}{\underline{\varrho}^2}$ with σ^2 set to $\bar{\varrho}^2$, that only affects the constants but not the main scaling (in terms of T) of the known bound for the homoscedastic maximum information gain.

$$\gamma_T^{\rho_x} \overset{①}{=} \max_{A\subset\mathcal{X},|A|=T} \frac{1}{2}\sum_{t=1}^{T} \ln\left(1 + \frac{\sigma^2_{t-1}(\boldsymbol{x}_t|\rho^2(\boldsymbol{x}_t))}{\rho^2(\boldsymbol{x}_t)}\right) \overset{②}{\leq} \max_{A\subset\mathcal{X},|A|=T} \frac{1}{2}\sum_{t=1}^{T} \ln\left(1 + \frac{\sigma^2_{t-1}(\boldsymbol{x}_t|\bar{\varrho}^2)}{\underline{\varrho}^2}\right) \tag{B.9}$$

$$\overset{③}{=} \max_{A\subset\mathcal{X},|A|=T} \frac{1}{2}\sum_{t=1}^{T} \ln\left(1 + \frac{\bar{\varrho}^2}{\underline{\varrho}^2}\frac{\sigma^2_{t-1}(\boldsymbol{x}_t|\bar{\varrho}^2)}{\bar{\varrho}^2}\right) \overset{④}{\leq} \max_{A\subset\mathcal{X},|A|=T} \frac{1}{2}\sum_{t=1}^{T} \frac{\bar{\varrho}^2}{\underline{\varrho}^2}\ln\left(1 + \frac{\sigma^2_{t-1}(\boldsymbol{x}_t|\bar{\varrho}^2)}{\bar{\varrho}^2}\right) \tag{B.10}$$

$$\overset{⑤}{=} \max_{A\subset\mathcal{X},|A|=T} \frac{\bar{\varrho}^2}{\underline{\varrho}^2}\frac{1}{2}\sum_{t=1}^{T} \ln\left(1 + \frac{\sigma^2_{t-1}(\boldsymbol{x}_t|\sigma^2)}{\sigma^2}\right) = \frac{\bar{\varrho}^2}{\underline{\varrho}^2}\gamma_T^{\sigma}, \tag{B.11}$$

where ① follows from Eq. (B.4). In ②, we lower bound the denominator $\rho^2(\boldsymbol{x}_t)$ and upper bound the numerator $\sigma^2_{t-1}(\boldsymbol{x}_t)$ (due to monotonicity w.r.t. noise variance, i.e., $\sigma^2_{t-1}(\boldsymbol{x}_t|\Sigma_t) \leq \sigma^2_{t-1}(\boldsymbol{x}_t|\bar{\varrho}^2\mathbf{I}_t)$). In ③, we multiply by $1 = \bar{\varrho}^2/\bar{\varrho}^2$. In ④ we use Bernoulli inequality since $\bar{\varrho}^2/\underline{\varrho}^2 \geq 1$. The obtained expression can be interpreted as a standard information gain for homoscedastic noise and, particularly, with the variance-proxy σ^2 set to $\bar{\varrho}^2$ due to ⑤. Finally, the upper bounds on γ_T^{σ} typically scale sublinearly in T for most of the popularly used kernels [Sri+10], e.g, for linear kernel $\gamma_T = \mathcal{O}(d\log T)$, and for squared exponential kernel $\gamma_T = \mathcal{O}(d(\log T)^{d+1})$.

$$\gamma_T = \mathcal{O}(d(\log T)^{d+1})$$

B.2 Tighter bounds for the variance-proxy.

Assumption 2 states that noise $\eta(\boldsymbol{x})$ from Eq. (3.13) is $\rho^2_\eta(\boldsymbol{x})$-sub-Gaussian with variance-proxy $\rho^2_\eta(\boldsymbol{x})$ being known. In practice, $\rho^2_\eta(\boldsymbol{x})$ might be unknown.

Here, we describe a way to estimate $\rho^2_\eta(\boldsymbol{x})$ under the following two assumptions: the evaluation noise $\xi(\boldsymbol{x})$ is strictly sub-Gaussian (that is already reflected in the Assumption 1)

and the noise $\eta(\boldsymbol{x})$ of variance evaluation is also strictly sub-Gaussian, that is, $\mathbb{V}ar[\eta(\boldsymbol{x})] = \rho_\eta^2(\boldsymbol{x})$ and $\mathbb{V}ar[\xi(\boldsymbol{x})] = \rho^2(\boldsymbol{x})$.

(i) Reformulation of the sample variance. We first rewrite the sample variance defined in Eq. (5.2) as the average over squared differences over all pairs $\{y_1(\boldsymbol{x}), \dots, y_k(\boldsymbol{x})\}$:

$$\hat{s}_k^2(x) \overset{①}{=} \frac{1}{2k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} (y_i(\boldsymbol{x}) - y_j(\boldsymbol{x}))^2 \overset{②}{=} \frac{1}{2k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} (\xi_i(\boldsymbol{x}) - \xi_j(\boldsymbol{x}))^2, \quad \text{(B.12)}$$

where ② is due to $y_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \xi_i(\boldsymbol{x})$, and ① is equivalent to the Eq. (5.2) due to the following:

$$\frac{1}{2k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} (y_i - y_j)^2 = \frac{1}{2k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} (y_i - \hat{m}_k + \hat{m}_k - y_j)^2 \quad \text{(B.13)}$$

$$= \frac{1}{2k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} \left[\left(y_i - \hat{m}_k\right)^2 + \left(y_j - \hat{m}_k\right)^2 - 2\left(y_i - \hat{m}_k\right)\left(y_j - \hat{m}_k\right) \right]$$

$$= \frac{1}{2k(k-1)} \sum_{i=1}^{k} \sum_{j=1}^{k} \left[\left(y_i - \hat{m}_k\right)^2 + \left(y_j - \hat{m}_k\right)^2 \right]$$

$$= \frac{1}{k-1} \sum_{i=1}^{k} (y_i - \hat{m}_k)^2. \quad \text{(B.14)}$$

(ii) Variance of the sample variance $\mathbb{V}ar[\hat{s}_k^2(x)]$. In Eq. (B.12), we show that sample variance can be written in terms of the noise $\xi(\boldsymbol{x})$. In [Ben18] (see Eq. (37)), it is shown that for i.i.d observations $\{\xi_1(\boldsymbol{x}), \dots, \xi_k(\boldsymbol{x})\}$, sampled from a distribution with the 2nd and 4th central moments $\mathbb{V}ar[\xi(\boldsymbol{x})]$ and $\mu_4(\boldsymbol{x}) = \mathbb{E}[\xi^4(\boldsymbol{x})]$, respectively, the variance of the sample variance can be computed as follows:

$$\mathbb{V}ar[\hat{s}_k^2(x)] = \mathbb{E}[\left(\hat{s}_k^2(x)\right)^2] - \mathbb{E}[\hat{s}_k^2(x)]^2 = \frac{\mu_4(\boldsymbol{x})}{k} - \frac{(k-3)\mathbb{V}ar^2[\xi(\boldsymbol{x})]}{k(k-1)}.$$

Since $\xi(\boldsymbol{x})$ is strictly $\rho(\boldsymbol{x})$–sub-Gaussian, the latter can be further adapted as

$$\mathbb{V}ar[\hat{s}_k^2(x)] = \frac{\mu_4(\boldsymbol{x})}{k} - \frac{(k-3)\rho^4(\boldsymbol{x})}{k(k-1)}.$$

(iii) Due to $\eta(\boldsymbol{x})$ being strictly sub-Gaussian, i.e., $\rho_\eta^2(\boldsymbol{x}) = \mathbb{V}ar[\eta(\boldsymbol{x})] = \mathbb{V}ar[\hat{s}_k^2(x)]$, the derivation above also holds for the variance-proxy $\rho_\eta^2(\boldsymbol{x})$:

$$\rho_\eta^2(\boldsymbol{x}) = \frac{\mu_4(\boldsymbol{x})}{k} - \frac{(k-3)\rho^4(\boldsymbol{x})}{k(k-1)}.$$

(iv) Bound 4th moment $\mu_4(\boldsymbol{x})$. The 4th moment $\mu_4(\boldsymbol{x})$ can expressed in terms of the distribution kurtosis that is bounded under our assumptions. Particularly, *kurtosis* $\text{Kurt}[\xi] := \frac{\mathbb{E}[(\xi - \mathbb{E}[\xi])^4]}{\mathbb{V}ar^2(\xi)}$ is measure that identifies the tails behaviour of the distribution of ξ;

$\mathrm{Kurt}(\xi) = 3$ for normallly distribute ξ and $\mathrm{Kurt}(\xi) \leq 3$ for strictly sub-Gaussian random variable ξ (see ([AMN19])). This implies

$$\mu_4(\boldsymbol{x}) = \mathrm{Kurt}\big(\xi(\boldsymbol{x})\big)\rho^4(\boldsymbol{x}) \leq 3\rho^4(\boldsymbol{x}).$$

(v) Bound variance-proxy. There

$$\rho_\eta^2(\boldsymbol{x}) \leq \frac{3(k-1)\rho^4(\boldsymbol{x}) - (k-3)\rho^4(\boldsymbol{x})}{k(k-1)} = \frac{3k-3-k+3}{k(k-1)}\rho^4(\boldsymbol{x}) = \frac{2\rho^4(\boldsymbol{x})}{k-1}.$$

In case of the known bound $\bar{\varrho}^2 \geq \rho^2(\boldsymbol{x})$, we bound the unknown $\rho_\eta^2(\boldsymbol{x})$ as follows:

$$\rho_\eta^2(\boldsymbol{x}) \leq \frac{2\bar{\varrho}^4}{k-1}.$$

Then, we can show that $\rho_\eta^2(\boldsymbol{x}) = \frac{\rho(\boldsymbol{x})^4 * \mathrm{Kurt}(\xi(\mathrm{x}))}{k} - \frac{\rho^4(\boldsymbol{x})(k-3)}{k(k-1)} \leq \frac{3(k-1)\rho^4(\boldsymbol{x}) - \rho^4(\boldsymbol{x})(k-3)}{k(k-1)} = \rho^4(\boldsymbol{x})\frac{3k-3-k+3}{k(k-1)} = \frac{2\rho^4(\boldsymbol{x})}{k-1}$.

B.3 Method details: GP-estimator of variance-proxy ρ^2

According to the Assumption 2, variance-proxy $\rho^2 \in \mathcal{H}_{\kappa^{var}}$ is smooth, and $\eta(\boldsymbol{x}) = \hat{s}_k^2(\boldsymbol{x}) - \rho^2(\boldsymbol{x})$ is $\rho_\eta(\boldsymbol{x})$-sub-Gaussian with known variance-proxy $\rho_\eta^2(\boldsymbol{x})$. In this case, confidence bounds for $\rho^2(\boldsymbol{x})$ follow the ones derived in Lemma 3 with β_t^{var} based on Σ_t^{var}. Particularly, we collect noise variance evaluations $\{x_t, \hat{s}_k(\boldsymbol{x}_t)\}_{t=0}^T$. Then the estimates for $\mu_T^{var}(\boldsymbol{x})$ and $\sigma_T^{var}(\boldsymbol{x})$ for ρ^2 follow the corresponding estimates for $f(\boldsymbol{x})$. Particularly,

$$\mu_t^{var}(\boldsymbol{x}) = \kappa_t^{var}(\boldsymbol{x})^T(K_t^{var} + \lambda\Sigma_t^{var})^{-1}\hat{s}_{1:t}, \tag{B.15}$$

$$\sigma_t^{var}(\boldsymbol{x})^2 = \frac{1}{\lambda}(\kappa^{var}(\boldsymbol{x}, x) - \kappa_t^{var}(\boldsymbol{x})^\top(K_t^{var} + \lambda\Sigma_t^{var})^{-1}\kappa_t^{var}(\boldsymbol{x})), \tag{B.16}$$

where $\Sigma_t^{var} = \mathrm{diag}[\rho_\eta^2(\boldsymbol{x}_1), \ldots, \rho_\eta^2(\boldsymbol{x}_t)]$, $\kappa_t^{var}(\boldsymbol{x}) = [\kappa^{var}(\boldsymbol{x}_1, x), \ldots, \kappa^{var}(\boldsymbol{x}_t, x)]^T$ and $(K_t^{var})_{i,j} = \kappa^{var}(\boldsymbol{x}_i, x_j)$. The confidence bounds are then:

$$\mathrm{ucb}_t^{var}(\boldsymbol{x}) = \mu_{t-1}^{var}(\boldsymbol{x}) + \beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x})$$
$$\mathrm{lcb}_t^{var}(\boldsymbol{x}) = \mu_{t-1}^{var}(\boldsymbol{x}) - \beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}),$$

with $\{\beta_t^{var}\}_{t=1}^T$ set according to Lemma 3.

B.4 Proof of Theorem 1

Theorem 1. *Consider any $f \in \mathcal{H}_\kappa$ with $\|f\|_\kappa \leq \mathcal{B}_f$ and sampling model in Eq. (3.1) with unknown variance-proxy $\rho^2(\boldsymbol{x})$ that satisfies Assumptions 1 and 2. Let $\{x_t\}_{t=1}^T$ denote the set of actions chosen by* RAHBO *(Algorithm 2) over T rounds. Set $\{\beta_t\}_{t=1}^T$ and $\{\beta_t^{var}\}_{t=1}^T$ according to Lemma 3 with $\lambda = 1$, $\mathcal{R}^2 = \max_{x\in\mathcal{X}} \rho_\eta^2(\boldsymbol{x}_t)$ and $\rho(\cdot) \in [\underline{\varrho}, \bar{\varrho}]$. Then,*

the risk-averse cumulative regret R_T of RAHBO *is bounded as follows:*

$$\Pr\left\{R_T \leq \beta_T k\sqrt{\frac{2T\hat{\gamma}_T}{\ln(1+k/\bar{\varrho}^2)}} + \alpha\beta_T^{var}k\sqrt{\frac{2T\Gamma_T}{\ln(1+\mathcal{R}^{-2})}}, \quad \forall T \geq 1\right\} \geq 1-\delta. \tag{B.17}$$

Proof. The main steps of our proof are as follows: In *Step 1*, we derive the upper and the lower confidence bounds, $\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)$ and $\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)$, on $\mathrm{MV}(\boldsymbol{x}_t)$ at iteration t. In *Step 2*, we bound the instantaneous risk-averse regret $r(\boldsymbol{x}_t) := \mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_t)$. In *Step 3*, we derive mutual information both for function and variance-proxy evaluations. In *Step 4*, we bound the sum of variances via mutual information. In *Step 5*, we bound the cumulative regret $R_T = \sum_{t=1}^T r(\boldsymbol{x}_t)$ based on the previous steps.

Step 1: *On confidence bounds for MV($\boldsymbol{x}$).*

(i) On confidence bounds for $\rho^2(\boldsymbol{x})$. According to Eq. (B.16), with probability $1-\delta$ the following confidence bounds hold with $\{\beta_t^{var}\}_{t=1}^T$ set according to Lemma 3:

$$\begin{aligned}\mathrm{ucb}_t^{var}(\boldsymbol{x}) &= \mu_{t-1}^{var}(\boldsymbol{x}) + \beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}),\\ \mathrm{lcb}_t^{var}(\boldsymbol{x}) &= \mu_{t-1}^{var}(\boldsymbol{x}) - \beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}).\end{aligned}$$

(ii) On confidence bounds for $f(\boldsymbol{x})$. Here we adapt confidence bounds introduced in Eq. (B.1)-(B.2) since Eq. (3.5) relies on the unknown variance-proxy $\rho^2(\boldsymbol{x})$ incorporated into Σ_T. Conditioning on the event that $\rho^2(\boldsymbol{x})$ is upper bounded by $\mathrm{ucb}_t^{var}(\boldsymbol{x}) \geq \rho(\boldsymbol{x})^2$ defined in (i), the confidence bounds for f with probability $1-\delta$ are:

$$\mathrm{ucb}_t^{f}(\boldsymbol{x}) = \mu_{t-1}(\boldsymbol{x}|\hat{\Sigma}_{t-1}) + \beta_t\sigma_{t-1}(\boldsymbol{x}|\hat{\Sigma}_{t-1}), \tag{B.18}$$

$$\mathrm{lcb}_t^{f}(\boldsymbol{x}) = \mu_{t-1}(\boldsymbol{x}|\hat{\Sigma}_{t-1}) - \beta_t\sigma_{t-1}(\boldsymbol{x}|\hat{\Sigma}_{t-1}), \ \forall x, t. \tag{B.19}$$

(iii) On confidence bounds for MV($\boldsymbol{x}$). Finally, combining (i) and (ii) and using the union bound, with probability $1-2\delta$, we get $\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}) \leq \mathrm{MV}(\boldsymbol{x}) \leq \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x})$ with

$$\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}) = \mathrm{ucb}_t^{f}(\boldsymbol{x}) - \alpha\mathrm{lcb}_t^{var}(\boldsymbol{x}), \tag{B.20}$$

$$\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}) = \mathrm{lcb}_t^{f}(\boldsymbol{x}) - \alpha\mathrm{ucb}_t^{var}(\boldsymbol{x}), \ \forall x, t. \tag{B.21}$$

Step 2: *On bounding the instantaneous regret.*
First, we bound instantaneous regret of a single measurement at point x_t, but with unknown variance-proxy $\rho^2(\boldsymbol{x})$ as follows:

$$\begin{aligned}r_t := \mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_t) &\leq \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}^*) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)\\ &\leq \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}_t) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)\\ &= \mathrm{ucb}_t^{f}(\boldsymbol{x}_t) - \mathrm{lcb}_t^{f}(\boldsymbol{x}_t) + \alpha(\mathrm{ucb}_t^{var}(\boldsymbol{x}_t) - \mathrm{lcb}_t^{var}(\boldsymbol{x}_t))\\ &= 2\beta_t\sigma_{t-1}(\boldsymbol{x}_t|\hat{\Sigma}_{t-1}) + 2\alpha\beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}_t).\end{aligned} \tag{B.22}$$

The second inequality is due to the acquisition algorithm. The last equality is due to the fact that $\mathrm{ucb}_t^{f}(\boldsymbol{x}) - \mathrm{lcb}_t^{f}(\boldsymbol{x}) = 2\beta_t\sigma_{t-1}(\boldsymbol{x}_t)$ by definition, as well as $\mathrm{ucb}_t^{var}(\boldsymbol{x}) - \mathrm{lcb}_t^{var}(\boldsymbol{x}) =$

$2\beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}_t)$.

Note that at each iteration t we take k measurements, hence the total number of measurements is Tk. Thus, we can bound the cumulative regret by

$$R_T = \sum_{t=1}^{T} kr(\boldsymbol{x}_t) \leq k\sum_{t=1}^{T} 2\beta_t\sigma_{t-1}(\boldsymbol{x}_t|\hat{\Sigma}_{t-1}) + k\sum_{t=1}^{T} 2\alpha\beta_t^{var}\sigma_{t-1}^{var}(\boldsymbol{x}_t)$$
$$\leq 2k\beta_T\sum_{t=1}^{T}\sigma_{t-1}(\boldsymbol{x}_t|\hat{\Sigma}_{t-1}) + 2k\alpha\beta_T^{var}\sum_{t=1}^{T}\sigma_{t-1}^{var}(\boldsymbol{x}_t). \tag{B.23}$$

***Step 3**: On bounding maximum information gain.*
We follow the notion of information gain $I(\hat{m}_{1:T}, f_{1:T})$ computed assuming that $\hat{m}_{1:T} = [\hat{m}_k(\boldsymbol{x}_1), \ldots, \hat{m}_k(\boldsymbol{x}_T)]^T$ with $\hat{m}_k(\boldsymbol{x}_t) = \frac{1}{k}\sum_{i=1}^{k} y_i(\boldsymbol{x}_t)$ (Eq. (5.2)). Under the modelling assumptions $f_{1:T} \sim \mathcal{N}(0, \lambda^{-1}K_T)$, and $\hat{m}_{1:T} \sim \mathcal{N}(f_{1:T}, \mathrm{diag}(\bar{\varrho}^2/k))$ with variance-proxy $\bar{\varrho}^2/k$, the information gain is:

$$I(\hat{m}_{1:T}, f_{1:T}) := \sum_{t=1}^{T}\frac{1}{2}\ln\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k))}{\bar{\varrho}^2/k}\right). \tag{B.24}$$

We define the corresponding maximum information gain $\hat{\gamma}_T = \max_{A\subset\mathcal{X},|A|=T} I(\hat{m}_{1:T}, f_{1:T})$

$$\hat{\gamma}_T := \max_{A\subset\mathcal{X},|A|=T}\sum_{t=1}^{T}\frac{1}{2}\ln\left(1 + \frac{\sigma_{t-1}^2(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k))}{\bar{\varrho}^2/k}\right). \tag{B.25}$$

Analogously, for $\rho(\boldsymbol{x})$ with the posterior $\mathcal{N}(\mu_t^{var}(\boldsymbol{x}), (\sigma_t^{var}(\boldsymbol{x}))^2)$, the information gain is defined as:

$$I(\hat{s}_{1:T}^2, \rho_{1:T}^2) := \frac{1}{2}\sum_{t=1}^{T}\ln\left(1 + \frac{(\sigma_{t-1}^{var})^2(\boldsymbol{x})}{\rho_\eta^2(\boldsymbol{x}_t)}\right). \tag{B.26}$$

Then, the corresponding maximum information gain Γ_T is as follows:

$$\Gamma_T := \max_{A\subset\mathcal{X},|A|=T} I(\hat{s}_{1:T}^2, \rho_{1:T}^2) = \max_{A\subset\mathcal{X},|A|=T}\frac{1}{2}\sum_{t=1}^{T}\ln\left(1 + \frac{(\sigma_{t-1}^{var})^2(\boldsymbol{x})}{\rho_\eta^2(\boldsymbol{x}_t)}\right), \tag{B.27}$$

where A is again a set of size T with points $\{x_1, \ldots, x_T\}$.

***Step 4**: On bounding $\sum_{t=1}^{T}\sigma_{t-1}(\boldsymbol{x}_t|\hat{\Sigma}_{t-1})$ and $\sum_{t=1}^{T}\sigma_{t-1}^{var}(\boldsymbol{x}_t)$*
We repeat the corresponding derivation for known $\rho^2(\boldsymbol{x})$, recalling that $\rho^2(\boldsymbol{x}) \leq \bar{\varrho}^2, \forall x \in \mathcal{X}$:

$$\sum_{t=1}^{T}\sigma_{t-1}(\boldsymbol{x}_t|\hat{\Sigma}_{t-1}) = \sum_{t=1}^{T}\frac{\bar{\varrho}}{\bar{\varrho}}\sigma_{t-1}(\boldsymbol{x}_t|\hat{\Sigma}_{t-1}) \leq \sqrt{T\sum_{t=1}^{T}\frac{\bar{\varrho}^2}{k}\frac{\sigma_{t-1}^2\big(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k)\big)}{\bar{\varrho}^2/k}}$$
$$\leq \sqrt{T\frac{\bar{\varrho}^2}{k}\frac{k/\bar{\varrho}^2}{\ln(1+k/\bar{\varrho}^2)}\sum_{t=1}^{T}\ln\left(1 + \frac{\sigma_{t-1}^2\big(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k)\big)}{\bar{\varrho}^2/k}\right)}$$

$$\leq \sqrt{\frac{2T}{\ln(1+k/\bar{\varrho}^2)} \underbrace{\sum_{t=1}^{T} \frac{1}{2} \ln\left(1 + \frac{\sigma_{t-1}^2\big(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k)\big)}{\bar{\varrho}^2/k}\right)}_{\text{mutual information Eq. (B.24)}}}. \tag{B.28}$$

Here, the first inequality follows from Cauchy-Schwarz inequality and the fact that $\sigma_t(\boldsymbol{x}_t|\hat{\Sigma}_t) \leq \sigma_t(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k))$. The latter holds by the definition of $\hat{\Sigma}_t$, particularly:

$$\begin{aligned}
\sigma_t^2(\boldsymbol{x}_t|\hat{\Sigma}_t) &= \frac{1}{\lambda}(\kappa(\boldsymbol{x},x) - \kappa_t(\boldsymbol{x})^\top (K_t + \lambda\hat{\Sigma}_t)^{-1}\kappa_t(\boldsymbol{x})),\\
\sigma_t^2\big(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k)\big) &= \frac{1}{\lambda}(\kappa(\boldsymbol{x},x) - \kappa_t(\boldsymbol{x})^\top \big(K_t + \lambda\mathrm{diag}(\bar{\varrho}^2/k)\big)^{-1}\kappa_t(\boldsymbol{x})),\\
\hat{\Sigma}_t &= \tfrac{1}{k}\mathrm{diag}\big(\min\{\mathrm{ucb}_t^{var}(\boldsymbol{x}_1), \bar{\varrho}^2\}, \ldots, \min\{\mathrm{ucb}_t^{var}(\boldsymbol{x}_t), \bar{\varrho}^2\}\big),
\end{aligned}$$

then $\hat{\Sigma}_t \preceq \mathrm{diag}(\bar{\varrho}^2/k)$, and $-(K_t + \lambda\hat{\Sigma}_t)^{-1} \preceq -(K_t + \lambda\mathrm{diag}(\bar{\varrho}^2/k))^{-1}$. That implies

$$\begin{aligned}
\sigma_t^2(\boldsymbol{x}_t|\hat{\Sigma}_t) - \sigma_t^2\big(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k)\big) &= -\kappa_t(\boldsymbol{x})^\top \big(K_t + \lambda\hat{\Sigma}_t)^{-1}\kappa_t(\boldsymbol{x})\big)\\
&\quad + \kappa_t(\boldsymbol{x})^\top (K_t + \lambda\mathrm{diag}(\bar{\varrho}^2/k))^{-1}\kappa_t(\boldsymbol{x})\big)\\
&\leq 0.
\end{aligned}$$

The second inequality in Eq. (B.28) is due to the fact that for any $s^2 \in [0, k/\bar{\varrho}^2(\boldsymbol{x}_t)]$ we can bound $s^2 \leq \frac{k/\bar{\varrho}^2(\boldsymbol{x}_t)}{\ln(1+k/\bar{\varrho}^2(\boldsymbol{x}_t))}\ln(1+s^2)$, that also holds for $s^2 := \frac{\sigma_{t-1}^2\left(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k)\right)}{\bar{\varrho}^2/k}$ since for any $\lambda \geq 1$

$$\frac{\sigma_{t-1}^2\big(\boldsymbol{x}_t|\mathrm{diag}(\bar{\varrho}^2/k)\big)}{\bar{\varrho}^2/k} \leq \frac{\lambda^{-1}\kappa(\boldsymbol{x}_t,x_t)}{\bar{\varrho}^2/k} \leq k/\bar{\varrho}^2.$$

Similarly, we bound

$$\begin{aligned}
\sum_{t=1}^{T} \sigma_{t-1}^{var}(\boldsymbol{x}_t) = \sum_{t=1}^{T} \frac{\rho_\eta(\boldsymbol{x}_t)}{\rho_\eta(\boldsymbol{x}_t)}\sigma_{t-1}^{var}(\boldsymbol{x}_t) &\leq \sqrt{T\sum_{t=1}^{T}\rho_\eta^2(\boldsymbol{x}_t)\frac{(\sigma_{t-1}^{var})^2(\boldsymbol{x}_t)}{\rho_\eta^2(\boldsymbol{x}_t)}}\\
&\leq \sqrt{\frac{2T}{\ln(1+\mathcal{R}^{-2})} \underbrace{\sum_{t=1}^{T}\frac{1}{2}\ln\left(1 + \frac{(\sigma_{t-1}^{var})^2(\boldsymbol{x}_t)}{\rho_\eta^2(\boldsymbol{x}_t)}\right)}_{\text{mututal information Eq. (B.26)}}},
\end{aligned} \tag{B.29}$$

in the above we define $\mathcal{R}^2 := \max_{x\in A}\rho_\eta^2(\boldsymbol{x}), A = \{x_1, \ldots, x_T\}$.

***Step 5**: On bounding cumulative regret $R_T = \sum_{t=1}^{T} kr(\boldsymbol{x}_t)$*

Combining the above three steps together, we obtain with probability $1 - 2\delta$

$$R_T \leq \beta_T k\sqrt{\frac{2T\hat{\gamma}_T}{\ln(1+k/\bar{\varrho}^2)}} + \alpha\beta_T^{var}k\sqrt{\frac{2T\Gamma_T}{\ln(1+\mathcal{R}^{-2})}}. \tag{B.30}$$

B.5 Proof of Corollary 1

Corollary 1.1 *Consider the setup of Theorem 1. Let $A = \{x_t\}_{t=1}^T$ denote actions selected by* RAHBO *over T rounds. Then, with probability at least $1-\delta$, the reported point $\hat{\boldsymbol{x}}_T := \arg\max_{x_t \in A} \mathrm{lcb}_t^{MV}(\boldsymbol{x}_t)$, where $\mathrm{lcb}_t^{MV}(\boldsymbol{x}_t) = \mathrm{lcb}_t^{f}(\boldsymbol{x}) - \alpha\ \mathrm{ucb}_t^{var}(\boldsymbol{x})$, achieves ϵ-accuracy, i.e., $MV(\boldsymbol{x}^*) - MV(\hat{\boldsymbol{x}}_T) \leq \epsilon$, after $T \geq \frac{4\beta_T^2 \hat{\gamma}_T / \ln(1+k/\bar{\varrho}^2) + 4\alpha(\beta_t^{var})^2 \Gamma_T / \ln(1+\mathcal{R}^{-2})}{\epsilon^2}$ rounds.*

Proof. We select the maximizer of $\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)$ over the past points x_t:

$$\hat{x}_T := x_{t^*}, \text{ where } t^* := \arg\max_t \{\mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)\} = \arg\min_t \{\mathrm{MV}(\boldsymbol{x}^*) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)\},$$

since adding a constant does not change the solution. We denote $\hat{r}(\boldsymbol{x}_t) := \mathrm{MV}(\boldsymbol{x}^*) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)$. Then we obtain the following bound

$$\begin{aligned}
\mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_{t^*}) &\leq \mathrm{MV}(\boldsymbol{x}^*) - \mathrm{lcb}_{t^*}^{\mathrm{MV}}(\boldsymbol{x}_{t^*}) = \frac{1}{T}\sum_{t=1}^{T} \hat{r}(\boldsymbol{x}_{t^*}) \\
&\leq \frac{1}{T}\sum_{t=1}^{T} \hat{r}(\boldsymbol{x}_t) = \frac{1}{T}\sum_{t=1}^{T} \left(\mathrm{MV}(\boldsymbol{x}^*) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)\right) \\
&\leq \frac{1}{T}\sum_{t=1}^{T} \left(\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}^*) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)\right) \\
&\leq \frac{1}{T}\sum_{t=1}^{T} \left(\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}_t) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)\right). \qquad \text{(B.31)}
\end{aligned}$$

In the above, the first inequality holds with high probability by definition $\mathrm{lcb}_{t^*}^{\mathrm{MV}}(\boldsymbol{x}_{t^*}) \leq \mathrm{MV}(\boldsymbol{x}_{t^*})$, the second inequality is due to $t^* := \arg\min_t \hat{r}(\boldsymbol{x}_t)$ and therefore $\hat{r}(\boldsymbol{x}_{t^*}) \leq \hat{r}(\boldsymbol{x}_t)\ \forall t = 1, \ldots, T$. The third inequality holds since $\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}) \geq \mathrm{MV}(\boldsymbol{x})$ with high probability, and the fourth is due to $\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}_t) \geq \mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x})$ for every x, since x_t is selected via Algorithm 2.

Recalling Eq. (B.30), note that the following bounds hold:

$$R_T = \sum_{t=1}^{T} kr(\boldsymbol{x}_t) \leq \sum_{t=1}^{T} k(\mathrm{ucb}_t^{\mathrm{MV}}(\boldsymbol{x}_t) - \mathrm{lcb}_t^{\mathrm{MV}}(\boldsymbol{x}_t)) \leq \beta_T k \sqrt{\frac{2T\hat{\gamma}_T}{\ln(1+k/\bar{\varrho}^2)}} + \alpha\beta_T^{var} k \sqrt{\frac{2T\Gamma_T}{\ln(1+\mathcal{R}^{-2})}}. \qquad \text{(B.32)}$$

Combining the above Eq. (B.32) with Eq. (B.31) we can get the following upper bound

$$\begin{aligned}
\mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_{t^*}) &\leq \frac{\beta_T k\sqrt{2T\hat{\gamma}_T / \ln(1+k/\bar{\varrho}^2)} + \alpha\beta_T^{var} k\sqrt{2T\Gamma_T / \ln(1+\mathcal{R}^{-2})}}{kT} \\
&\leq \frac{\sqrt{4\left(k\beta_T^2\hat{\gamma}_T / \ln(1+k/\bar{\varrho}^2) + \alpha k(\beta_t^{var})^2\Gamma_T / \ln(1+\mathcal{R}^{-2})\right)}}{\sqrt{kT}}.
\end{aligned}$$

Therefore, for Tk samples with $Tk \geq \frac{4(k\beta_T^2\hat{\gamma}_T/\ln(1+k/\bar{\varrho}^2)+\alpha k(\beta_t^{var})^2\Gamma_T/\ln(1+\mathcal{R}^{-2}))}{\epsilon^2}$ we finally obtain

$$\mathrm{MV}(\boldsymbol{x}^*) - \mathrm{MV}(\boldsymbol{x}_{t^*}) \leq \epsilon.$$

■

APPENDIX C

Experiments and settings of Chapter 3

Implementation and resources We implemented all our experiments using Python and BoTorch [Bal+20].[1] We ran our experiments on an Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz machine.

C.1 Synthetic examples

C.1.1 Example function

We provide additional visualizations for the example sine function in Figure C.1. These examples demonstrate that exploration-exploitation trade-off (as in GP-UCB) might not be enough to prefer points with lower noise and GP-UCB might tend to acquire points with higher variance. In contrast, RAHBO, initialized with the same point, prefers points with lower risk inherited in noise.

C.1.2 Branin

We provide additional visualizations, experimental details and results. Firstly, we plot the noise-perturbed objective function in Figure C.2 in addition to the visualization in Figure 3.1c. In Figure C.3, we plot cumulative regret and simple mean-variance regrets that extends the results in Figure 3.5a with RAHBO-US. The general setting is the same as described for Figure 3.5a: we use 10 initial samples, repeat each evaluation $k = 10$ times, and RAHBO-US additionally uses 10 samples for learning the variance function with uncertainty sampling. During the optimization, RAHBO-US updates the GP model for variance function after every acquired point.

[1] https://botorch.org/

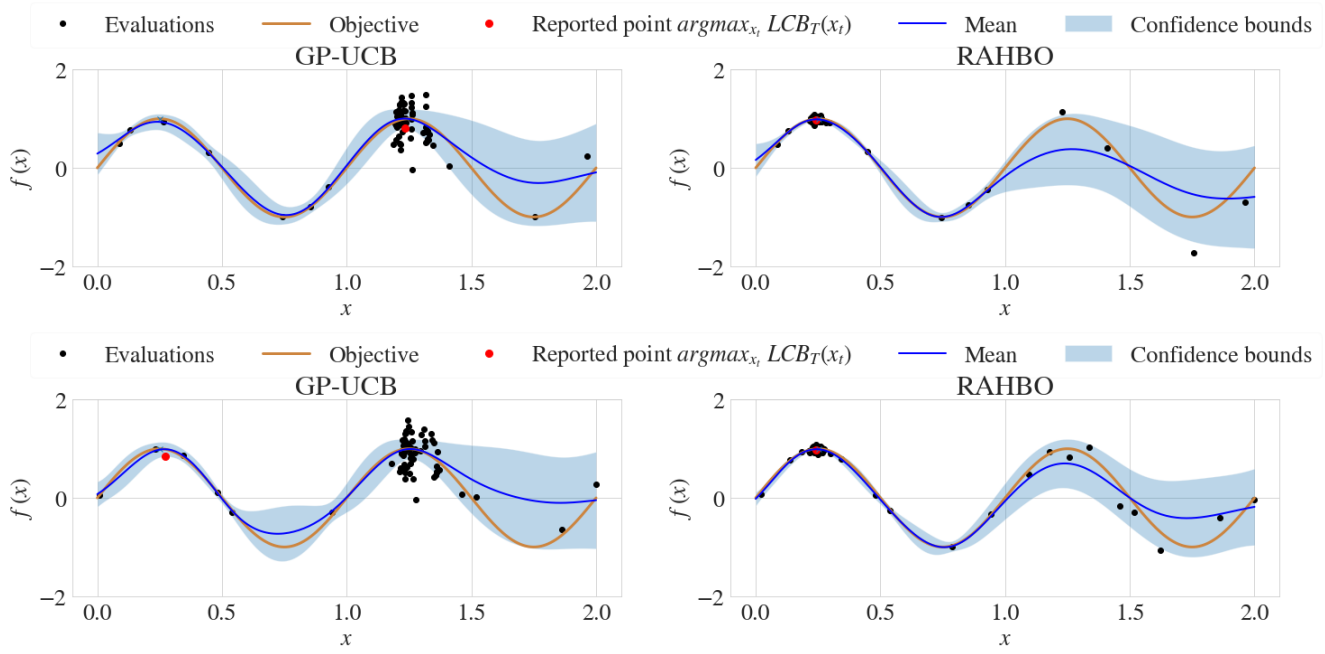
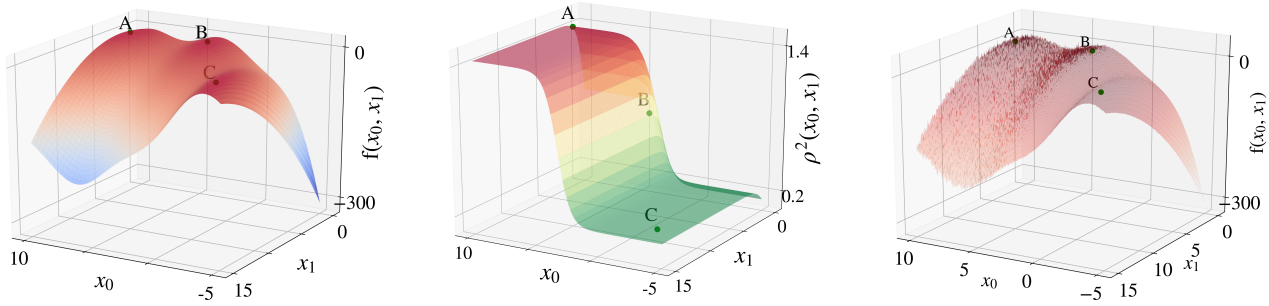
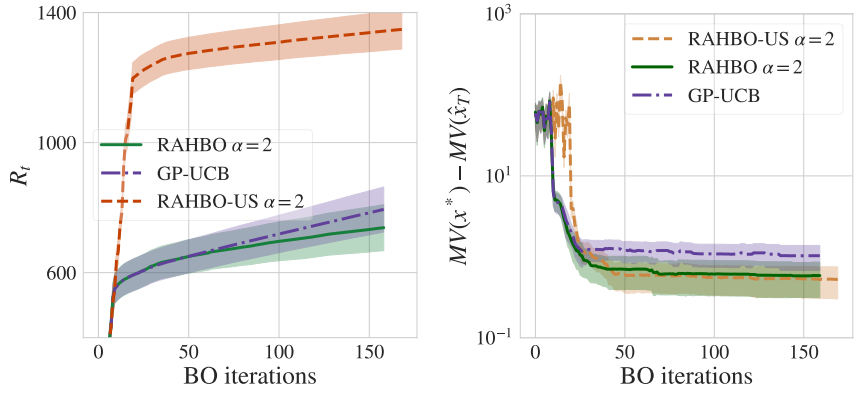


Figure C.1: Additional examples for Section 3.4.1 (each row corresponds to one initialization). GP models fitted for GP-UCB (left column) and RAHBO (right column) for sine function. After initialization with the same sampled points, GP-UCB concentrates on the high-noise region whereas RAHBO prefers small variance.



(a) Unknown objective **(b)** Unknown variance **(c)** Noise-perturbed evaluations

Figure C.2: Visualization of noise-perturbed function landscape:(a) Unknown objective with 3 global maxima marked as (A, B, C). (b) Heteroscedastic noise variance over the same domain: the noise level at (A, B, C) varies according to the sigmoid function. (c) Noise-perturbed evaluations: A is located in the noisiest region.



(a) **(b)**

Figure C.3: Branin: (a) Cumulative regret. (b) Suboptimality w.r.t. MV

C.2 Random Forest tuning

Experiment motivation: Consider the motivating example first: the optimized RF model will be exploited under the data drift over time, e.g., detecting fraud during a week. We are interested not only in high performance *on average* but also in low variance across the results. Particularly, the first can be a realization of the decent result in the first days and unacceptable result in the last days, and the latter ensures lower dispersion over the days while keeping a reasonable mean. In this case, when training an over-parametrized model that is prone to overfitting (to the training data), e.g., Random Forest (RF) with deep trees, high variance in validation error might be observed. In contrast, a model that is less prone to overfitting can result into a similar validation error with lower variance.

RF specifications: We use scikit-learn implementation of RF. The RF search spaces for BO are listed in Table D.1 and other parameters are the default provided by scikit-learn. [2] During BO, we transform the parameter space to the unit-cube space.

Dataset: We tune RF on a dataset of fraudulent credit card transactions [LB21] originally announced for Kaggle competition.[3] It is a highly imbalanced dataset that consists of 285k transactions and only 0.2% are fraud examples. The transactions occurred in two days and each has a time feature that contains the seconds elapsed between each transaction and the first transaction in the dataset. We use the time feature to split the data into train and validation sets such that validation transactions happen later than the training ones. The distribution of the fraud and non-fraud transactions in time is presented in Figure C.4.

In BO, we collect evaluation in the following way: we fix the training data to be the first half of the transactions, and the rest we split into 5 validation folds that are consecutive in time. The RF model is then trained on the fixed training set, and evaluated on the validations sets. We use a balanced accuracy score that takes the imbalance in the data into account.

task	hyperparameter	search space
RandomForest	n_estimators	[1, 100]
	max_features	[5, 28]
	max_depth	[1, 15]

Table C.1: Search space description for RF.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] https://www.kaggle.com/mlg-ulb/creditcardfraud

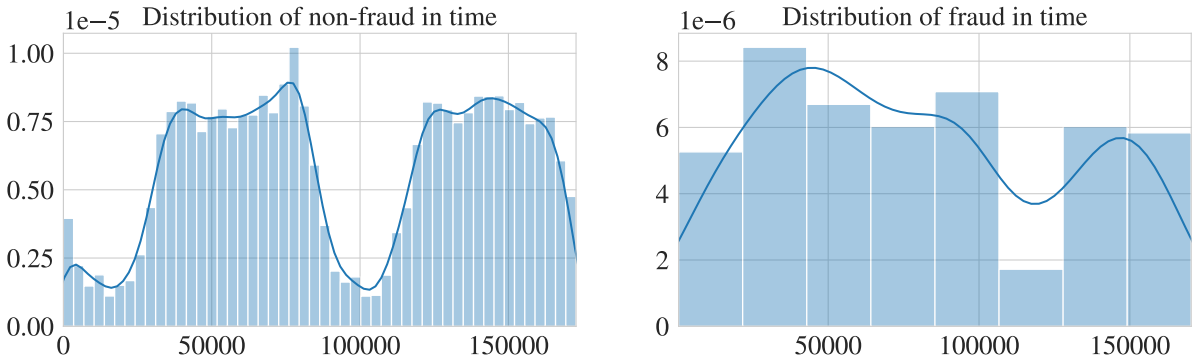
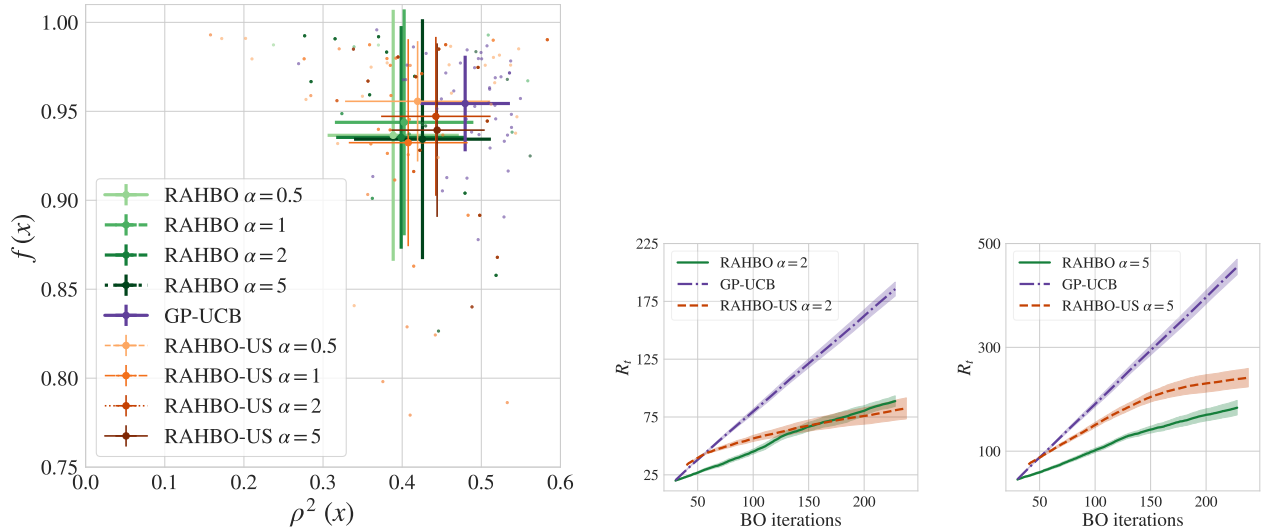


Figure C.4: Distribution of non-fraud (left) and fraud (right) transactions in the dataset

C.3 Tuning Swiss free-electron laser (SwissFEL)



(a) Mean-variance tradeoff (SwissFEL) **(b)** Cum. regret ($\alpha = 0.5$) **(c)** Cum. regret ($\alpha = 1$)

Figure C.5: (a) We plot standard deviation error bars for $f(\boldsymbol{x})$ and $\rho^2(\boldsymbol{x})$ at the *reported point* by the best observed value $x^{(T)} = \arg\max_{x_t} y_t(\boldsymbol{x}_t)$ after BO completion for SwissFEL. The mean and std of the error bars are taken over the repeated BO experiments. The results demonstrate, that reporting based on the best-observed value inherits high noise and as the result all methods perform similarly. Intuitively, when noise variance is high, it is possible to observe higher values. That however also inherits observing much lower value at this point, this leading to very non-robust solutions. (b-c) Cumulative regret for $\alpha = 2$ and $\alpha = 5$.

APPENDIX D

Experiments and settings for Chapter 4

D.1 Experiments setting

D.1.1 BO setting

We used an internal BO implementation where expected improvement (EI) together with Mat'ern-52 kernel in the GP are used. The hyperparameters of the GP include output noise, a scalar mean value, bandwidths for every input dimension, 2 input warping parameters, and a scalar covariance scale parameter. The closest open-source implementations are GPyOpt using input warped GP [1] or AutoGluon BayesOpt searcher [2]. We maximize type II likelihood to learn the GP hyperparameters in our experiments.

D.1.2 Algorithm

We present the pseudo-code for the termination criterion in Algorithm 5.

D.1.3 Search spaces for cross-validation experiments

XGBoost (XGB) and RandomForest (RF) are based on scikit-learn implementations and their search spaces are listed in Table D.1.

D.1.4 Datasets in cross-validation experiments

We list the datasets used in our experiments, as well as their characteristics and sources in Table Table D.2. For each dataset, we first randomly draw 20% as a test set and for the rest, we use 10-fold cross-validations for regression datasets and 10-fold stratified cross-validation for classification datasets. The actual data splits depend on the seed controlled in our experiments. For a given experiment, we use the same data splits for the whole tuning problem. For the experiments without cross-validation, we use 20% dataset as a validation set and the rest as a training set.

[1] https://github.com/SheffieldML/GPyOpt
[2] https://github.com/awslabs/autogluon

Algorithm 5 BO for HPO with cross-validation and automatic termination

Require: Model $\mathcal{M}_{\boldsymbol{x}}$ parametrized by $\boldsymbol{x} \in \mathcal{X}$, data $\{\mathrm{D}_1, \ldots, \mathrm{D}_k\}$ for k-fold cross-validation each $\mathrm{D} = \{(\mathrm{x}_i, \mathrm{y}_i\}_i$,
acquisition function $\alpha(\boldsymbol{x})$

Initialize $y_t^* = +\infty$ and $\mathcal{D}_t = \{\}$
for $t = 1, 2, \ldots$ **do**
  Sample $\boldsymbol{x}_t \in \arg\max_{\boldsymbol{x} \in \mathcal{X}} \alpha(\boldsymbol{x})$
  for $i = 1, 2, \ldots, k$ **do**
    Fit the model $\mathcal{M}_{\boldsymbol{x}_t}(\cdot; \mathrm{D}_{-i})$, where $\mathrm{D}_{-i} = \cup_{j \neq i} \mathrm{D}_i$
    Evaluate the fitted model $y_t^i = \frac{1}{|\mathrm{D}_i|} \sum_{\mathrm{x}_i, \mathrm{y}_i \in \mathrm{D}_i} \ell(\mathrm{y}_i, \mathcal{M}_{\boldsymbol{x}_t}(\mathrm{x}_i, \mathrm{D}_{-i}))$
  end for
  Calculate the sample mean $y_t = \frac{1}{k} \sum_k y_t^i$,
  if $y_t \leq y_t^*$ **then**
    Update $y_t^* = y_t$ and the current best $\boldsymbol{x}_t^* = \boldsymbol{x}_t$
    Calculate the sample variance $s_{\mathrm{cv}}^2 = \frac{1}{k} \sum_i (y_t - y_t^i)^2$
    Calculate the variance estimate $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}_t^*) \approx \left(\frac{1}{k} + \frac{|\mathrm{D}_i|}{|\mathrm{D}_{-i}|}\right) s_{\mathrm{cv}}^2$ from Eq. (4.8)
  end if
  Update $\mathcal{D}_t = \mathrm{D}_{t-1} \cup \boldsymbol{x}_t$ and $y_{1:t} = y_{1:t-1} \cup y_t$
  Update σ_t, μ_t with (2.8) and (2.9)
  Calculate upper bound $\bar{r}_t := \min_{\boldsymbol{x} \in \mathcal{D}_t} \mathrm{ucb}_t(\boldsymbol{x}) - \min_{\boldsymbol{x} \in \boldsymbol{x}} \mathrm{lcb}_t(\boldsymbol{x})$ for simple regret from Eq. (4.7)
  if the condition $\bar{r}_t \leq \sqrt{\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}_t^*)}$ holds **then**
    terminate BO loop
  end if
end for
Output: $\boldsymbol{x}_t^*$

D.2 Detailed results

Figure 4.3 demonstrates the results of threshold study EI and PI baselines for cross-validation. Figure 4.4 shows results for extended set of parameter i for Conv-i baseline.

We also show the scatter plots of RTC and RYC scores for different automatic termination methods on HPO-Bench-datasets in Figure D.1 and the results on NAS-Bench-201 in Figure 4.2.

4.2.1 Detailed numbers of RYC and RTC scores

We report detailed RYC scores and RTC scores of different HPO automatic termination methods for the experiments in the main text in Table 4.3, Table 4.4 and Table 4.5.

4.2.2 Correlation between validation and test metrics

In Figure 4.5, we show the correlation between validation and test metrics of hyperparameters when tuning XGB and RF on tst-census dataset in Figure 4.5.

tasks	hyperparameter	search space	scale
XGBoost	n_estimators	$[2, 2^9]$	log
	learning_rate	$[10^{-6}, 1]$	log
	gamma	$[10^{-6}, 2^6]$	log
	min_child_weight	$[10^{-6}, 2^5]$	log
	max_depth	$[2, 2^5]$	log
	subsample	$[0.5, 1]$	linear
	colsample_bytree	$[0.3, 1]$	linear
	reg_lambda	$[10^{-6}, 2]$	log
	reg_alpha	$[10^{-6}, 2]$	log
RandomForest	n_estimators	$[1, 2^8]$	log
	min_samples_split	$[0.01, 0.5]$	log
	max_depth	$[1, 5]$	log

Table D.1: Search spaces description for each algorithm.

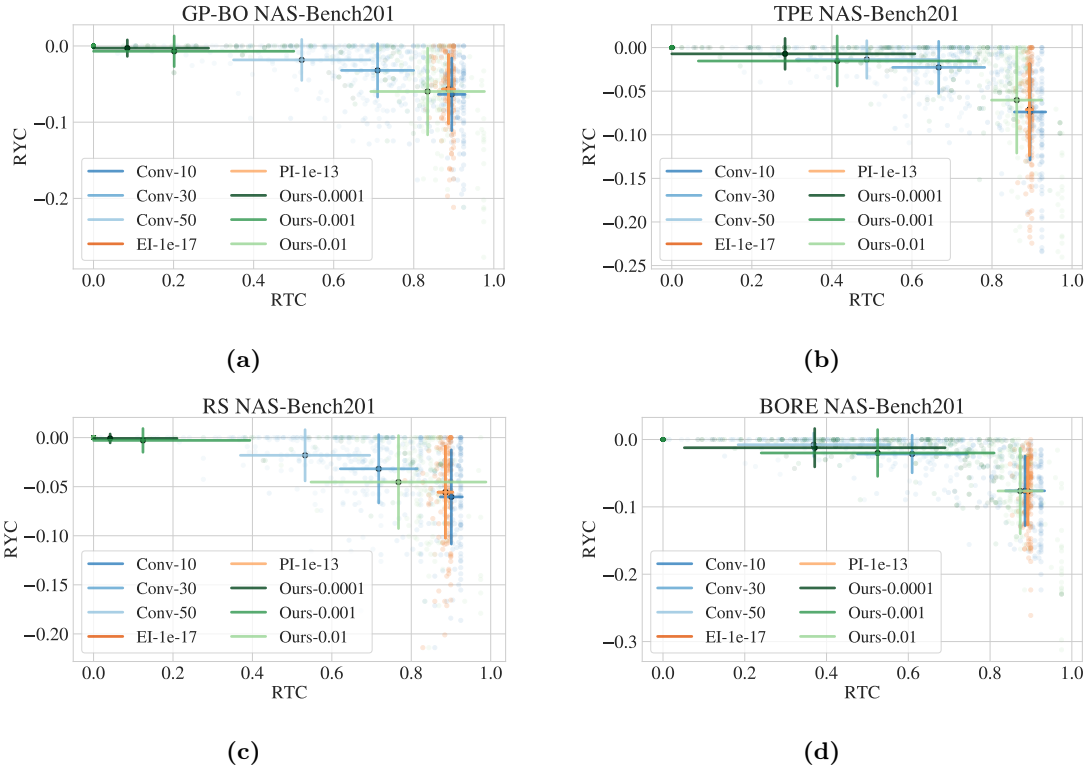


Figure 4.2: Fig. (a) - (d), the mean and standard deviation of RYC and RTC scores for considered automatic termination methods on NAS-Bench-201 datasets using GP-based BO (GP-BO), Random Search (RS), TPE and BORE optimizers. The mean value is shown in the big dot and the standard deviation is shown as an error bar in both dimensions.

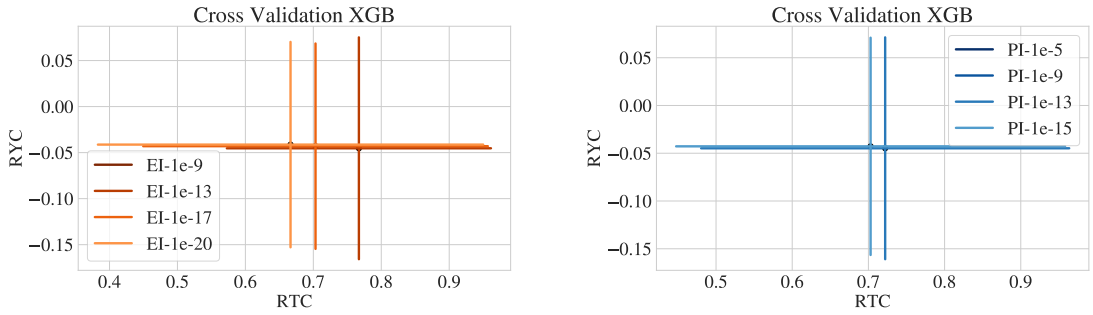
Figure D.1: Fig. (a) - (d), the mean and standard deviation of RYC and RTC scores for considered automatic termination methods on HPO-Bench datasets using GP-based BO (GP-BO), Random Search (RS), TPE and BORE optimizers. The mean value is shown in the big dot and the standard deviation is shown as an error bar in both dimensions.

4.2.3 The choice of parameter β_t

High-probability concentration inequalities (aka confidence bounds) are important to reason about the unknown objective function and are used for theoretically grounded

dataset	problem_type	n_rows	n_cols	n_classes	source
openml14	classification	1999	76	10	openml
openml20	classification	1999	240	10	openml
tst-hate-crimes	classification	2024	43	63	data.gov
openml-9910	classification	3751	1776	2	openml
farmads	classification	4142	4	2	uci
openml-3892	classification	4229	1617	2	openml
sylvine	classification	5124	21	2	openml
op100-9952	classification	5404	5	2	openml
openml28	classification	5619	64	10	openml
philippine	classification	5832	309	2	data.gov
fabert	classification	8237	801	2	openml
openml32	classification	10991	16	10	openml
openml34538	regression	1744	43	-	openml
tst-census	regression	2000	44	-	data.gov
openml405	regression	4449	202	-	openml
tmdb-movie-metadata	regression	4809	22	-	kaggle
openml503	regression	6573	14	-	openml
openml558	regression	8191	32	-	openml
openml308	regression	8191	32	-	openml

Table D.2: Datasets used in our experiments including their characteristics and sources.



(a) **(b)**

Figure 4.3: Detailed threshold study for the EI (left) and PI (right) baselines. Mean and standard deviation of RYC and RTC scores for EI and PI.

convergence guarantees in some (GP-UCB-based) BO methods ([Sri+10; Ha+19; Kir+20; Mak+21b]). There, β_t stands for the parameter that balances between exploration vs. exploitation and ensures the validity of the confidence bounds. The choice of β_t is then guided by the assumptions made on the unknown objective, for example, the objective being a sample from a GP or the objective having the bounded norm in RKHS (more agnostic case used in Section 3).

In our experiments, we follow the common practice of scaling down β_t which is usually used to improve performance over the (conservative) theoretically grounded values (see e.g., [Sri+10; Kir+20; Mak+21b]). Particularly, throughout this paper, we set $\beta_t = 2\log(|\mathcal{X}|t^2\pi^2/6\delta)$ where $\delta = 0.1$ and $|\mathcal{X}|$ is set to be the number of hyperparameters.

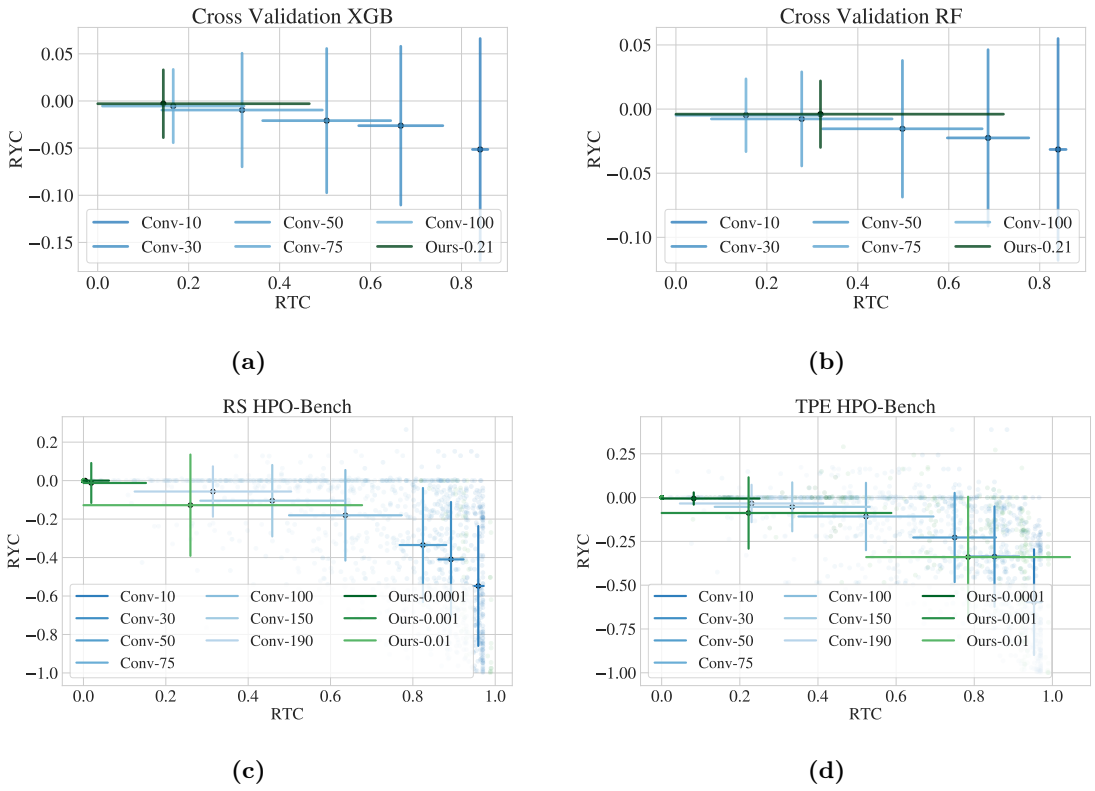
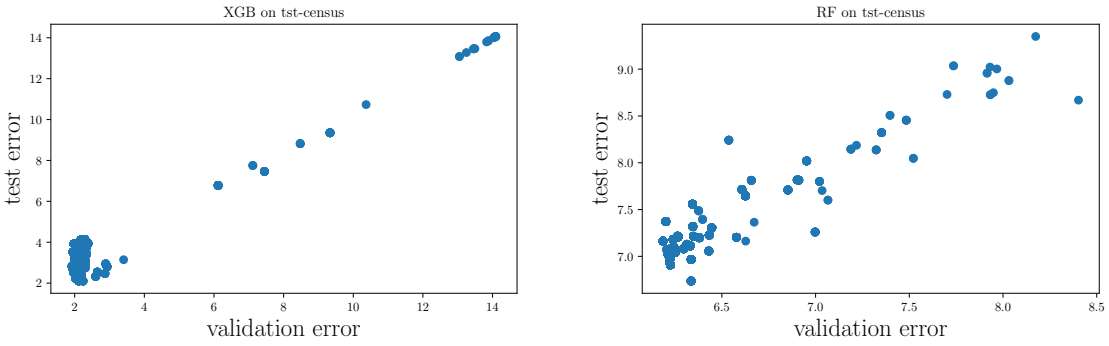


Figure 4.4: Parameter i study for Conv-i baseline. The mean and standard deviation of RYC and RTC scores for (first line) HPO tuning XGB (a) and RF (b) using cross-validation and (second line) for HPO-Bench datasets (c-d). The Figure shows that even though there is an ideal for Conv-i, it changes not only across tasks and methods but even across different repetitions of a single method. The latter makes it particularly challenging to define a suitable i, since i is a fixed, predetermined choice, that does not take the observed data into account (in contrast to our method that refines the regret estimation as the data is being observed).



(a) Training XGB on test-census dataset

(b) Training RF on test-census dataset.

Figure 4.5: We show validation error for training XGB (a) and RF (b) on `tst-census` dataset on the x-axis and test error on the y-axis. In the *low* error region, the validation metrics are not well correlated with the test metrics.

We then further scale it down by a factor of 5 as defined in the experiments in [Sri+10]. We provide an ablation study on the choice of β_t in Figure 4.6.

	RTC		RYC	
algo	RF	XGB	RF	XGB
Conv_10	0.840	0.841	-0.031	-0.051
Conv_30	0.686	0.666	-0.022	-0.026
Conv_50	0.498	0.504	-0.015	-0.021
EI_1e-08	0.896	0.850	-0.057	-0.052
EI_1e-12	0.895	0.779	-0.055	-0.047
EI_1e-16	0.893	0.718	-0.052	-0.045
PI_1e-4	0.898	0.875	-0.059	-0.059
PI_1e-08	0.895	0.814	-0.055	-0.052
PI_1e-12	0.894	0.739	-0.055	-0.044
Ours_0.21	0.318	0.144	-0.004	-0.003
Ours_0.5	0.580	0.224	-0.013	-0.006

Table 4.3: RTC and RYC scores for early stopping methods in cross-validation benchmarks.

	RTC				RYC			
dataset	naval	parkinsons	protein	slice	naval	parkinsons	protein	slice
Conv_10	0.943	0.947	0.946	0.942	-0.605	-0.582	-0.117	-0.432
Conv_30	0.826	0.837	0.837	0.840	-0.064	-0.235	-0.021	-0.119
Conv_50	0.748	0.729	0.734	0.747	-0.038	-0.107	-0.008	-0.058
Ours_0.0001	0.790	0.018	0.198	0.822	-0.041	-0.012	-0.005	-0.072
Ours_0.001	0.910	0.038	0.271	0.934	-0.220	-0.031	-0.018	-0.281
Ours_0.01	0.941	0.901	0.906	0.953	-0.498	-0.378	-0.071	-0.466

Table 4.4: RTC and RYC scores for early stopping methods in HPO-Bench.

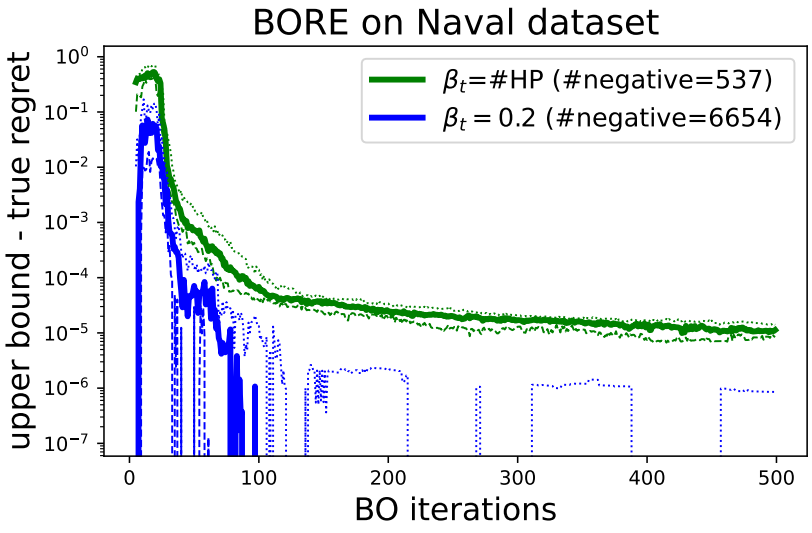


Figure 4.6: The differences between upper bound and true regret for every BO iteration when using BORE to tune an MLP on the Naval dataset. The number of negative differences (the upper bound is smaller than the true regret) is shown in the legend next to the two options for computing β_t.

	RTC			RYC		
dataset	ImageNet	cifar10	cifar100	ImageNet	cifar10	cifar100
Conv_10	0.880	0.889	0.888	-0.034	-0.098	-0.097
Conv_30	0.612	0.611	0.606	-0.010	-0.019	-0.036
Conv_50	0.372	0.361	0.372	-0.004	-0.006	-0.014
Ours_0.0001	0.274	0.311	0.519	-0.002	-0.008	-0.026
Ours_0.001	0.377	0.622	0.582	-0.005	-0.023	-0.033
Ours_0.01	0.837	0.902	0.879	-0.022	-0.106	-0.099

Table 4.5: RTC and RYC scores for early stopping methods in NAS-Bench-201.

CHAPTER 5

On improving the statistical characteristics of the threshold

5.1 Stopping threshold under homoscedastic variance

In Section 4.1, we introduce a problem of Bayesian optimization-based hyperparameter optimization: the irreducible discrepancy in the objectives, i.e., the available objective being optimized (validation loss) and the true but unavailable objective (generalization performance). This irreducible discrepancy is characterized in terms of the statistical error of the cross-validation-based estimator. Consequently, a termination criterion for BO stops the iterative reduction of the validation loss to 0 when it might not bring any benefits, i.e., when the statistical error dominates. The termination criterion uses the connection between the statistical error (or the variance) and sample variance obtained from cross-validation evaluations ([Ben00]). Notably, the stopping condition relies on the variance Eq. (4.8) at a particular hyperparameter, whiich is the best one found so far. Though this empirical assumption demonstrates competitive practical results, it actually might not hold uniformly over the whole domain. To this end, we show how to use a set of collected sample variances to get confidence bounds for the unknown variance assumed to be homoscedastic.

Method

We follow the notation introduced in Section 4.1. There, $\hat{f}(\boldsymbol{x})$ denotes the *point estimator* $f(\boldsymbol{x})$, i.e., function or statistics used to estimate the unknown $f(\boldsymbol{x})$, and $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}) = \mathbb{E}[(\hat{f}(\boldsymbol{x}) - \mathbb{E}\hat{f}(\boldsymbol{x}))^2]$ is the estimator variance. Let us further denote the evaluations obtained during the k-fold cross-validation by $\{y_t^i\}_{i=1}^k$. Then, we can compute the sample variance $\hat{s}_t^2$ and sample mean y_t, where the latter is a *point estimate* of $f(\boldsymbol{x}_t)$, i.e., observed numerical value:

$$y_t^i = \frac{1}{|\mathrm{D}_i|} \sum_{\mathrm{x}_i, \mathrm{y}_i \in \mathrm{D}_i} \ell(\mathrm{y}_i, \mathcal{M}_{\boldsymbol{x}_t}(\mathrm{x}_i, \mathrm{D}_{-i})), \tag{5.1}$$

$$y_t = \frac{1}{k} \sum_{i=1}^{k} y_t^i \quad \text{and} \quad \hat{s}_t^2 = \frac{1}{k-1} \sum_{i=1}^{k} \left(y_t^i - y_t\right)^2. \tag{5.2}$$

The same notation is used in Chapter 3 for risk-averse BO with repeated experiment setup.

The question we are interested in is: What type of uncertainty do we have when we estimate the unknown functions $\hat{f}(\boldsymbol{x}_t)$ and $f(\boldsymbol{x}_t)$? Each evaluation y_t^i inherits randomness due to the training procedure: it is an epistemic uncertainty in the model $\mathcal{M}_{\boldsymbol{x}_t}(\mathrm{x}_i, \mathrm{D}_{-i})$ in Eq. (5.1) due to the scarcity of the data D_{-i}, and aleatoric uncertainty due to the stochasticity of the gradient-base optimization. Therefore, we assume that the true $\hat{f}(\boldsymbol{x}_t)$ is perturbed by noise resulting in the following observation model:

$$y_t^i = \hat{f}(\boldsymbol{x}_t) + \xi_t^i, \tag{5.3}$$

where ξ_t^i is a realization of some *homoscedastic* zero-mean Gaussian noise $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ independent across different time steps t and data folds i. In the following proposition, we show how to compute lower confidence bound for the noise variance σ_ξ^2.

Proposition 2. *Consider the expected loss f and its k-fold cross-validation estimator $\hat{f}$ defined in (4.2) and (4.3), and assume the estimator variance $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x})$ is homoscedastic, i.e., $\mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}) = \mathbb{V}\mathrm{ar}\hat{f}(\boldsymbol{x}'), \forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. Let $\{\hat{s}_t^2\}_{t=0}^T$ be the sample variance evaluations defined in Eq. (5.2) and collected according to the observational model Eq. (5.3) with Gaussian noise $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$. Then, with probability $1 - \delta/2$, $\delta \in (0, 1)$, the unknown noise variance is lower bounded as follows:*

$$\sigma_\xi^2 \geq \frac{k-1}{2\tilde{\gamma}^{-1}\left(\frac{k}{2}, \frac{1-\delta}{2}\right)} \sum_{t=0}^{T} \hat{s}_t^2, \tag{5.4}$$

where $\tilde{\gamma}^{-1}(\cdot, \cdot)$ is the inverse of the normalized lower incomplete gamma function.

Proof. In short, the proposition can be proved in four steps as follows: (1) Show that sample variance to noise variance $\frac{(k-1)\hat{s}_t^2}{\sigma_\xi^2} \sim \chi_{k-1}^2$ is chi-squared distributed. (2) Show that the sample variance $\hat{s}_t^2$ is gamma distributed $\hat{s}_t^2 \sim \mathrm{Gamma}\left(\frac{k-1}{2}, \frac{k-1}{2\sigma_\xi^2}\right)$. (3) Show that the mean over sample variances is distributed as $\frac{1}{T}\sum_t^T \hat{s}_t^2 \sim \mathrm{Gamma}\left(\frac{k-1}{2}T, \frac{k-1}{2\sigma_\xi^2}T\right)$. (4) Finally, we can compute an δ-quantile for the obtained Gamma distribution, resulting in the lower bound of interest. Each of the steps is presented below.

Step 1: *Show that the sample variance divided by the noise variance $\frac{(k-1)\hat{s}_t^2}{\sigma_\xi^2} \sim \chi_{k-1}^2$ is chi-squared distributed.*
The noise ξ is zero-mean Gaussian noise, i.e., $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$. Then, following the model $y_t^i = \hat{f}(\boldsymbol{x}_t) + \xi_t^i$, from Eq. (5.3), each evaluation $y_t^i \sim \mathcal{N}(\hat{f}(\boldsymbol{x}_t), \sigma_\xi^2)$ with (unknown) mean $\hat{f}(\boldsymbol{x}_t)$. Then, standardised $z_i := \frac{y_t^i - \hat{f}(\boldsymbol{x}_t)}{\sigma_\xi}$ is a random variable from standard normal distribution $z_i \sim \mathcal{N}(0, 1)$. Let us then show, that the value of interest can be represented as a sum of standard normal random variables, thus, resulting in a chi-squared distribution:

$$\frac{(k-1)\hat{s}_t^2}{\sigma_\xi^2} \overset{①}{=} \frac{\sum_{i=1}^k (y_t^i - y_t)^2}{\sigma_\xi^2} \overset{②}{=} \sum_{i=1}^k \left(\frac{y_t^i - \frac{1}{k}\sum_j y_t^j}{\sigma_\xi} - \frac{\hat{f}(\boldsymbol{x}_t) - \hat{f}(\boldsymbol{x}_t)}{\sigma_\xi}\right)^2 \tag{5.5}$$

$$\overset{③}{=} \sum_{i=1}^{k} \Big(\frac{y_t^i - \hat{f}(\boldsymbol{x}_t)}{\sigma_\xi} - \frac{1}{k} \sum_{j=1}^{k} \frac{y_t^j - \hat{f}(\boldsymbol{x}_t)}{\sigma_\xi} \Big)^2 \tag{5.6}$$

$$\overset{④}{=} \sum_{i=1}^{k} \big(z_i - \hat{m}_z \big)^2 \sim \chi^2_{k-1}, \tag{5.7}$$

where ①, ② are due to the sample variance and mean definitions Eq. (5.2), and ③, ④ are due to reorganizing the terms to get the standard normal variables with sample mean $\hat{m}_z = \frac{1}{k} \sum_{j=1}^{k} \frac{y_t^j - \hat{f}(\boldsymbol{x}_t)}{\sigma_\xi}$. Thus, we have a sum of k terms $(z_i - \hat{m}_z)^2$ that is distributed as χ^2_{k-1}.

Step 2: *Show that the sample variance $\hat{s}_t^2$ is gamma distributed $\hat{s}_t^2 \sim Gamma\Big(\frac{k-1}{2}, \frac{k-1}{2\sigma_\xi^2}\Big)$.*

Recap about gamma distribution. Let us denote gamma distribution with shape a and rate b as $\text{Gamma}(a, b)$. Then, the probabilistic density function is given by $f(x; a, b) = \frac{x^{a-1} e^{-bx} b^a}{\Gamma(a)}$ for $x > 0$ and $a, b > 0$. Note, that chi-squared distribution χ^2_k is gamma distribution with $a = k/2$ and $b = 1/2$, i.e., $\chi^2_k = \text{Gamma}(\frac{k}{2}, \frac{1}{2})$. Moreover, chi-squared χ^2_k scaled by some constant c is gamma with parameters scaled as $c\chi^2_k = \text{Gamma}(\frac{k}{2}, \frac{1}{2c})$. Finally, if T evaluations are drawn from $\text{Gamma}(a, b)$, their sum is distributed as $\text{Gamma}(aT, b)$.

Let us first note that $\hat{s}_t^2$ is distributed as scaled chi-squared $\hat{s}_t^2 \sim \frac{\sigma_\xi^2}{k-1} \chi^2_{k-1}$. Then, scaling chi-squared distribution corresponds to gamma distribution with parameters scaled respectively:

$$\hat{s}_t^2 \sim \frac{\sigma_\xi^2}{k-1} \chi^2_{k-1} = \text{Gamma}\Big(\frac{k-1}{2}, \frac{k-1}{2\sigma_\xi^2} \Big). \tag{5.8}$$

Step 3: *Show that the mean over sample variances is distributed as $\frac{1}{T} \sum_t^T \hat{s}_t^2 \sim Gamma\big(\frac{k-1}{2} T, \frac{k-1}{2\sigma_\xi^2} T\big)$.*

Let the mean over sample variances $\hat{s}^2 := \frac{1}{T} \sum_t^T \hat{s}_t^2$. If T evaluations $\{\hat{s}_1^2 \dots \hat{s}_T^2\}$ are drawn from a gamma distribution with some shape a and rate b, i.e., $\hat{s}_t^2 \sim \text{Gamma}(a, b)$, then, their sample mean $\hat{s}^2$ is gamma distributed $\hat{s}^2 \sim \text{Gamma}(aT, bT)$. Following Eq. (5.8), where $a = \frac{k-1}{2}$ and $b = \frac{k-1}{2\sigma_\xi^2}$, we conclude that $\hat{s}^2 \sim \text{Gamma}\big(\frac{k-1}{2} T, \frac{k-1}{2\sigma_\xi^2} T\big)$.

Step 4: *Derive the confidence interval for unknown σ_ξ^2.*

We use $\hat{s}^2 := \frac{1}{T} \sum_t^T \hat{s}_t^2$ as an estimator for σ_ξ, where $\hat{s}^2$ has gamma distribution as shown above.

Let G_{ab}^p denote a p-quantile of gamma distribution $\text{Gamma}(a, b)$ such that $\mathbb{P}(\hat{s}^2 \geq G_{ab}^p) = p$. Then, $G_{ab}^p := G_{ab}(p) = \frac{\tilde{\gamma}^{-1}(a,p)}{b}$, where $\Gamma(\cdot)$ is the gamma function and $\tilde{\gamma}^{-1}(\cdot, \cdot)$ is the inverse of the normalized lower incomplete gamma function $\tilde{\gamma}(a, bx) = \frac{1}{\Gamma(a)} \int_0^{bx} t^{a-1} e^{-t} dt$ as follows:

$$F_X^{ab}(x) = \frac{1}{\Gamma(a)} \int_0^{bx} t^{a-1} e^{-t} dt = \frac{\gamma(a, bx)}{\Gamma(a)}, \tag{5.9}$$

$$G_{ab}(p) = F_X^{-1}(x), \tag{5.10}$$

$$p = \frac{\gamma(a, bx)}{\Gamma(a)} := \tilde{\gamma}(a, bx), \tag{5.11}$$

$$x = \frac{1}{b}\tilde{\gamma}^{-1}(a, p). \tag{5.12}$$

Then, for some $0 \leq \delta \leq 1$, $\mathbb{P}(\hat{s}^2 \geq G_{ab}^{1-\delta/2}) = \delta/2$ and $\mathbb{P}(\hat{s}^2 \leq G_{ab}^{\delta/2}) = \delta/2$, leading to

$$\mathbb{P}(G_{ab}^{\delta/2} \leq \hat{s}^2 \leq G_{ab}^{1-\delta/2}) = 1 - \delta.$$

$$\mathbb{P}\Big[\tilde{\gamma}^{-1}\big(a, \frac{\delta}{2}\big) \leq b\hat{s}^2 \leq \tilde{\gamma}^{-1}\big(a, 1 - \frac{\delta}{2}\big)\Big] = 1 - \delta, \tag{5.13}$$

$$\mathbb{P}\Big[\frac{T\hat{s}^2(k-1)}{2\tilde{\gamma}^{-1}\big(a, \frac{\delta}{2}\big)} \geq \sigma_\xi^2 \geq \frac{T\hat{s}^2(k-1)}{2\tilde{\gamma}^{-1}\big(a, 1 - \frac{\delta}{2}\big)}\Big] = 1 - \delta, \tag{5.14}$$

That results into the lower confidence bound for σ_ξ^2:

$$\mathrm{lcb}_{\sigma_\xi^2} = \frac{T\hat{s}^2(k-1)}{2\tilde{\gamma}^{-1}\big(a, \frac{1-\delta}{2}\big)}. \tag{5.15}$$

■

5.1.1 Visualizations

Here, we provide the visualization of the confidence bounds for σ_ξ^2 from Eq. (5.15) and compare them with the value used in the termination condition Eq. (4.8).

First, we visualize the convergence of the confidence bounds in Eq. (5.14) as a function of the number of BO iteration T, or, in other words, the number of samples in the sample mean $\hat{s}^2 := \frac{1}{T}\sum_t^T \hat{s}_t^2$. To this end, we assume $\hat{s}^2 = 1$ and plot multiplications factors in

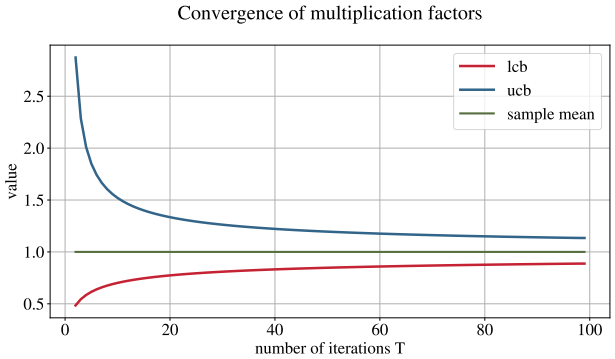


Figure 5.1: Convergence of the confidence bounds lcb and ucb in Eq. (5.14) as a function of number of iterations T. Here, we assume $\hat{s}^2 = 1$ (plotted in green).

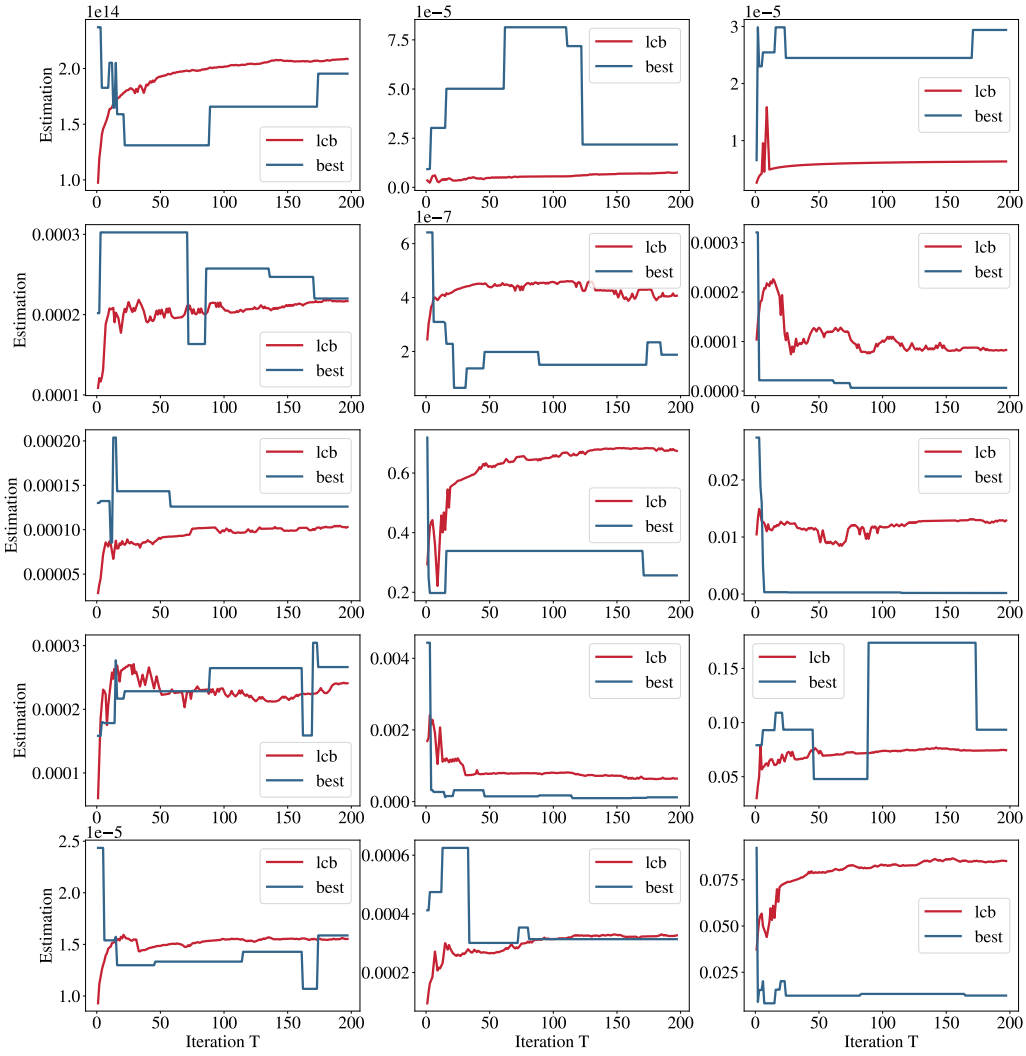


Figure 5.2: Comparison of two stopping thresholds: the lower confidence bound-based from Eq. (5.15) (red) and the variance of the best-found solution (blue) used in Section 4.1.

CHAPTER 6

Method details for Chapter 5

6.1 Sparse linear model via sparsity-encouraging prior

While the number and degree of the features used in the surrogate model (see Section 5.2.1) is a design choice, in practice it is typically unknown which variable interactions matter and thus which features to choose. To discard irrelevant features, one may impose a sparsity-encouraging prior over the weight vector ω [BP18]. However, due to non-conjugacy to the Gaussian likelihood, exact Bayesian inference of the resulting posterior distribution is in general intractable, imposing the need for approximate inference methods. One choice for such a prior is the Laplace distribution, i.e. $p(\omega|\alpha) \propto \exp(-\alpha^{-1}\|\omega\|_1)$, with inverse scale parameter $\alpha > 0$, for which approximate inference techniques based on expectation propagation [Min01] and variational inference [WJ+08] were developed in [See08; SN08; SN11]. Alternatively, one can use a horseshoe prior and use Gibbs sampling to sample from the posterior over weights [BP18]. However, this comes with a significantly larger computational burden, which is a well-known issue for sampling based inference techniques [Bis06]. Lastly, one may consider a spike-and-slab prior with expectation propagation for approximate posterior inference [HHD13; HHS15].

6.2 Pseudocode for Thompson sampling

Algorithm 6 shows pseudocode for the Thompson sampling procedure within MiVaBo.

Algorithm 6 Thompson Sampling

Require: model features $\boldsymbol{\phi}(\boldsymbol{x})$

1: Set $\mathbf{S} = \mathbf{I}$, $\mathbf{m} = \mathbf{0}$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Sample $\widetilde{\omega}_t \sim \mathcal{N}(\mathbf{m}, \mathbf{S}^{-1})$
4: Select input $\widehat{\boldsymbol{x}}_t \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} \widetilde{\omega}_t^\top \boldsymbol{\phi}(\boldsymbol{x})$
5: Query output $y_t = f(\widehat{\boldsymbol{x}}_t) + \varepsilon$
6: Update $\mathbf{S}, \mathbf{m}$ as described in Section 5.2.2.
7: **end for**
8: **Output:** $\widehat{\boldsymbol{x}}_* \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} \mathbf{m}^\top \boldsymbol{\phi}(\boldsymbol{x})$

6.3 Derivation of dual decomposition

One interesting interpretation of our acquisition function optimization problem as defined in Section 5.2.3 is as *maximum a posteriori* (MAP) inference in the undirected graphical model, or Markov random field (MRF) [KFB09], induced by the dependency graph of the involved variables (i.e. the graph in which vertices correspond to variables, and edges appear between variables that interact in some way). We take this perspective and devise a dual decomposition to tackle the MAP estimation problem induced by our particular setting (i.e., interpreting our acquisition function as the energy function of the graphical model), following the formulation of [SGJ11].[1]

Consider a graphical model on the vertex set $\mathcal{V} = V_d \cup V_c$, where the vertices $V_d = \{1, \dots, D_d\}$ and $V_c = \{D_d + 1, \dots, D_d + D_c\}$ correspond to the discrete and continuous variables $\boldsymbol{x}^d \in \mathcal{X}^d$ and $\boldsymbol{x}^c \in \mathcal{X}^c$, respectively. Furthermore, consider a set F of subsets of both discrete and continuous variables/vertices, i.e., $\forall f \in F : f = (f^d \cup f^c) \subseteq V, \emptyset \neq f^d \subseteq V_d, \emptyset \neq f^c \subseteq V_c$, where each subset corresponds to the domain of one of the factors.

Now assume that we are given the following functions on these factors as well as on all discrete/continuous variables:

- A factor $\theta^d(\boldsymbol{x}^d), \theta^d : \mathcal{X}^d \to \mathbb{R}$ on all discrete variables
- A factor $\theta^c(\boldsymbol{x}^c), \theta^c : \mathcal{X}^c \to \mathbb{R}$ on all continuous variables
- $|F|$ mixed factors $\theta_f^m(\boldsymbol{x}_f^d, \boldsymbol{x}_f^c), \theta_f^m : \mathcal{X}_f^d \times \mathcal{X}_f^c \to \mathbb{R}$ on subsets $f \in F$ of both discrete and continuous variables, where $\boldsymbol{x}_f^d \in \mathcal{X}_f^d$ and $\boldsymbol{x}_f^c \in \mathcal{X}_f^c$ respectively denote subvectors of $\boldsymbol{x}^d$ and $\boldsymbol{x}^c$ from the (typically low-dimensional) subspaces $\mathcal{X}_f^d \subseteq \mathcal{X}^d$ and $\mathcal{X}_f^c \subseteq \mathcal{X}^c$, indexed by the vertices contained in f

The goal of our MAP problem is to find and assignment to all variables $\boldsymbol{x}^d$ and $\boldsymbol{x}^c$ which maximizes the sum of the factors:

$$\mathrm{MAP}(\boldsymbol{\theta}) = \max_{\boldsymbol{x}} \left\{ \theta^d(\boldsymbol{x}^d) + \theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \theta_f^m(\boldsymbol{x}_f^d, \boldsymbol{x}_f^c) \right\} \tag{6.1}$$

We now slightly reformulate this problem by duplicating the variables x_i^d and x_j^c, once for each mixed factor $\theta_f^m(\boldsymbol{x}_f^d, \boldsymbol{x}_f^c)$, and then enforce that these variables are equal to the ones appearing in the factors $\theta^d(\boldsymbol{x}^d)$ and $\theta^c(\boldsymbol{x}^c)$, respectively. Let x_i^{df} and x_j^{cf} respectively denote the copy of x_i^d and x_j^c used by factor f. Moreover, denote by $\boldsymbol{x}_f^{df} = \{x_i^{df}\}_{i \in f^d}$ and $\boldsymbol{x}_f^{cf} = \{x_j^{cf}\}_{j \in f^c}$ the set of variables used by factor f, and by $\boldsymbol{x}^F = \{\boldsymbol{x}_f^{df}, \boldsymbol{x}_f^{cf}\}_{f \in F}$ the set of all variable copies. We then get the equivalent (but now constrained) optimization problem

$$\max_{\boldsymbol{x}, \boldsymbol{x}^F} \left\{ \theta^d(\boldsymbol{x}^d) + \theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \theta_f^m(\boldsymbol{x}_f^{df}, \boldsymbol{x}_f^{cf}) \right\} \tag{6.2}$$

[1] In accordance with the notation in [SGJ11], we will here denote the factors by θ instead of f (i.e., in contrast to the main text).

$$\text{s.t.} \quad x_i^{df} = x_i^d, \quad \forall f \in F, i \in f^d$$
$$x_j^{cf} = x_j^c, \quad \forall f \in F, j \in f^c$$

To remove the coupling constraints, [SGJ11] now propose to use the technique of *Lagrangian relaxation* and introduce a Lagrange multiplier / dual variable $\lambda_{fi}(x_i)$ for every choice of $f \in F$, $i \in f$ and x_i (i.e. for every factor, for every variable in that factor, and for every value of that variable). These multipliers may then be interpreted as the *message* that factor f sends to variable i about its state x_i.

While this works well if all variables are discrete, in our model we also have continuous variables x_j^c, and it is clearly not possible to have a Lagrange multiplier for every possible value of x_j^c. To mitigate this issue, we follow [KPT11] and instead only introduce a multiplier λ_{fi} for every choice of $f \in F$ and $i \in f$, and model the interaction with the variables as $\lambda_{fi}(x_i) = \lambda_{fi} x_i$ (i.e., the product of a multiplier λ_{fi} and variable x_i). Observe that since our goal is to relax the coupling constraints, it is sufficient to introduce one multiplier per constraint. Since we have a constraint for every factor $f \in F$ and every discrete variable $i \in f^d$ and continuous variable $j \in f^c$ in that factor, our approach is clearly viable.

Note that in contrast to [KPT11], that introduces a set of multipliers for *every* factor / subgraph, we only introduce multipliers for the *mixed* factors $f \in F$. This is because in contrast to [KPT11], we do not introduce a full set of variable copies for *every* factor and then couple them to another global set of "original" variables, but we instead only introduce variable copies for the *mixed* factors and couple them to the variables appearing in the *discrete and continuous* factors, which we assume to be the "original" variables instead. This essentially is the same approach used in [SGJ11], with the difference that [SGJ11] introduce a singleton factor for each variable (i.e., a factor which depends only on a single variable), which they consider to be the "original" variable. They then simply couple the variable copies appearing in the *higher-order* factors to the "original" variables appearing in the *singleton* factors. In contrast, in our formulation we don't introduce singleton factors to model the "original" variables, but instead use the *fully discrete and continuous* factors for this purpose, which clearly works equally well. Note that as a result of this modeling choice, our optimization problem will be unconstrained, regardless of the number of factors, similar as in [SGJ11]. In contrast, [KPT11] end up with constraints enforcing that some of the dual variables sum to zero, since they are optimizing out the global set of "original" variables from their objective, while we keep the set of "original" variables within our discrete and continuous factors. For this reason, we will in contrast to [KPT11] later not require a projection step within the subgradient method used to optimize the dual; this is to be detailed further below.

For clarity, we treat discrete and continuous variables distinctly and for factor $f \in F$ denote λ_{fi}^d and λ_{fj}^c respectively for the Lagrange multipliers corresponding to its discrete variables $i \in f^d$ (or rather, the constraints $x_i^{df} = x_i^d$) and its continuous variables $j \in f^c$ (or rather, the constraints $x_j^{cf} = x_j^c$). For every factor $f \in F$, we furthermore aggregate its multipliers into the vectors $\boldsymbol{\lambda}_f^d = \{\lambda_{fi}^d\}_{i \in f^d} \in \mathbb{R}^{|\mathcal{X}_f^d|}$ and $\boldsymbol{\lambda}_f^c = \{\lambda_{fj}^c\}_{j \in f^c} \in \mathbb{R}^{|\mathcal{X}_f^c|}$. The set of all Lagrange multipliers is thus $\boldsymbol{\lambda} = \{\lambda_{fi}^d : f \in F, i \in f^d\} \cup \{\lambda_{fj}^c : f \in F, j \in f^c\} =$

$\{\boldsymbol{\lambda}_f^d, \boldsymbol{\lambda}_f^c\}_{f \in F}$. We then define the Lagrangian

$$
\begin{aligned}
L(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{x}^F) &= \theta^d(\boldsymbol{x}^d) + \theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \theta_f^m(\boldsymbol{x}_f^{df}, \boldsymbol{x}_f^{cf}) \\
&+ \sum_{f \in F} \sum_{i \in f^d} \lambda_{fi}^d \left(x_i^d - x_i^{df} \right) + \sum_{f \in F} \sum_{j \in f^c} \lambda_{fj}^c \left(x_j^c - x_j^{cf} \right) \\
&= \left(\theta^d(\boldsymbol{x}^d) + \sum_{f \in F} \sum_{i \in f^d} \lambda_{fi}^d x_i^d \right) \\
&+ \left(\theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \sum_{j \in f^c} \lambda_{fj}^c x_j^c \right) \\
&+ \sum_{f \in F} \left(\theta_f^m(\boldsymbol{x}_f^{df}, \boldsymbol{x}_f^{cf}) - \sum_{i \in f^d} \lambda_{fi}^d x_i^{df} - \sum_{j \in f^c} \lambda_{fj}^c x_j^{cf} \right) .
\end{aligned}
$$

This results in the following optimization problem:

$$
\begin{aligned}
&\max_{\boldsymbol{x}, \boldsymbol{x}^F} L(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{x}^F) \\
\text{s.t.} \quad & x_i^{df} = x_i^d, \quad \forall f \in F, i \in f^d \\
& x_j^{cf} = x_j^c, \quad \forall f \in F, j \in f^c
\end{aligned}
\tag{6.3}
$$

Note that the problem in Eq. (6.3) is still equivalent to our (hard) original problem in Eq. (6.1) for any assignment of $\boldsymbol{\lambda}$, since the Lagrange multipliers cancel out if all coupling constraints are fulfilled.

To obtain a tractable problem, we thus simply omit the coupling constraints in Eq. (6.3) and define the dual function $L(\boldsymbol{\lambda})$ as

$$
\begin{aligned}
L(\boldsymbol{\lambda}) &= \max_{\boldsymbol{x}, \boldsymbol{x}^F} L(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{x}^F) \\
&= \max_{\boldsymbol{x}^d} \left(\theta^d(\boldsymbol{x}^d) + \sum_{f \in F} \sum_{i \in f^d} \lambda_{fi}^d x_i^d \right) \\
&+ \max_{\boldsymbol{x}^c} \left(\theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \sum_{j \in f^c} \lambda_{fj}^c x_j^c \right) \\
&+ \sum_{f \in F} \max_{\boldsymbol{x}_f^{df}, \boldsymbol{x}_f^{cf}} \left(\theta_f^m(\boldsymbol{x}_f^{df}, \boldsymbol{x}_f^{cf}) - \sum_{i \in f^d} \lambda_{fi}^d x_i^{df} - \sum_{j \in f^c} \lambda_{fj}^c x_j^{cf} \right)
\end{aligned}
$$

Note that the maximizations are now fully independent, such that we can (without introducing any ambiguity) simplify the notation for the variables involved in the mixed terms to denote $\boldsymbol{x}_f^d$ and $\boldsymbol{x}_f^c$ instead of $\boldsymbol{x}_f^{df}$ and $\boldsymbol{x}_f^{cf}$, respectively[2], resulting in the slightly

[2] I.e., we replace all variable copies $\boldsymbol{x}_f^{df}, \boldsymbol{x}_f^{cf}$ in the mixed terms by the "original" variables $\boldsymbol{x}_f^d, \boldsymbol{x}_f^c$.

simpler dual formulation

$$
\begin{aligned}
L(\boldsymbol{\lambda}) = &\max_{\boldsymbol{x}^d} \left(\theta^d(\boldsymbol{x}^d) + \sum_{f \in F} \sum_{i \in f^d} \lambda^d_{fi} x^d_i \right) \\
&+ \max_{\boldsymbol{x}^c} \left(\theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \sum_{j \in f^c} \lambda^c_{fj} x^c_j \right) \\
&+ \sum_{f \in F} \max_{\boldsymbol{x}^d_f, \boldsymbol{x}^c_f} \left(\theta^m_f(\boldsymbol{x}^d_f, \boldsymbol{x}^c_f) - \sum_{i \in f^d} \lambda^d_{fi} x^d_i - \sum_{j \in f^c} \lambda^c_{fj} x^c_j \right)
\end{aligned}
$$

Let $\boldsymbol{x}^d_{|f} \in \mathcal{X}^d_f$ and $\boldsymbol{x}^c_{|f} \in \mathcal{X}^c_f$ respectively denote the subvectors of $\boldsymbol{x}^d$ and $\boldsymbol{x}^c$ containing only the variables of factor f. The shorthands (or *reparameterizations* [SGJ11])

$$
\begin{aligned}
\bar{\theta}^{\boldsymbol{\lambda}}_d(\boldsymbol{x}^d) &= \theta^d(\boldsymbol{x}^d) + \sum_{f \in F} \sum_{i \in f^d} \lambda^d_{fi} x^d_i \\
&= \theta^d(\boldsymbol{x}^d) + \sum_{f \in F} \boldsymbol{\lambda}^d_f \boldsymbol{x}^d_{|f} && (6.4) \\
\bar{\theta}^{\boldsymbol{\lambda}}_c(\boldsymbol{x}^c) &= \theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \sum_{j \in f^c} \lambda^c_{fj} x^c_j \\
&= \theta^c(\boldsymbol{x}^c) + \sum_{f \in F} \boldsymbol{\lambda}^c_f \boldsymbol{x}^c_{|f} && (6.5) \\
\bar{\theta}^{\boldsymbol{\lambda}}_f(\boldsymbol{x}^d_f, \boldsymbol{x}^c_f) &= \theta^m_f(\boldsymbol{x}^d_f, \boldsymbol{x}^c_f) - \sum_{i \in f^d} \lambda^d_{fi} x^d_i - \sum_{j \in f^c} \lambda^c_{fj} x^c_j \\
&= \theta^m_f(\boldsymbol{x}^d_f, \boldsymbol{x}^c_f) - \boldsymbol{\lambda}^d_f \boldsymbol{x}^d_f - \boldsymbol{\lambda}^c_f \boldsymbol{x}^c_f && (6.6)
\end{aligned}
$$

further simplify the dual function $L(\boldsymbol{\lambda})$ to

$$
L(\boldsymbol{\lambda}) = \max_{\boldsymbol{x}^d} \bar{\theta}^{\boldsymbol{\lambda}}_d(\boldsymbol{x}^d) + \max_{\boldsymbol{x}^c} \bar{\theta}^{\boldsymbol{\lambda}}_c(\boldsymbol{x}^c) + \sum_{f \in F} \max_{\boldsymbol{x}^d_f, \boldsymbol{x}^c_f} \bar{\theta}^{\boldsymbol{\lambda}}_f(\boldsymbol{x}^d_f, \boldsymbol{x}^c_f) \ . \tag{6.7}
$$

First, observe that since we maximize over $\boldsymbol{x}$ and $\boldsymbol{x}^F$, the dual function $L(\boldsymbol{\lambda})$ is a function of just the Lagrange multipliers $\boldsymbol{\lambda}$. Note that since $L(\boldsymbol{\lambda})$ maximizes over a larger space (since instead of forcing that there must be one global assignment maximizing the objective, we allow the discrete/continuous potentials to be maximized independently of the mixed potentials, meaning that $\boldsymbol{x}$ may not coincide with $\boldsymbol{x}^F$), we have for all $\boldsymbol{\lambda}$ that

$$
\mathrm{MAP}(\boldsymbol{\theta}) \leq L(\boldsymbol{\lambda}) \ . \tag{6.8}
$$

The *dual problem* now is to find the tightest upper bound by optimizing the Lagrange multipliers, i.e.

$$
\min_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda}) \tag{6.9}
$$

We also call the dual problem in Eq. (6.9) the *master problem*, which coordinates the

$2 + |F|$ *slave problems* (i.e., one for each factor)

$$s^d(\boldsymbol{\lambda}) = \max_{\boldsymbol{x}^d} \bar{\theta}_d^{\boldsymbol{\lambda}}(\boldsymbol{x}^d) \tag{6.10a}$$

$$s^c(\boldsymbol{\lambda}) = \max_{\boldsymbol{x}^c} \bar{\theta}_c^{\boldsymbol{\lambda}}(\boldsymbol{x}^c) \tag{6.10b}$$

$$s^f(\boldsymbol{\lambda}) = \max_{\boldsymbol{x}_f^d, \boldsymbol{x}_f^c} \bar{\theta}_f^{\boldsymbol{\lambda}}(\boldsymbol{x}_f^d, \boldsymbol{x}_f^c), \quad \forall f \in F \ . \tag{6.10c}$$

where we refer to s^d, s^c and s^f as the discrete slave, the continuous slave, and the mixed slaves, respectively. Using the notation in Eqs. (6.10a)-(6.10c), the dual function further simplifies to

$$L(\boldsymbol{\lambda}) = s^d(\boldsymbol{\lambda}) + s^c(\boldsymbol{\lambda}) + \sum_{f \in F} s^f(\boldsymbol{\lambda}) \ . \tag{6.11}$$

Intuitively, the goal of Eq. (6.9) is as follows: The master problem wants the discrete/continuous slaves to agree with the mixed slaves/factors in which the corresponding discrete/continuous variables appear, and conversely, it wants the mixed slaves to agree with the slaves/factors of the discrete/continuous variables in its scope. The master problem will thus incentivize the discrete/continuous slaves and the mixed slaves to agree with each other, which is done by updating the dual variables $\boldsymbol{\lambda}$ accordingly.

The key property of the function $L(\boldsymbol{\lambda})$ is that it only involves maximization over local assignments of $\boldsymbol{x}^d, \boldsymbol{x}^c$ and $\boldsymbol{x}_f^d, \boldsymbol{x}_f^c$, which are tasks we assume to be tractable. The dual thus decouples the original problem, resulting in a problem that can be optimized using local operations. Algorithms that minimize the approximate objective $L(\boldsymbol{\lambda})$ use local updates where each iteration of the algorithms repeatedly finds a maximizing assignment for the subproblems individually, using these to update the dual variables $\boldsymbol{\lambda}$ that glue the subproblems together. There are two main classes of algorithms of this kind, one based on a subgradient method and another based on block coordinate descent [SGJ11].

CHAPTER 7

Experiments for Chapter 5

7.0.1 Synthetic benchmark for unconstrained optimization

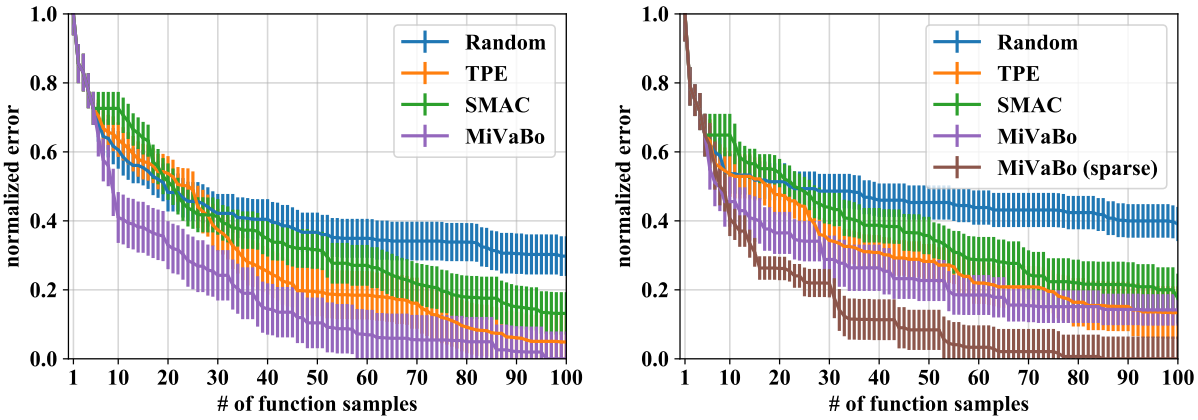


Figure 7.1: Results on the synthetic benchmark, with the Gaussian (left) and Laplace prior (right). Mean plus/minus one standard deviation of the normalized error over 16 random initializations. (Left) MiVaBo outperforms its competitors. (Right) MiVaBo with a sparse prior outperforms its competitors, including MiVaBo with a Gaussian prior

We assess the performance on an unconstrained synthetic linear benchmark function of the form $f(\boldsymbol{x}) = \omega^\top \boldsymbol{\phi}$. We choose a fairly high-dimensional objective with $D_d = 8$ discrete and $D_c = 8$ continuous variables, thus resulting in a total input space dimensionality of $D = D_d + D_c = 16$. For the discrete model part, we choose the $M_d \in \mathcal{O}(D_d^2)$ features $\boldsymbol{\phi}^d$ proposed in Section 5.2.1. For the continuous features $\boldsymbol{\phi}^c$, we choose $M_c = 16$ dimensional Random Fourier Features to approximate a GP with a squared exponential kernel with bandwidth $\sigma = 1.0$. For the mixed representation, we construct a feature vector $\boldsymbol{\phi}^m$ by stacking all pairs of discrete and continuous features, as proposed in Section 5.2.1. We consider two settings for the weight vector: Firstly, we sample it from a zero-mean Gaussian, $\omega \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^M$. Secondly, we sample it from a Laplace distribution, i.e. $\omega \sim p(\omega|\alpha) \propto \exp(-\alpha^{-1}\|\omega\|_1)$, with inverse scale parameter $\alpha = 0.1$, and then prune all weights smaller than 10 to zero to induce sparsity over the weight vector. For the second setting, we also assess MiVaBo using a Laplace prior and the approximate inference technique from [SN11] (see also Section 6.1)[1]. As we do not know the true optimum of the

[1]We use the MATLAB implementation provided in the `glm-ie` toolbox [Nic12] by the same authors.

function and thus cannot compute the regret, we normalize all observed function values to the interval $[0, 1]$, resulting in a normalized error as the metric of comparison. We can observe from our results shown in Figure 7.1 that MiVaBo outperforms the competing methods in this setting, demonstrating the effectiveness of our approach when its modeling assumptions are fulfilled.

7.0.2 Synthetic benchmark for constrained optimization

In another experiment, we demonstrate the capability of our algorithm to incorporate linear constraints on the discrete variables. In particular, we want to enforce a solution that is sparse in the discrete variables via adding a hard cardinality constraint of the type $\sum_{i=1}^{D_d} x_i^d \leq k$, which we can simply specify in the Gurobi optimizer. Cardinality constraints of this type are very relevant in practice, as many real-world problems desire sparse solutions (e.g., sparsification of ising models, contamination control, aero-structural multi-component problems [BP18]). We consider the same functional form as before, i.e. again with $D_d = D_c = 8$, and set $k = 2$, meaning that a solution should have at most two of our binary variables set to one, while all others shall be set to zero. To enable comparison with TPE, SMAC and random search, which provide no capability of modeling these kinds of constraints, we assume the objective f to be unconstrained, but instead return a large penalty value if a method acquires an evaluation of f at a point that violates the constraint. Thus, the baseline algorithms are forced to learn the constraint from observations, which is a challenging problem.

One can notice from Figure 7.2 that the ability to explicitly encode the cardinality constraint into the discrete optimization oracle significantly increases performance.

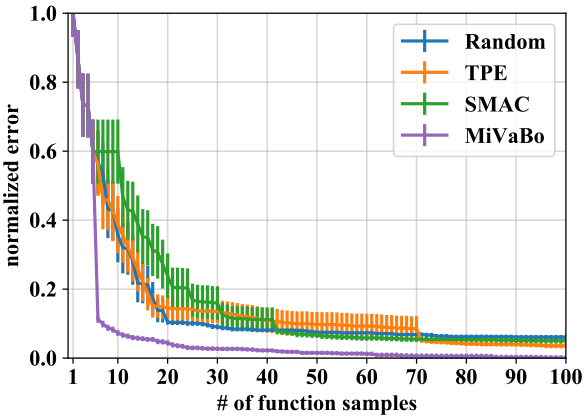


Figure 7.2: Results on the synthetic benchmark with cardinality constraints. The curves represent the mean plus/minus one standard deviation of the normalized error over 16 random initializations. One can observe that MiVaBo outperforms its competitors.

7.1 More details on XGBoost hyperparameter tuning

Please refer to the corresponding websites for details on the OpenML XGBoost benchmark [2], on the underlying implementation [3], and on the `steel-plates-fault` [4] and `monks-problem-1` [5] datasets. Finally, see Table 7.1 for a description of the hyperparameters involved in XGBoost.

Name	Type	Domain
booster	discr.	['gbtree', 'gblinear']
nrounds	discr.	$[3, 5000]$
alpha	contin.	$[0.000985, 1009.209690]$
lambda	contin.	$[0.000978, 999.020893]$
colsample_bylevel	contin.	$[0.046776, 0.998424]$
colsample_bytree	contin.	$[0.062528, 0.999640]$
eta	contin.	$[0.000979, 0.995686]$
max_depth	discr.	$[1, 15]$
min_child_weight	contin.	$[1.012169, 127.041806]$
subsample	contin.	$[0.100215, 0.999830]$

Table 7.1: Hyperparameters of the XGBoost algorithm. 10 parameters, 7 of which are continuous, and 3 of which are discrete.

7.2 More details on VAE hyperparameter tuning

7.2.1 Hyperparameters of VAE

We used the `PyTorch` library to implement the VAE used in the experiment. Table 7.2 describes the names, types and domains of the involved hyperparameters that we tune. Whenever we refer to a "deconvolutional layer" (also called transposed convolution or fractionally-strided convolution), we mean the functional mapping implemented by a `ConvTranspose2d` layer in `PyTorch`[6]. Since our approach operates on a binary encoding of the discrete parameters, we also display the number of bits required to encode each discrete parameter. In total, we consider 25 discrete parameters (resulting in 50 when binarized) as well as three continuous ones.

7.2.2 Description of constraints

We now describe the constraints arising from the mutual dependencies within the hyperparameter space of the deconvolutional VAE (as described in Section 7.2.1).

[2] https://www.openml.org/f/6767
[3] https://www.rdocumentation.org/packages/xgboost/versions/0.6-4
[4] https://www.openml.org/t/9967
[5] https://www.openml.org/t/146064
[6] See https://pytorch.org/docs/stable/nn.html#convtranspose2d for details.

#	Name	Type	Domain	Bits
1	Number of conv. layers in encoder	discrete	[0,1,2]	2
	Parameters of `C1`			
2	Number of channels of `C1`	discrete	[4,8,16,24]	2
3	Stride of `C1`	discrete	[1,2]	1
4	Filter size of `C1`	discrete	[3,5]	1
5	Padding of `C1`	discrete	[0,1,2,3]	2
	Parameters of `C2`			
6	Number of channels of `C2`	discrete	[8,16,32,48]	2
7	Stride of `C2`	discrete	[1,2]	1
8	Filter size of `C2`	discrete	[3,5]	1
9	Padding of `C2`	discrete	[0,1,2,3]	2
10	Number of fc. layers in encoder	discrete	[0,1,2]	2
11	Number of units of `F1`	discrete	$[0 \dots 960]$	4
12	Dimensionality d_z of `z`	discrete	$[16 \dots 64]$	6
13	Number of fc. layers in decoder	discrete	[0,1,2]	2
14	Number of units of `F4`	discrete	$[0 \dots 960]$	4
15	Number of deconv. layers in decoder	discrete	[0,1,2]	2
	Parameters of `D1`			
16	Number of channels of `D1`	discrete	[8,16,32,48]	2
17	Stride of `D1`	discrete	[1,2]	1
18	Filter size of `D1`	discrete	[3,5]	1
19	Padding of `D1`	discrete	[0,1,2,3]	2
20	Output padding of `D1`	discrete	[0,1,2,3]	2
	Parameters of `D2`			
21	Number of channels of `D2`	discrete	[4,8,16,24]	2
22	Stride of `D2`	discrete	[1,2]	1
23	Filter size of `D2`	discrete	[3,5]	1
24	Padding of `D2`	discrete	[0,1,2,3]	2
25	Output padding of `D2`	discrete	[0,1,2,3]	2
26	Learning rate	continuous	$[10^{-4}, 10^{-2}]$	-
27	Learning rate decay factor	continuous	$[0.5, 1.0]$	-
28	Weight decay regularization	continuous	$[10^{-6}, 10^{-2}]$	-
	Total			**50**

Table 7.2: Hyperparameters of the VAE. The architecture of the VAE (if all layers are enabled) is `C1-C2-F1-F2-z-F3-F4-D1-D2`, with `C` denoting a convolutional (conv.) layer, `F` a fully-connected (fc.) layer, `D` a deconvolutional (deconv.) layer and `z` the latent space. Layers `F2` and `F3` have fixed sizes of $2d_z$ and d_z units respectively, where d_z denotes the dimensionality of the latent space `z`. The domain of the number of units of the fc. layers `F1` and `F4` is discretized with a step size of 64, i.e. $[0, 64, 128, \dots, 832, 896, 960]$, denoted by $[0 \dots 960]$ in the table for brevity. For d_z, the domain $[16 \dots 64]$ refers to all integers within that interval.

Encoder constraints. For the convolutional layers (up to two in our case) of the encoder, we need to ensure that the chosen combination of stride, padding and filter size transforms the input image into an output image whose shape is integral (i.e., not fractional). More precisely, denoting the input image size by W_{in} (i.e., the input image is quadratic with shape $W_{\text{in}} \times W_{\text{in}}$), the stride by S, the filter size by F, and the padding by P, we need to

ensure that the output image size W_{out} is integral, i.e.

$$W^e_{\text{out}} = (W^e_{\text{in}} - F^e + P^e)/S^e + 1 \in \mathbb{N} \tag{7.1}$$

where superscripts e are used to make clear that we are considering the encoder. Let us illustrate this with an example[7]: For $W_{\text{in}} = 10$, $P = 0$, $S = 2$ and $F = 3$, we would get an invalid fractional output size of $W_{\text{out}} = (10 - 3 + 0)/2 + 1 = 4.5$. To obtain a valid output size, one could, e.g., instead consider a padding of $P = 1$, yielding $W_{\text{out}} = (10 - 3 + 1)/2 + 1 = 5$. Alternatively, one could also consider a stride of $S = 1$ to obtain $W_{\text{out}} = (10 - 3 + 0)/1 + 1 = 8$, or a filter size of $F = 4$ to obtain $W_{\text{out}} = (10-4+0)/2+1 = 4$ (though the latter is very uncommon and thus not allowed in our setting; we only allow $F \in \{3, 5\}$, as described in Section 7.2.1). While this constraint is not trivially fulfilled (which can be verified by manually trying different configurations of W_{in}, F, S, P), it is also not too challenging to find valid configurations.

Note that this constraint is required to be fulfilled for every convolutional layer; we thus obtain the following two constraints in our specific two-layer setting, where $W_{\text{in}} = 28$ (as `MNIST` and `FashionMNIST` images are of shape 28×28):

$$W^e_{\text{out1}} = (28 - F^e_1 + P^e_1)/S^e_1 + 1 \qquad \in \mathbb{N}, \tag{7.2}$$

$$W^e_{\text{out2}} = (W^e_{\text{out1}} - F^e_2 + P^e_2)/S^e_2 + 1 \qquad \in \mathbb{N}. \tag{7.3}$$

where the subscripts in $\{1, 2\}$ denote the index of the convolutional layer.

Finally, observe that the constraints in Eq. (7.2) and Eq. (7.3) are, respectively, linear and quadratic in the discrete variables $F^e_1, F^e_2, P^e_1, P^e_2, S^e_1, S^e_2$, and can thus be readily incorporated into the integer programming solver (e.g. Gurobi [Opt14] or CPLEX [IBM09]) we employ as a subroutine within our acquisition function optimization strategy.

Decoder constraints. While the constraints on the decoder architecture are similar in nature to those for the encoder, they are significantly more difficult to fulfill, which we will now illustrate.

In particular, we need to ensure that the decoder produces images of shape 28×28. By inverting the formula in Eq. (7.1), we see that for a deconvolutional layer (which intuitively implements an inversion of the convolution operation), the output image size W_{out} can be computed as

$$W^d_{\text{out}} = (W^d_{\text{in}} - 1) \times S^d + F^d - 2P^d + O^d \tag{7.4}$$

where superscripts d are used to make clear that we are considering the decoder, and where O is an additional output padding parameter which can be used to adjust the shape of the output image[8]. Note that we now have a factor of $2P$ in Eq. (7.4) instead of

[7] This example is taken from http://cs231n.github.io/convolutional-networks/#conv (paragraph "Constraints on strides"), which also describes the constraints discussed here. Note that they define the padding P in a slightly different way (i.e., they only consider symmetric padding, while we also allow for asymmetric padding) and thus end up with a term of $2P$ instead of P in the formula.

[8] See e.g. https://pytorch.org/docs/stable/nn.html#convtranspose2d for a description of the output

P (as for the encoder, i.e. in Eq. (7.1)), since we only consider symmetric padding for the decoder, while we allow for asymmetric padding for the encoder (to make it easier to fulfill the integrality constraints for the encoder due to an increased number of valid configurations). The output padding parameter O is required since the mapping from W^e_{in} to W^e_{out} in a convolutional layer (i.e. in the encoder) is not bijective: there are different combinations of W^e_{in}, F, S, P that result in the same W^e_{out} (which can be easily verified). Thus, given an output size W^e_{out} (now serving as the input size W^d_{in} of the deconvolutional layer), there is no unique corresponding input size W^e_{in} (now serving as the output size W^d_{out} of the deconvolutional layer). The output padding parameter O can thus be used to disambiguate this relation. Note that W^d_{out} in Eq. (7.4) is always integral, so there are no integrality constraints involved here, in constrast to the encoder.

In the context of our decoder model, i.e. with up to two deconvolutional layers, and with a required output image size of 28, we thus obtain the following constraints:

$$W^d_{\text{out}} = (W^d_{\text{in}} - 1) \times S^d_1 + F^d_1 - 2P^d_1 + O^d_1, \tag{7.5}$$

$$28 = (W^d_{\text{out}} - 1) \times S^d_2 + F^d_2 - 2P^d_2 + O^d_2, \tag{7.6}$$

i.e. we need to choose the parameters $F^d_1, F^d_2, P^d_1, P^d_2, S^d_1, S^d_2, O^d_1, O^d_2$ such that the output size is 28, which is challenging, as only a small number of parameter configurations fulfill this property. While this problem is already challenging when assuming a given fixed input image shape W^d_{in}, in our setting it is more difficult, as W^d_{in} has to be of a suitable size as well. Note that W^d_{in} is determined by the size of the fully-connected layer preceding the first deconvolutional layer, which yields an additional challenge: the size of the last fully-connected layer has to be set such that it can be resized to an image of shape $C^d_1 \times W^d_{\text{in}} \times W^d_{\text{in}}$ (i.e., such that it can be fed into a deconvolutional layer), where C^d_1 denotes the number of channels of the first deconvolutional layer of the decoder. As the resulting problem would be too challenging for any algorithm to produce a valid solution in a reasonable amount of time, we simplify it slightly by only treating C^d_1 as a design parameter (as described in Section 7.2.1), but keeping $W^d_{\text{in}} = 7$ fixed. The value 7 is chosen since $16 \times 7 \times 7 = 784$, i.e., when setting $C^d_1 = 16$, the last fully-connected layer has the correct output shape (since $28 \times 28 = 784$ for an `MNIST` and `FashionMNIST` image). This way, a valid decoder architecture can be achieved by deactivating all convolutional layers and choosing $C^d_1 = 16$, constituting an alternative if fulfilling the decoder constraints in Eq. (7.5) and Eq. (7.6) is too challenging for an algorithm.

Finally, the constraints in Eq. (7.5) and Eq. (7.6) are, respectively, linear and quadratic in the discrete variables $F^d_1, F^d_2, P^d_1, P^d_2, S^d_1, S^d_2, O^d_1, O^d_2$, which again allows us to incorporate them into our optimization routine.

7.2.3 Effect of different constraint violation penalty values

We now analyze the effect of the constraint violation penalty value on the performance of SMAC, TPE and GPyOpt. Note that random search and MiVaBo are not affected by the penalty. We do this analysis to show that the choice of penalty does not qualitatively affect

padding in the context of the `PyTorch` library we use.

the reported results. In addition to the penalty of 500 nats considered in the experiments in the main paper, we assessed two smaller alternative penalties of 250 nats and 125 nats, respectively. The results in Table 7.3 show that the performance of the methods improves marginally with decreasing penalty values. This can be intuitively explained by the fact that the smaller the penalty, the smaller the region in hyperparameter space that the penalty discourages from searching. In fact, a large penalty may not only discourage infeasible configurations, but also feasible configurations that lie "close" to the penalized infeasible one (where closeness is defined by the specific surrogate model employed by the method). However, even for the smallest penalty of 125 nats, SMAC, TPE and GPyOpt still perform worse than random search, and thus still significantly worse than MiVaBo. Imposing penalties that are significantly smaller than 125 is not sensible, as this will encourage the model-based methods to violate the constraints, and in turn discourage them from ever evaluating a valid configuration (as this would yield a worse score).

Finally, Table 7.4 shows the number of constraint violations by the different methods, depending on the violation penalty.

Algorithm	**Penalty (nats)**		
	500	**250**	**125**
SMAC	113.0 ± 1.8	112.1 ± 1.8	111.1 ± 1.6
TPE	108.8 ± 1.2	108.1 ± 1.3	108.1 ± 1.3
GPyOpt	108.5 ± 1.1	108.5 ± 0.6	106.5 ± 1.4
RS			106.3 ± 0.9
MiVaBo			$\mathbf{94.4 \pm 0.8}$

Table 7.3: Mean plus/minus one standard deviation of the negative test log-likelihood over 8 random initializations, achieved by the best VAE configuration found by SMAC, TPE and GPyOpt after 16 BO iterations, for constraint violation penalties of 500, 250 and 125 nats. Performance values of MiVaBo and random search (which are not affected by the penalty) are included for reference.

Algorithm	**Penalty (nats)**		
	500	**250**	**125**
SMAC	37 ± 21.7	36 ± 21.9	28 ± 11.6
TPE	67 ± 21.3	68 ± 22.2	68 ± 22.2
GPyOpt	36 ± 19.3	32 ± 18.0	27 ± 10.4
Random search	71 ± 25.5	71 ± 25.5	71 ± 25.5

Table 7.4: Mean plus/minus one standard deviation of the number of constraint violations by SMAC, TPE, GPyOpt and random search within 16 BO iterations over 8 random initializations, for constraint violation penalties of 500, 250 and 125 nats.

7.2.4 Visualization of reconstruction quality

While log-likelihood scores allow for a principled quantitative comparison between different algorithms, they are typically hard to interpret for humans. We thus in Figure 7.3 visualize the reconstruction quality achieved by the best VAE configuration found by the different

methods after 32 BO iterations. The VAEs were trained for 32 epochs each (as in the BO experiments). The log-likelihood scores seem to be correlated with quality of visual appearance, and the model found by MiVaBo thus may be perceived to produce the visually most appealing reconstructions among all models.

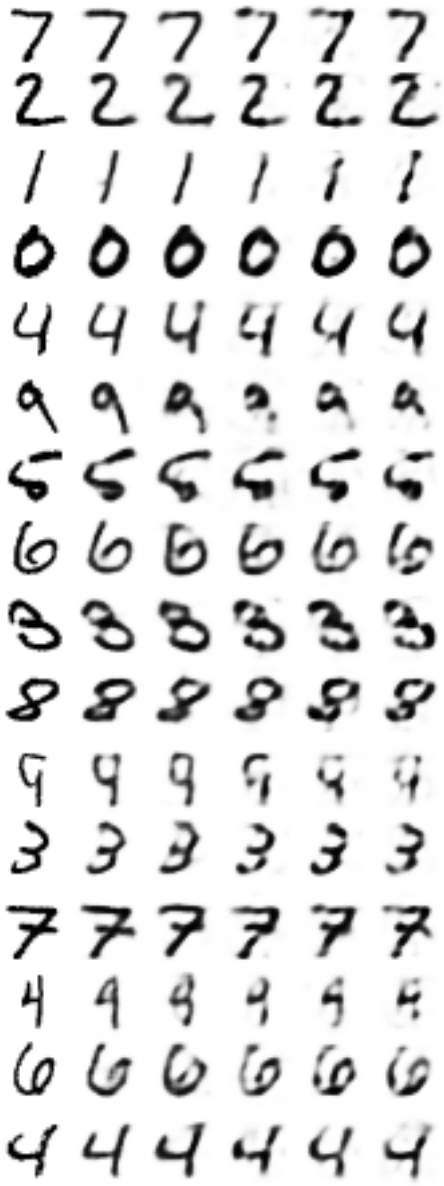


Figure 7.3: Visualization of the reconstruction quality of a random subset of (non-binarized) images from the `MNIST` test set, as achieved by the best VAE model (trained for 32 epochs) found by each method. From left to right: ground truth, MiVaBo, random search, GPyOpt, TPE and SMAC. The images are thus ordered (from left to right) by increasing the negative test log-likelihood achieved by the VAEs used for reconstruction. Interestingly, the log-likelihood seems to capture the quality of visual appearance, as the reconstruction quality may be roughly perceived to decrease from left to right.

7.3 Discussion and details on baselines in empirical evaluation

We decided to compare against SMAC [HHL11] and TPE [Ber+11b], as these are state-of-the-art mixed-variable BO methods. We used their publicly available `Python` implementations under `https://github.com/automl/SMAC3` (SMAC) and `https://github.com/hyperopt/hyperopt` (TPE). We furthermore compare against the popular GPyOpt BO `Python` package [Gon16] (`https://github.com/SheffieldML/GPyOpt`) as a reference implementation of a state-of-the-art continuous BO method (which extends to the mixed-variable setting via relaxation and rounding of the discrete variables). We use these `Python` packages with their respective default settings. Moreover, to isolate the benefit of the model choice from the acquisition function optimization procedure, we consider baselines that, respectively, use the MiVaBo and GP models, and optimize the resulting acquisition function using simulated annealing (SA) [KGV83]. For the baseline that combines a GP with simulated annealing, we use the popular GPy `Python` package [GPy12] (which also serves as the GP backend of GPyOpt) together with the simulated annealing implementation at `https://github.com/perrygeo/simanneal`. Finally, we compare against random search (using a custom implementation due to its simplicity), which has been shown to be an effective baseline for hyperparameter optimization [Ber+11b].

There are several other methods that address problem settings related to the (constrained) mixed-variable paradigm we consider. We here briefly clarify why we decided to not compare against them in our empirical evaluation. Firstly, [BP18; Oh+19; Kim+19] extend BO to tackle purely discrete/combinatorial problems; these approaches can thus not straightforwardly handle the continuous variables present in mixed-variable problems. [Ru+19] address BO problems with multiple continuous and *categorical* input variables (i.e. unordered ones), whereas MiVaBo includes ordered discrete variables such as integer variables. As pointed out in the introduction, Hyperband [Li+18] and BOHB [FKH18] are complementary to MiVaBo in that they do not propose new mixed-variable methods, but rather extend existing ones (such as random search and TPE) to the multi-fidelity setting; they should thus not be perceived as competing methods. The work of [GH18] extends GP-based BO to integer variables, but cannot handle discrete constraints. While several works [Her+15; Gar+14; Sui+15] propose extensions of continuous BO methods to handle unknown constraints, they can neither handle mixed-variable problems nor known (discrete) constraints, and might thus again be viewed as complementary to our approach.

Finally, a recent line of work extends continuous BO methods to general highly structured input spaces such as graphs or images (which also includes mixed discrete-continuous problems), by first training a deep generative model such as a VAE on the input data, and then using standard continuous BO methods in the continuous latent space learned by the VAE [Góm+18]. This so-called *latent space optimization* approach has recently been successfully applied to application domains including automatic chemical design and automatic machine learning [Góm+18; KPH17; Ngu+16; Luo+18; Lu+18; JBJ18; TDH20], and might thus be perceived to be a promising method for the mixed-variable hyperparameter tuning tasks we consider in this paper. However, despite these successes, the latent space optimization paradigm is at an early stage and current methods still suffer from critical shortcomings. One of the most severe issues is that the BO

procedure tends to progress into regions of the latent space that are too far away from the regions corresponding to the training data, which often results in the BO method suggesting meaningless or even invalid inputs to query (e.g. unreasonable/invalid hyperparameter configurations). Despite recent efforts attempting to mitigate this issue [KPH17; GH20; Dai+18; DH19; MH19], a robust and principled solution has yet to be found. This issue also reveals that the latent space optimization paradigm makes it difficult to incorporate (discrete) constraints, as the optimization is performed in a learned continuous latent space rather than in the original input space (over which the constraints are defined). As a result, we decided to not compare against latent space optimization methods at this stage, although we point out that this would be an interesting direction for future work.

7.4 Acquisition function optimization with theoretical guarantees via dual decomposition

As an alternative to the alternating optimization scheme proposed in Section 5.2.3, one can also minimize the acquisition function in Eq. (5.4) via dual decomposition - a powerful approach based on Lagrangian optimization, which has well-studied theoretical properties and has been successfully used for many different problems [KPT11; SGJ11; RC12]. Despite its versatility, the core idea is simple: decompose the initial problem into smaller solvable subproblems and then extract a solution by cleverly combining the solutions from these subproblems [KPT11]. This requires the following two components: (1) A set of *subproblems* which are defined such that their sum corresponds to the optimization objective, and which can each be optimized globally, and (2) a so-called *master* problem that coordinates the subproblems to find a solution to the original problem. One major advantage of dual decomposition algorithms is that they have well-understood theoretical properties[9], in particular through connections to linear programming (LP) relaxations. In fact, they enjoy the best theoretical guarantees in terms of convergence properties, when compared to other algorithms solving this problem [KPT11]. These theoretical properties further facilitate the convergence analysis of MiVaBo outlined in Section 5.2.5, making dual decomposition algorithms particularly useful for settings where optimization accuracy is of crucial importance.

We now describe how to devise a dual decomposition for our problem, by demonstrating how it can be reformulated in terms of master- and sub-problems (see Section 6.3 for a detailed derivation). For convenience, let us denote the discrete, continuous and mixed parts of Eq. (5.4) by $f^d(\boldsymbol{x}^d) = {\omega^d}^\top \boldsymbol{\phi}^d(\boldsymbol{x}^d)$, $f^c(\boldsymbol{x}^c) = {\omega^c}^\top \boldsymbol{\phi}^c(\boldsymbol{x}^c)$ and $f^m(\boldsymbol{x}^d, \boldsymbol{x}^c) = {\omega^m}^\top \boldsymbol{\phi}^m(\boldsymbol{x}^d, \boldsymbol{x}^c)$, respectively, thus resulting in the representation $f(\boldsymbol{x}) = f^d(\boldsymbol{x}^d) + f^c(\boldsymbol{x}^c) + f^m(\boldsymbol{x}^d, \boldsymbol{x}^c)$. First, we note that the discrete $f^d(\boldsymbol{x}^d)$ and continuous $f^c(\boldsymbol{x}^c)$ parts of Eq. (5.4) already represent easy to solve subproblems (as we assume to have access to an optimization oracle). It thus remains to discuss the mixed part $f^m(\boldsymbol{x}^d, \boldsymbol{x}^c)$. As f^m is generally difficult to optimize directly, we assume that it decomposes into a sum $f^m(\boldsymbol{x}) = \sum_{k=1}^{|F|} f_k^m(\boldsymbol{x}_k^d, \boldsymbol{x}_k^c)$ of so-called *factors* $f_k^m : \mathcal{X}_k^d \times \mathcal{X}_k^c \to \mathbb{R}$, where $\boldsymbol{x}_k^d \in \mathcal{X}_k^d$ and $\boldsymbol{x}_k^c \in \mathcal{X}_k^c$ respectively denote subvectors of $\boldsymbol{x}^d$ and $\boldsymbol{x}^c$ from the (typically low-

[9] For details, we refer the interested reader to [KPT11; SGJ11; RC12]

dimensional) subspaces $\mathcal{X}_k^d \subseteq \mathcal{X}^d$ and $\mathcal{X}_k^c \subseteq \mathcal{X}^c$. Here, F denotes a set of subsets $k \in F$ of the variables. Given this formulation, the initial problem then reduces to the minimization of the dual function $L(\boldsymbol{\lambda})$ w.r.t. Lagrange multipliers $\boldsymbol{\lambda}$, i.e., the master problem $\min_{\boldsymbol{\lambda}} L(\boldsymbol{\lambda})$, with dual function $L(\boldsymbol{\lambda}) = \max_{\boldsymbol{x}^d} \left\{ f^d(\boldsymbol{x}^d) + \sum_{k \in F} \boldsymbol{\lambda}_k^d \boldsymbol{x}_{|k}^d \right\} + \max_{\boldsymbol{x}^c} \left\{ f^c(\boldsymbol{x}^c) + \sum_{k \in F} \boldsymbol{\lambda}_k^c \boldsymbol{x}_{|k}^c \right\} + \sum_{k \in F} \max_{\boldsymbol{x}_k^d, \boldsymbol{x}_k^c} \{ f_k^m(\boldsymbol{x}_k^d, \boldsymbol{x}_k^c) - \boldsymbol{\lambda}_k^d \boldsymbol{x}_k^d - \boldsymbol{\lambda}_k^c \boldsymbol{x}_k^c \}$. Here, the master problem coordinates the $2 + |F|$ maximization subproblems, where $\boldsymbol{x}_{|k}^d$and $\boldsymbol{x}_{|k}^c$ respectively denote the subvectors of $\boldsymbol{x}^d$ and $\boldsymbol{x}^c$ containing only the variables of factor $k \in F$, $\boldsymbol{\lambda}_k^d$ and $\boldsymbol{\lambda}_k^c$ are their corresponding Lagrange multipliers. Intuitively, by updating the dual variables $\boldsymbol{\lambda}$, the master problem ensures agreement on the involved variables between the discrete and continuous subproblems and the mixed factors. Importantly, the dual function $L(\boldsymbol{\lambda})$ only involves independent maximization over local assignments of $\boldsymbol{x}^d, \boldsymbol{x}^c$ and $\boldsymbol{x}_k^d, \boldsymbol{x}_k^c$, which are assumed to be tractable. There are two main classes of algorithms used for the maximization, namely subgradient methods and block coordinate descent [SGJ11].

Bibliography

[Abb12] Y. Abbasi-Yadkori. "Online learning for linearly parametrized control problems". PhD thesis. Edmonton, Alberta, 2012.

[AL+17] M. Abeille, A. Lazaric, et al. "Linear Thompson sampling revisited". In: *EJS* (2017).

[Agl+21] V. Aglietti, N. Dhir, J. Gonzalez, and T. Damoulas. "Dynamic Causal Bayesian Optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[Agl+20] V. Aglietti, X. Lu, A. Paleyes, and J. González. "Causal Bayesian Optimization". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[AG13] S. Agrawal and N. Goyal. "Thompson sampling for contextual bandits with linear payoffs". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2013.

[ABK07] C. Antoniou, M. Ben-Akiva, and H. N. Koutsopoulos. "Nonlinear Kalman Filtering Algorithms for On-Line Calibration of Dynamic Traffic Assignment Models". In: *IEEE Transactions on Intelligent Transportation Systems* 8 (4 2007), pp. 661–670.

[Ant+15] C. Antoniou, C. Lima Azevedo, L. Lu, F. Pereira, and M. Ben-Akiva. "W-SPSA in Practice: Approximation of Weight Matrices and Calibration of Traffic Simulation Models". In: *Transportation Research Part C: Emerging Technologies* 7 (June 2015).

[AMN19] J. Arbel, O. Marchal, and H. D. Nguyen. "On strict sub-Gaussianity, optimal proxy variance and symmetry for bounded random variables". In: *arXiv:1901.09188* (2019).

[AFF10] J. Azimi, A. Fern, and X. Z. Fern. "Batch bayesian optimization via simulation matching". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2010.

[BH10] L. Bajer and M. Holeňa. "Surrogate Model for Continuous and Discrete Genetic Optimization Based on RBF Networks". In: *Proceedings of the 11th International Conference on Intelligent Data Engineering and Automated Learning*. 2010.

[Bal+19] M. Balac, H. Becker, F. Ciari, and K. Axhausen. "Modeling competing free-floating carsharing operators - A case study for Zurich, Switzerland". In: *Transportation Research: Part C* 98 (2019), pp. 101–117.

[Bal+18] M. Balac, A. Vetrella, R. Rothfeld, and B. Schmid. "Demand estimation for aerial vehicles in urban settings". In: *IEEE Intelligent Transportation Systems Magazine* (2018).

[BCA15] M. Balac, F. Ciari, and K. W. Axhausen. "Carsharing demand estimation: Zurich, switzerland, area case study". In: *Transportation Research Record* 2563.1 (2015), pp. 10–18.

[BRH19] M. Balac, R. L. Rothfeld, and S. Hörl. "The Prospects of on-demand Urban Air Mobility in Zurich, Switzerland". In: *22nd IEEE Intelligent Transportation Systems Conference*. IEEE. Oct. 2019, pp. 906–913.

[BBK07] R. Balakrishna, M. Ben-Akiva, and H. Koutsopoulos. "Offline calibration of dynamic traffic assignment: simultaneous demand-and-supply estimation". In: *Transportation Research Record* 2003 (2007), pp. 50–58.

[Bal+20] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. "BoTorch: a framework for efficient Monte-Carlo Bayesian optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[BP18] R. Baptista and M. Poloczek. "Bayesian Optimization of Combinatorial Structures". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2018.

[Bay+20] P. Bayle, A. Bayle, L. Janson, and L. Mackey. "Cross-validation Confidence Intervals for Test Error". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[Ben00] Y. Bengio. "Gradient-based optimization of hyperparameters". In: *Neural computation* 12.8 (2000), pp. 1889–1900.

[Ben18] E. Benhamou. "A few properties of sample variance". In: *arXiv:1809.03774* (2018).

[Ber+11a] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. "Algorithms for hyperparameter optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2011.

[BB12] J. Bergstra and Y. Bengio. "Random Search for HyperParameter Optimization". In: *Journal of Machine Learning Research (JMLR)* (2012).

[Ber+11b] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. "Algorithms for hyper-parameter optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2011.

[BKS16] F. Berkenkamp, A. Krause, and A. P. Schoellig. "Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics". In: *arXiv:1602.04450* (2016).

[Bin+19] M. Binois, J. Huang, R. B. Gramacy, and M. Ludkovski. "Replication or Exploration: Sequential Design for Stochastic Simulation Experiments". In: *Technometrics*. 2019.

[BGL18] M. Binois, R. B. Gramacy, and M. Ludkovski. "Practical Heteroscedastic Gaussian Process Modeling for Large Simulation Experiments". In: *Journal of Computational and Graphical Statistics* (2018).

[Bis06] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[BK21] I. Bogunovic and A. Krause. "Misspecified Gaussian Process Bandit Optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[BKS20] I. Bogunovic, A. Krause, and J. Scarlett. "Corruption-tolerant Gaussian Process bandit optimization". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[Bog+18] I. Bogunovic, J. Scarlett, S. Jegelka, and V. Cevher. "Adversarially Robust Optimization with Gaussian Processes". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[Bog+16] I. Bogunovic, J. Scarlett, A. Krause, and V. Cevher. "Truncated variance reduction: A unified approach to Bayesian optimization and level-set estimation". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[BH02] E. Boros and P. L. Hammer. "Pseudo-boolean optimization". In: *Discrete applied mathematics* 123.1-3 (2002), pp. 155–225.

[Bor+21] V. Borovitskiy, I. Azangulov, A. Terenin, P. Mostowsky, M. Deisenroth, and N. Durrande. "Matérn Gaussian Processes on Graphs". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2021.

[Bor+22] V. Borovitskiy, M. R. Karimi, V. R. Somnath, and A. Krause. *Isotropic Gaussian Processes on Finite Spaces of Graphs*. 2022.

[BCD10] E. Brochu, V. M. Cora, and N. De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599* (2010).

[BGS16] Y. Burda, R. Grosse, and R. Salakhutdinov. "Importance weighted autoencoders". In: *International Conference on Learning Representations (ICLR)*. 2016.

[Cak+20] S. Cakmak, R. Astudillo, P. I. Frazier, and E. Zhou. "Bayesian Optimization of Risk Measures". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[Cal+16] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth. "Bayesian optimization for learning gaits under uncertainty". In: *Annals of Mathematics and Artificial Intelligence* (2016).

[CG16] T. Chen and C. Guestrin. "XGBoost: A scalable tree boosting system". In: *KDD*. 2016.

[Che+18] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas. "Bayesian Optimization in AlphaGo". In: *arXiv:1812.06855* (2018).

[Chi+11] Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks. "Dynamic traffic assignment: A primer". In: *Transportation Research Circular* E-C153 (2011).

[CG17] S. R. Chowdhury and A. Gopalan. "On kernelized multi-armed bandits". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2017.

[Cho+22] S. R. Chowdhury, P. Saux, O.-A. Maillard, and A. Gopalan. *Bregman Deviations of Generic Exponential Families*. 2022. URL: https://arxiv.org/abs/2201.07306.

[CT06] T. M. Cover and J. A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. July 2006.

[Cow+21] A. I. Cowen-Rivers, W. Lyu, R. Tutunov, Z. Wang, A. Grosnit, R. R. Griffiths, H. Jianye, J. Wang, and H. B. Ammar. "An Empirical Study of Assumptions in Bayesian Optimisation". In: *arXiv:2012.03826* (2021).

[Dai+18] H. Dai, Y. Tian, B. Dai, S. Skiena, and L. Song. "Syntax-directed variational autoencoder for structured data". In: *arXiv preprint arXiv:1802.08786* (2018).

[Dau+22] S. Daulton, S. Cakmak, M. Balandat, M. A. Osborne, E. Zhou, and E. Bakshy. "Robust Multi-Objective Bayesian Optimization Under Input Noise". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2022.

[DH19] E. Daxberger and J. M. Hernández-Lobato. "Bayesian variational autoencoders for unsupervised out-of-distribution detection". In: *arXiv preprint arXiv:1912.05651* (2019).

[Dax+20] E. Daxberger, A. Makarova, M. Turchetta, and A. Krause. "Mixed-Variable Bayesian Optimization". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. July 2020.

[DFO20] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. *Mathematics for Machine Learning*. 2020.

[Dep19] S. Depeweg. "Modeling Epistemic and Aleatoric Uncertainty with Bayesian Neural Networks and Latent Variables". PhD thesis. 2019.

[Dep+18] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2018.

[DKB14] T. Desautels, A. Krause, and J. W. Burdick. "Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization". In: *Journal of Machine Learning Research (JMLR)* 15 (2014), pp. 4053–4103.

[Dha+20] A. Dhall, A. Makarova, O. Ganea, D. Pavllo, M. Greeff, and A. Krause. "Hierarchical Image Classification using Entailment Cone Embeddings". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Differential Geometry*. 2020.

[Dju14] T. Djukic. "Dynamic OD demand estimation and prediction for dynamic traffic management". PhD thesis. Delft University of Technology, 2014.

[DY20] X. Dong and Y. Yang. "NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search". In: *International Conference on Learning Representations (ICLR)*. 2020.

[DMP18] A. Durand, O.-A. Maillard, and J. Pineau. "Streaming kernel regression with provably adaptive mean, variance, and regularization". In: *Journal of Machine Learning Research (JMLR)* (2018).

[Egg+15] K. Eggensperger, F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Efficient Benchmarking of Hyperparameter Optimizers via Surrogates." In: *AAAI*. 2015.

[Eri+19] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek. "Scalable Global Optimization via Local Bayesian optimization". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[FKH18] S. Falkner, A. Klein, and F. Hutter. "BOHB: Robust and efficient hyperparameter optimization at scale". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2018.

[Fed17] Federal Statistical Office, Federal Office for Spatial Development. “Verkehrsverhalten der Bevölkerung. Ergebnisse des Mikrozensus Mobilität und Verkehr 2015”. In: (2017).

[Fei+20] Y. Fei, Z. Yang, Y. Chen, Z. Wang, and Q. Xie. “Risk-sensitive reinforcement learning: Near-optimal risk-sample tradeoff in regret”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[Fis30] R. A. Fisher. “The genetical theory of natural selection: a complete variorum edition”. In: *Oxford University Press* (1930).

[Flö17] G. Flötteröd. “A search acceleration method for optimization problems with transport simulation constraints”. In: *Transportation Research Part B: Methodological* 98 (2017), pp. 239–260.

[GG16] Y. Gal and Z. Ghahramani. “A Theoretically Grounded Application of Dropout in Recurrent Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[GST13] N. Galichet, M. Sebag, and O. Teytaud. “Exploration vs Exploitation vs Safety: Risk-Aware Multi-Armed Bandits”. In: *Proceedings of the 5th Asian Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR, 2013.

[Gar+14] J. R. Gardner, M. J. Kusner, Z. Xu, K. Q. Weinberger, and J. P. Cunningham. “Bayesian Optimization with Inequality Constraints”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2014.

[GH18] E. C. Garrido-Merchán and D. Hernández-Lobato. “Dealing with Categorical and Integer-valued Variables in Bayesian Optimization with Gaussian Processes”. In: *CoRR* (2018).

[Gei75] S. Geisser. “The predictive sample reuse method with applications”. In: *Journal of The American Statistical Association* (1975).

[Góm+18] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. “Automatic chemical design using a data-driven continuous representation of molecules”. In: *ACS central science* 4.2 (2018), pp. 268–276.

[Gon16] J. González. *GPyOpt: A Bayesian Optimization framework in Python*. 2016.

[Gon+16] J. González, Z. Dai, P. Hennig, and N. Lawrence. “Batch Bayesian Optimization via Local Penalization”. In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2016, pp. 648–657.

[Gon+15] J. González, J. Longworth, D. C. James, and N. D. Lawrence. “Bayesian Optimization for Synthetic Gene Design”. In: *arXiv:1505.01627* (2015).

[GPy12] GPy. *GPy: A Gaussian process framework in python*. `http://github.com/SheffieldML/GPy`. since 2012.

[GH20] R.-R. Griffiths and J. M. Hernández-Lobato. “Constrained Bayesian optimization for automatic chemical design using variational autoencoders”. In: *Chemical Science* (2020).

[Ha+19] H. Ha, S. Rana, S. Gupta, T. Nguyen, H. Tran-The, and S. Venkatesh. "Bayesian optimization with Unknown Search Space". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019, pp. 11795–11804.

[Han06] N. Hansen. "The CMA Evolution Strategy: A Comparing Review". In: *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*. Springer Berlin Heidelberg, 2006, pp. 75–102.

[Has17] T. J. Hastie. "Generalized additive models". In: *Statistical models in S*. Routledge, 2017.

[HKY17] E. Hazan, A. Klivans, and Y. Yuan. "Hyperparameter Optimization: A Spectral Approach". In: 2017.

[HS12] P. Hennig and C. Schuler. "Entropy Search for Information-Efficient Global Optimization". In: *Journal of Machine Learning Research (JMLR)* (2012).

[HHD13] D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont. "Generalized spike-and-slab priors for Bayesian group feature selection using expectation propagation". In: *Journal of Machine Learning Research* (2013).

[HHG14] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. "Predictive Entropy Search for Efficient Global Optimization of Black-box Functions". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014, pp. 918–926.

[Her+15] J. M. Hernández-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani. "Predictive entropy search for bayesian optimization with unknown constraints". In: *JMLR* (2015).

[HHS15] J. M. Hernández-Lobato, D. Hernández-Lobato, and A. Suárez. "Expectation propagation in linear regression models with spike-and-slab priors". In: *Machine Learning* 99.3 (2015), pp. 437–487.

[Hoa+18] T. N. Hoang, Q. M. Hoang, R. Ouyang, and K. H. Low. "Decentralized high-dimensional bayesian optimization with factor graphs". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[HBA19a] S. Hörl, F. Becker, and K. Axhausen. "Dynamische Nachfrageabschätzung für ein AMoD-System in Zürich". In: *Strassenverkehrstechnik* 2019.4 (2019), pp. 277–281.

[HBA19b] S. Hörl, M. Balac, and K. W. Axhausen. "Dynamic demand estimation for an AMoD system in Paris". In: *IEEE Intelligent Vehicles Symposium 2019*. Paris, 2019, pp. 260–266.

[HBA19c] S. Hörl, M. Balac, and K. W. Axhausen. "Pairing discrete mode choice models and agent-based transport simulation with MATSim". In: *98th Annual Meeting of the Transportation Research Board (TRB)*. 2019.

[Hör+19] S. Hörl, F. Becker, T. Dubernet, and K. W. Axhausen. *Induced demand by autonomous vehicles: An assessment*. Tech. rep. Zurich, 2019.

[HNA16] A. Horni, K. Nagel, and K. W. Axhausen, eds. *The Multi-Agent Transport Simulation MATSim*. London: Ubiquity, 2016.

[HW21] E. Hüllermeier and W. Waegeman. "Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods". In: *Machine Learning* (2021).

[Hut+21] M. J. Hutchinson, A. Terenin, V. Borovitskiy, S. Takao, Y. W. Teh, and M. P. Deisenroth. "Vector-valued Gaussian Processes on Riemannian Manifolds via Gauge Independent Projected Kernels". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[HHL10] F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Automated configuration of mixed integer programming solvers". In: *CPAIOR*. 2010.

[HHL11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. "Sequential model-based optimization for general algorithm configuration". In: *LION*. 2011.

[IBM09] IBM. "User's Manual for CPLEX". In: *International Business Machines Corporation* 46.53 (2009), p. 157.

[18] "Implementation of free-floating and station-based carsharing in an agent-based travel demand model". In: *Travel Behaviour and Society* (2018).

[IIT21] S. Iwazaki, Y. Inatsu, and I. Takeuchi. "Mean-Variance Analysis in Bayesian Optimization under Uncertainty". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2021.

[Jai+22] M. Jain, S. Lahlou, H. Nekoei, V. I. Butoi, P. Bertin, J. Rector-Brooks, M. Korablyov, and Y. Bengio. *DEUP: Direct Epistemic Uncertainty Prediction*. 2022.

[Jen+17] R. Jenatton, C. Archambeau, J. González, and M. Seeger. "Bayesian Optimization with Tree-structured Dependencies". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2017, pp. 1655–1664.

[JBJ18] W. Jin, R. Barzilay, and T. Jaakkola. "Junction tree variational autoencoder for molecular graph generation". In: *arXiv preprint arXiv:1802.04364* (2018).

[Kan+16] K. Kandasamy, G. Dasarathy, J. B. Oliva, J. G. Schneider, and B. Póczos. "Gaussian Process Bandit Optimisation with Multi-fidelity Evaluations". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 992–1000.

[Kan+17] K. Kandasamy, G. Dasarathy, J. G. Schneider, and B. Póczos. "Multi-fidelity Bayesian Optimisation with Continuous Approximations". In: *Proceedings of International Conference on Machine Learning (ICML)*. JMLR.org, 2017, pp. 1799–1808.

[KK21] P. Kassraie and A. Krause. *Neural Contextual Bandits without Regret*. 2021.

[Kim+19] J. Kim, M. McCourt, T. You, S. Kim, and S. Choi. "Bayesian optimization over sets". In: *arXiv preprint arXiv:1905.09780* (2019).

[KW14] D. P. Kingma and M. Welling. "Auto-encoding variational bayes". In: *International Conference on Learning Representations (ICLR)*. 2014.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *Science* 220.4598 (1983), pp. 671–680.

[Kir+20] J. Kirschner, I. Bogunovic, S. Jegelka, and A. Krause. "Distributionally Robust Bayesian Optimization". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[KK18] J. Kirschner and A. Krause. "Information Directed Sampling and Bandits with Heteroscedastic Noise". In: *Conference On Learning Theory (COLT)*. 2018.

[Kir+19] J. Kirschner, M. Mutny, N. Hiller, R. Ischebeck, and A. Krause. "Adaptive and Safe Bayesian Optimization in High Dimensions via One-Dimensional Subspaces". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2019.

[Kle+17] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter. "Learning Curve Prediction with Bayesian Neural Networks". In: *International Conference on Learning Representations (ICLR)*. 2017.

[Kle+16] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. "Fast bayesian optimization of machine learning hyperparameters on large datasets". In: *arXiv preprint arXiv:1605.07079* (2016).

[KH19] A. Klein and F. Hutter. "Tabular Benchmarks for Joint Architecture and Hyperparameter Optimization". In: *arXiv:1905.04970* (2019).

[Kle+20] A. Klein, L. C. Tiao, T. Lienart, C. Archambeau, and M. Seeger. "Model-based asynchronous hyperparameter and neural architecture search". In: *arXiv preprint arXiv:2003.10865* (2020).

[Koe+23] C. Koenig, M. Ozols, A. Makarova, E. C. Balta, A. Krause, and A. Rupenyan. "Safe Risk-averse Bayesian Optimization for Controller Tuning". In: *arXiv preprint arXiv:2306.13479* (2023).

[KLT03] T. G. Kolda, R. M. Lewis, and V. Torczon. "Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods". In: *SIAM Review* 45 (2003), pp. 385–482.

[KFB09] D. Koller, N. Friedman, and F. Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[KPT11] N. Komodakis, N. Paragios, and G. Tziritas. "MRF energy minimization and beyond via dual decomposition". In: *IEEE transactions on pattern analysis and machine intelligence* (2011).

[Kor+19] K. Korovina, S. Xu, K. Kandasamy, W. Neiswanger, B. Poczos, J. Schneider, and E. P. Xing. "ChemBO: Bayesian Optimization of Small Organic Molecules with Synthesizable Recommendations". In: *arXiv:1908.01425* (2019).

[KSG08] A. Krause, A. Singh, and C. Guestrin. "Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies". In: *JMLR* (2008).

[KFE18] V. Kuleshov, N. Fenner, and S. Ermon. "Accurate Uncertainties for Deep Learning Using Calibrated Regression". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2018, pp. 2796–2804.

[Kus64] H. J. Kushner. "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise". In: *Journal of Basic Engineering* 86.1 (1964), pp. 97–106.

[KPH17] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. "Grammar variational autoencoder". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2017, pp. 1945–1954.

[LR85a] T. Lai and H. Robbins. "Asymptotically Efficient Adaptive Allocation Rules". In: (1985).

[LR85b] T. Lai and H. Robbins. “Asymptotically efficient adaptive allocation rules”. In: *Advances in Applied Mathematics* (1985).

[LPB17] B. Lakshminarayanan, A. Pritzel, and C. Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[LS20] T. Lattimore and C. Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[LB21] Y.-A. Le Borgne and G. Bontempi. *Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Université Libre de Bruxelles, 2021.

[Li+20] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, J. Ben-Tzur, M. Hardt, B. Recht, and A. Talwalkar. “A system for massively parallel hyperparameter tuning”. In: *Proceedings of Machine Learning and Systems* 2 (2020), pp. 230–246.

[Li+17] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. “Hyperband: A novel bandit-based approach to hyperparameter optimization”. In: *Journal of Machine Learning Research (JMLR)* (2017).

[Li+18] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. “Hyperband: A novel bandit-based approach to hyperparameter optimization”. In: *JMLR* (2018).

[Liz08] D. J. Lizotte. “Practical Bayesian Optimization”. AAINR46365. PhD thesis. Edmonton, Alta., Canada: University of Alberta, 2008.

[Lor+16] R. Lorenz, R. P. Monti, I. R. Violante, A. A. Faisal, C. Anagnostopoulos, R. Leech, and G. Montana. “Stopping criteria for boosting automatic experimental design using real-time fMRI with Bayesian optimization”. In: *arXiv:1511.07827* (2016).

[Lu+15] L. Lu, Y. Xu, C. Antoniou, and M. Ben-Akiva. “An enhanced SPSA algorithm for the calibration of Dynamic Traffic Assignment models”. In: *Transportation Research Part C: Emerging Technologies* 51 (2015), pp. 149–166.

[Lu+18] X. Lu, J. Gonzalez, Z. Dai, and N. Lawrence. “Structured Variationally Auto-encoded Optimization”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2018, pp. 3273–3281.

[Luo+18] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu. “Neural architecture optimization”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 7827–7838.

[MH19] O. Mahmood and J. M. Hernández-Lobato. “A COLD Approach to Generating Optimal Samples”. In: *arXiv preprint arXiv:1905.09885* (2019).

[Mak+21a] A. Makarova, H. Shen, V. Perrone, A. Klein, J. B. Faddoul, A. Krause, M. Seeger, and C. Archambeau. “Overfitting in Bayesian Optimization: an empirical study and early-stopping solution”. In: *ICLR Workshop on Neural Architecture Search*. 2021.

[Mak+21b] A. Makarova, I. Usmanova, I. Bogunovic, and A. Krause. “Risk-averse Heteroscedastic Bayesian optimization”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[MT11] S. Mannor and J. Tsitsiklis. “Mean-Variance Optimization in Markov Decision Processes”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2011.

[Mar+17] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe. "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[Mar+16] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. "Automatic LQR tuning based on Gaussian process global optimization". In: *International Conference on Robotics and Automation (ICRA)*. 2016.

[MOR17] M. McLeod, M. A. Osborne, and S. J. Roberts. "Practical bayesian optimization for variable cost objectives". In: *arXiv preprint arXiv:1703.04335* (2017).

[MRO18] M. McLeod, S. Roberts, and M. A. Osborne. "Optimization, fast and slow: optimally switching between local and Bayesian optimization". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2018.

[Min01] T. P. Minka. "Expectation propagation for approximate Bayesian inference". In: *Uncertainty in Artificial Intelligence (UAI)*. 2001, pp. 362–369.

[Moč75] J. Močkus. "On Bayesian methods for seeking the extremum". In: *Optimization Techniques*. 1975.

[MA12] T. Moldovan and P. Abbeel. "Risk Aversion in Markov Decision Processes via Near Optimal Chernoff Bounds". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[Mos+20] H. Moss, D. Leslie, D. Beck, J. González, and P. Rayson. "BOSS: Bayesian optimization over string spaces". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[MK18] M. Mutný and A. Krause. "Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[NB03] C. Nadeau and Y. Bengio. "Inference for the generalization error". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2003.

[Nea96] R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996.

[NFP11] D. M. Negoescu, P. I. Frazier, and W. B. Powell. "The knowledge-gradient algorithm for sequencing experiments in drug discovery". In: *INFORMS* (2011).

[Ngu+16] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks". In: *Advances in neural information processing systems*. 2016, pp. 3387–3395.

[Ngu+21a] Q. P. Nguyen, Z. Dai, B. K. H. Low, and P. Jaillet. "Optimizing Conditional Value-At-Risk of Black-Box Functions". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan. 2021.

[Ngu+21b] Q. P. Nguyen, Z. Dai, B. K. H. Low, and P. Jaillet. "Value-at-Risk Optimization with Gaussian Processes". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2021.

[Ngu+20a] T. Nguyen, S. Gupta, H. Ha, S. Rana, and S. Venkatesh. "Distributionally Robust Bayesian Quadrature Optimization". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[Ngu+20b] T. T. Nguyen, S. Gupta, H. Ha, S. Rana, and S. Venkatesh. *Distributionally Robust Bayesian Quadrature Optimization*. 2020.

[Ngu+17] V. Nguyen, S. Gupta, S. Rana, C. Li, and S. Venkatesh. "Regret for expected improvement over the best-observed value and stopping condition". In: *Asian Conference on Machine Learning*. 2017.

[Nic12] H. Nickisch. "glm-ie: generalised linear models inference & estimation toolbox". In: *Journal of Machine Learning Research* (2012).

[OGW21] C. Oh, E. Gavves, and M. Welling. "Mixed variable Bayesian optimization with frequency modulated kernels". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*. Vol. 161. 2021.

[Oh+19] C. Oh, J. M. Tomczak, E. Gavves, and M. Welling. "Combinatorial Bayesian Optimization using Graph Representations". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[Opt14] G. Optimization. "Gurobi optimizer reference manual". In: *http://www.gurobi.com* (2014).

[Oso19] C. Osorio. "High-dimensional offline origin-destination (OD) demand calibration for stochastic traffic simulators of large-scale road networks". In: *Transportation Research Part B: Methodological* 124 (2019), pp. 18–43.

[OB13] C. Osorio and M. Bierlaire. "A simulation-based optimization framework for urban transportation problems". In: *Operations Research* 61.6 (2013), pp. 1333–1345.

[Per+17] V. Perrone, R. Jenatton, M. Seeger, and C. Archambeau. "Multiple Adaptive Bayesian Linear Regression for Scalable Bayesian Optimization with Warm Start". In: *arXiv preprint arXiv:1712.02902* (2017).

[Pic+13] V. Picheny, D. Ginsbourger, Y. Richet, and G. Caplin. "Quantile-based optimization of noisy computer experiments with tunable precision". In: *Technometrics* 55.1 (2013), pp. 2–13.

[RR08] A. Rahimi and B. Recht. "Random features for large-scale kernel machines". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2008.

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[RG19a] S. Ray Chowdhury and A. Gopalan. "Bayesian Optimization under Heavy-tailed Payoffs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[RG19b] S. Ray Chowdhury and A. Gopalan. "Bayesian Optimization under Heavy-tailed Payoffs". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[Rol+18] P. Rolland, J. Scarlett, I. Bogunovic, and V. Cevher. "High-Dimensional Bayesian Optimization via Additive Models with Overlapping Groups". In: *AISTATS*. 2018.

[Ru+19] B. Ru, A. S. Alvi, V. Nguyen, M. A. Osborne, and S. J. Roberts. "Bayesian optimisation over multiple continuous and categorical inputs". In: *arXiv preprint arXiv:1906.08878* (2019).

[RC12] A. M. Rush and M. Collins. "A tutorial on dual decomposition and Lagrangian relaxation for inference in natural language processing". In: *Journal of Artificial Intelligence Research* 45 (2012), pp. 305–362.

[SKW15] T. Salimans, D. P. Kingma, and M. Welling. "Markov chain Monte Carlo and variational inference: Bridging the gap". In: *Proceedings of International Conference on Machine Learning (ICML)*. 2015.

[SLM12] A. Sani, A. Lazaric, and R. Munos. "Risk-Aversion in Multi-armed Bandits". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[SBF18] W. Scherr, P. Bützberger, and N. Frischknecht. "Micro Meets Macro: A Transport Model Architecture Aiming at Forecasting a Passenger Railway's Future". In: *18th Swiss Transport Research Conference (STRC)*. 2018.

[SHS01] B. Schölkopf, R. Herbrich, and A. J. Smola. "A Generalized Representer Theorem". In: *In Proceedings of the Annual Conference on Computational Learning Theory*. 2001.

[See08] M. W. Seeger. "Bayesian inference and optimal design for the sparse linear model". In: *Journal of Machine Learning Research* (2008).

[SN08] M. W. Seeger and H. Nickisch. "Compressed sensing and Bayesian experimental design". In: *Proceedings of International Conference on Machine Learning (ICML)*. ACM. 2008.

[SN11] M. W. Seeger and H. Nickisch. "Large scale Bayesian inference and experimental design for sparse linear models". In: *SIAM Journal on Imaging Sciences* 4.1 (2011), pp. 166–199.

[Ses+20] P. G. Sessa, I. Bogunovic, M. Kamgarpour, and A. Krause. "Mixed Strategies for Robust Optimization of Unknown Objectives". In: *Conference on Artificial Intelligence and Statistics (AISTATS)*. 2020.

[Sha+16] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *IEEE* (2016).

[Sli19] A. Slivkins. *Introduction to Multi-Armed Bandits*. 2019.

[SLA12a] J. Snoek, H. Larochelle, and R. P. Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 2951–2959.

[SLA12b] J. Snoek, H. Larochelle, and R. P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2012.

[SGJ11] D. Sontag, A. Globerson, and T. Jaakkola. "Introduction to dual composition for inference". In: *Optimization for Machine Learning*. 2011.

[Spa98a] J. C. Spall. "An overview of the simultaneous perturbation method for efficient optimization". In: *Johns Hopkins APL Technical Digest* 19.4 (1998), pp. 482–492.

[Spa98b] J. C. Spall. “Implementation of the simultaneous perturbation algorithm for stochastic optimization”. In: *IEEE Transactions on Aerospace and Electronic Systems* 34.3 (1998), pp. 817–823.

[Spr+16] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. “Bayesian Optimization with Robust Bayesian Neural Networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[Sri+10] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. “Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2010.

[Sto74] M. Stone. “Cross-validatory choice and assessment of statistical predictions”. In: *Journal of the royal statistical society. Series B (Methodological)* (1974).

[Sui+15] Y. Sui, A. Gotovos, J. Burdick, and A. Krause. “Safe exploration for optimization with Gaussian processes”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2015.

[SMK23] S. Sussex, A. Makarova, and A. Krause. “Model-based Causal Bayesian Optimization”. In: *Proceedings of International Conference on Machine Learning (ICML)* (2023).

[Sus+23] S. Sussex, P. G. Sessa, A. Makarova, and A. Krause. “Adversarial Causal Bayesian Optimization”. In: *arXiv preprint arXiv:2307.16625* (2023).

[SB18] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2018.

[SSA14] K. Swersky, J. Snoek, and R. Adams. “Freeze-Thaw Bayesian Optimization”. In: *ArXiv* abs/1406.3896 (2014).

[Tak+20] S. Takeno, H. Fukuoka, Y. Tsukada, T. Koyama, M. Shiga, I. Takeuchi, and M. Karasuyama. “Multi-fidelity Bayesian Optimization with Max-value Entropy Search and its Parallelization”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2020.

[Tho33] W. R. Thompson. “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”. In: *Biometrika* (1933).

[Tho35] W. R. Thompson. “On the theory of apportionment”. In: *American Journal of Mathematics* (1935).

[Tia+21] L. C. Tiao, A. Klein, M. W. Seeger, E. V. Bonilla, C. Archambeau, and F. Ramos. “BORE: Bayesian Optimization by Density-Ratio Estimation”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2021.

[Tra09] K. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2009.

[TDH20] A. Tripp, E. Daxberger, and J. M. Hernández-Lobato. “Sample-efficient optimization in the latent space of deep generative models via weighted retraining”. In: *arXiv preprint arXiv:2006.09191* (2020).

[Tur+18] M. Turchetta, A. Makarova, S. Beyeler, and A. Krause. “Calibration of agent-based transport simulations with multi-fidelity Bayesian optimization”. In: (2018).

[Tym18] A. Tympakianaki. “Demand estimation and bottleneck management using heterogeneous traffic data”. PhD thesis. KTH Royal Institute of Technology, 2018.

[TKJ15] A. Tympakianaki, H. N. Koutsopoulos, and E. Jenelius. “c-SPSA: Cluster-wise simultaneous perturbation stochastic approximation algorithm and its application to dynamic origin-destination matrix estimation”. In: *Transportation Research Part C: Emerging Technologies* 55 (2015), pp. 231–245.

[Usv+21] M. Usvyatsov, A. Makarova, R. Ballester-Ripoll, M. Rakhuba, A. Krause, and K. Schindler. “Cherry-Picking Gradients: Learning Low-Rank Embeddings of Visual Data via Differentiable Cross-Approximation”. In: *International Conference on Computer Vision* (2021).

[VZ16] S. Vakili and Q. Zhao. “Risk-Averse Multi-Armed Bandit Problems Under Mean-Variance Measure”. In: *IEEE Journal of Selected Topics in Signal Processing* 10 (2016), pp. 1093–1111.

[Van+14] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo. “OpenML: networked science in machine learning”. In: *ACM SIGKDD* 15.2 (2014), pp. 49–60.

[Wah90] G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

[WJ+08] M. J. Wainwright, M. I. Jordan, et al. “Graphical models, exponential families, and variational inference”. In: *Foundations and Trends® in Machine Learning* 1.1–2 (2008), pp. 1–305.

[WJ17] Z. Wang and S. Jegelka. “Max-value Entropy Search for Efficient Bayesian Optimization”. In: *Proceedings of International Conference on Machine Learning (ICML)*. 2017, pp. 3627–3635.

[Was10] L. Wasserman. *All of statistics : a concise course in statistical inference*. Springer, 2010.

[WR20] I. Waudby-Smith and A. Ramdas. “Confidence sequences for sampling without replacement”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[Whi94] D. Whitley. “A genetic algorithm tutorial”. In: *Statistics and Computing* 4.2 (June 1994), pp. 65–85.

[WR06] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006.

[ZKN19] D. Ziemke, I. Kaddoura, and K. Nagel. “The MATSim Open Berlin Scenario: A multimodal agent-based transport simulation scenario based on synthetic demand modeling and open data”. In: *Procedia Computer Science* 151 (2019), pp. 870–877.

Curriculum Vitae

Name Anastasiia Makarova
Date of Birth September 7, 1994

Education

2017 - 2023 ETH Zurich
Zürich, Switzerland
Degree: Doctor of Sciences

2015 - 2017 Moscow Institute of Physics and Technology (MIPT)
Degree: Master of Science (with honors)
Applied Mathematics and Physics

2015 - 2017 Skoltech Institute of Science and Technology (Skoltech)
Degree: Master of Science
Computer Science and Engineering

2011 - 2015 Moscow Institute of Physics and Technology (MIPT)
Degree: Bachelor of Science (with honors)
Applied Mathematics and Physics

Employment

2023 **Research Scientist Intern**
Google DeepMind
Zurich, Switzerland

2020 **Research Scientist Intern**
Amazon Web Services
Berlin, Germany

2016 **Visiting Research Scholar**
Columbia University
New York City, United States

2016 **Research Intern**
Yandex
Moscow, Russia