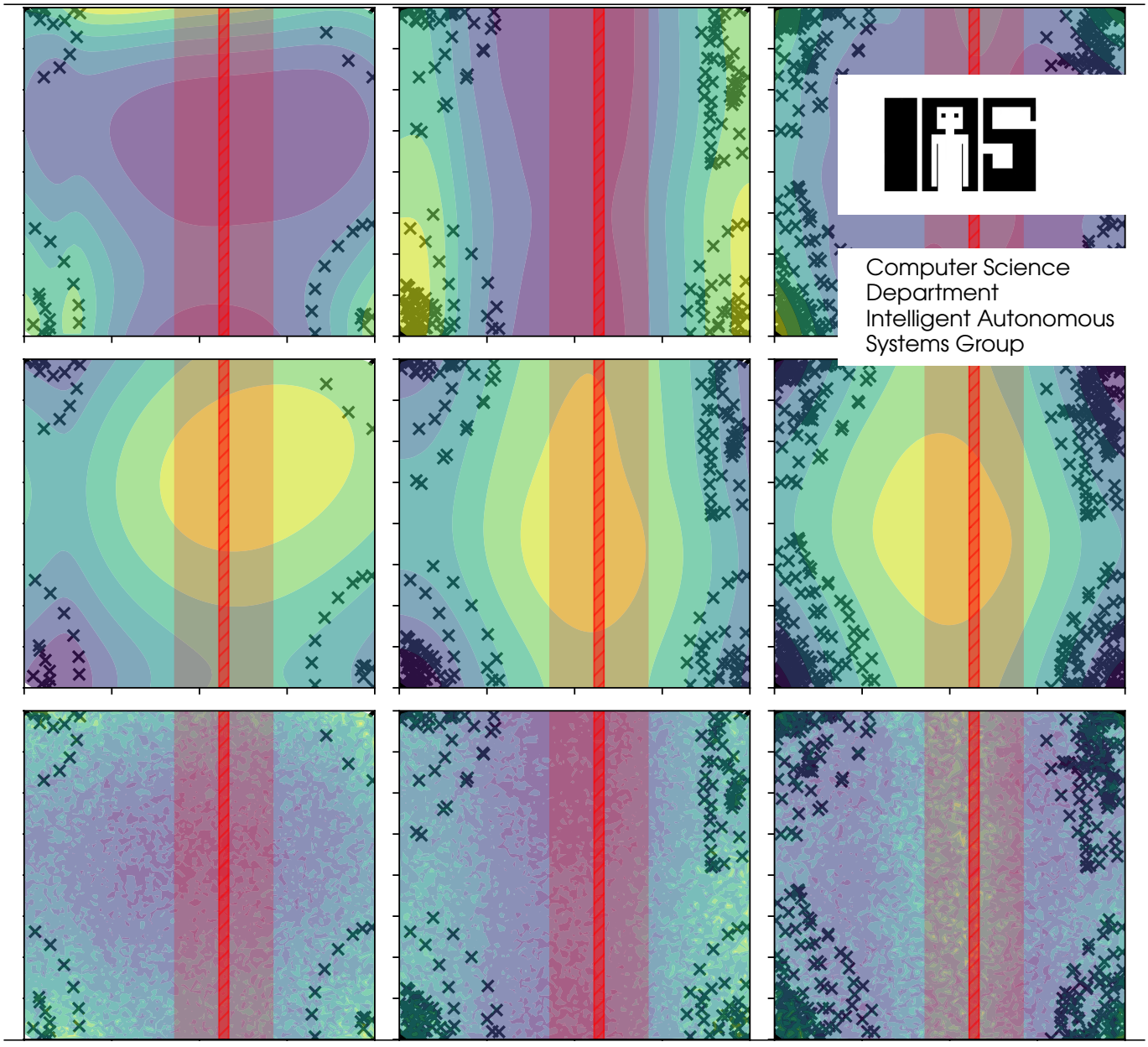
Pushing The Limits of Sample-Efficient Optimisation

Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
Genehmigte Dissertation von Alexander Imani Cowen-Rivers aus Nairobi
Tag der Einreichung: 29th October 2022, Tag der Prüfung: 13th December 2022

1. Gutachten: Prof. Jan Peters
2. Gutachten: Prof. Kristian Kersting
Darmstadt – D 17



Computer Science
Department
Intelligent Autonomous
Systems Group

Pushing The Limits of Sample-Efficient Optimisation

Accepted doctoral thesis by Alexander Imani Cowen-Rivers

1. Review: Prof. Jan Peters
2. Review: Prof. Kristian Kersting

Date of submission: 29th October 2022
Date of thesis defense: 13th December 2022

Darmstadt – D 17

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-241781
URL: http://tuprints.ulb.tu-darmstadt.de/24178

Dieses Dokument wird bereitgestellt von tuprints,
E-Publishing-Service der TU Darmstadt
http://tuprints.ulb.tu-darmstadt.de
tuprints@ulb.tu-darmstadt.de

Darmstadt , Technische Universität Darmstadt
Year thesis published in TUprints 2023
Published under CC BY-SA 4.0 International https://creativecommons.org/licenses/

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 29th October 2022

Alexander Imani Cowen-Rivers

Abstract

Humans excel at confronting problems with little to no prior information about, and with few interactions reasoning over the problems to propose adequate solutions. In other terms, when a human encounters an unknown function, they are able to, with few samples, find a variable such that the evaluation of this variable in the unknown function produces a satisfactory result. However, as the unknown function becomes increasingly non-linear, as well as the design space x becomes increasingly abstract, it becomes harder for a human to utilise prior knowledge around this abstracted black-box function. Bayesian Optimisation, on the other hand, provides a principled approach to reasoning over the, typically expensive to evaluate, unknown function f and exploring regions of uncertainty in an efficient manner. Ubiquitous applications of Bayesian Optimisation range from hyper-parameter tuning, molecule design, sensor placement, antenna design and laser optimisation. Thus improvements in the performance of Bayesian Optimisation can have wide-ranging implications in many practical applications. Bayesian Optimisation also offers the great potential to enable systems to autonomously tune their hyper-parameters, as well as automatically design the machine learning architectures (AutoML).

In this thesis, we want to advance the success of Bayesian optimisation algorithms through revisiting and contributing to the first two stages in detail. In the first stage, the surrogate model is chosen and constructed typically based on certain assumptions of the unknown function, such as whether the function is deterministic or stochastic? and if it's believed to be stochastic is its noise process homoscedastic or heteroscedastic? Do we believe the unknown function to be stationary or non-stationary? To assess the effect of these assumptions, we look at a broad range of applications of tuning machine learning models in typically studied domains. We find that through revisiting these initial assumptions imposed at the start of applying Bayesian Optimisation, we can construct a novel algorithm HEBO that achieves state-of-the-art performance compared to existing methods. HEBO is verified externally also, by the submission of our algorithm into the NeurIPS 2020 Black-box Optimisation challenge, whereby our proposed method achieved 1st place when evaluated on a wide variety of held-out tasks. We then visit the second stage of the optimisation process, and we look at new ways to optimise the acquisition functions by framing them in a mathematically equivalent compositional format, which allows for

the application of a new family of compositional optimisers. We show that on synthetic and real-world experiments, that these compositional methods perform favourably in the majority of applications. Lastly, we attempt to carry through a Bayesian optimisation perspective towards safe sequential decision making and propose new acquisition functions to determine the fitness value for safe reinforcement learning, and evaluate them on a variety of challenging benchmarks such as constrained robotic car, constrained point robot, constrained robotic arm control, constrained pendulum and constrained double pendulum. Whereby all robotic actuators are tasked with reading a goal state whilst avoiding an unsafe region defined within the state space. We find that the incorporation of an acquisition function helps guide exploration and leading to improved sample complexity in acquiring safe policies in training and evaluation.

To summarise, we have developed a new Bayesian optimisation algorithm that was successfully shown to be state-of-the-art internally against prior Black-box optimisation algorithms, as well as externally in the NeurIPS 2020 Black-box optimisation challenge. HEBO was shown to be two orders of magnitude more sample efficient than random search for certain black-box optimisation tasks. We also introduced novel formulations of popular acquisition functions in a mathematically equivalent compositional framework, allowing us to bridge the well-studied field of compositional optimisation together and show the success of doing so across commonly studied synthetic and real-world Bayesian optimisation benchmarks. Finally, we study the important problem of safe sequential decision making and construct novel acquisition functions that allow agents to safely explore their environments.

Acknowledgements

This PhD has been a very exciting and rich journey. There are many intersections encountered during a PhD where a new road can be chosen, and I would not have made the same decisions had it not been for my supervisors Prof Jan Peters and Dr Haitham Bou-Ammar. I am thankful for the knowledge my supervisors have bestowed on me, as well as the commitment to mentoring me. I would like to thank the many collaborators, from whom I have learnt, been inspired and who fundamentally shaped me as a researcher, such as Jun Wang, Aivar Sootla, Rasul Tutunov, Alexandre Maraval, Antoine Grosnit, Asif Khan, Derrick Goh Xin Deik, Wenlong Lu, Daniel Palenicek, Vincent Moens, Ryan Rhys Griffiths and Mohammed Amin Abdullah. I would also not have passed through this PhD so positively without the constant support, advice and love from my family Robert Cowen, Roberta Rivers, Sarah Cowen-Rivers and Daniel Cowen-Rivers, as well as my friends, who all jointly kept me uplifted during difficult times. I would like to further thank Robert Cowen for helping inspire me to go into research from an early age, as well as my Eddie Cowen whom was also an inspiration within our family for scientific research. Lastly, I would also like to further acknowledge my grandparents Robert Rivers, Georgina Booth and Sheila Teff. Without the support of my grandparents allowing me to comfortably study a Bachelors and Masters degree, this PhD would have not been obtainable.

List of Publications Included Thesis

The following three accepted journal articles are included in Chapter 3, 4 and 5 of this thesis respectively.

- Grosnit A, Cowen-Rivers AI, Tutunov R, Griffiths RR, Wang J, Bou-Ammar H. Are we forgetting about compositional optimisers in bayesian optimisation?. Journal Machine Learning Research. 2021 Jan 1;22(1):160-.
- Cowen-Rivers AI, Lyu W, Tutunov R, Wang Z, Grosnit A, Griffiths RR, Maraval AM, Jianye H, Wang J, Peters J, Ammar HB. An empirical study of assumptions in bayesian optimisation. Journal Artificial Intelligence Research. 2022 Jul 1;74(1):1269-1350.
- Cowen-Rivers AI, Palenicek D, Moens V, Abdullah MA, Sootla A, Wang J, Bou-Ammar H. Samba: Safe model-based & active reinforcement learning. Machine Learning. 2022 Jan;111(1):173-203.

Contents

1. Introduction

In Section 1.1 we will first motivate the research problem we dedicate a majority of this thesis to tackling. We will then move on to discuss our approach in Section 1.2, followed by a detailed outline in Section 1.3 and lastly by a summary of the thesis contributions in Section 1.4.

1.1. Motivation For Bayesian Optimisation

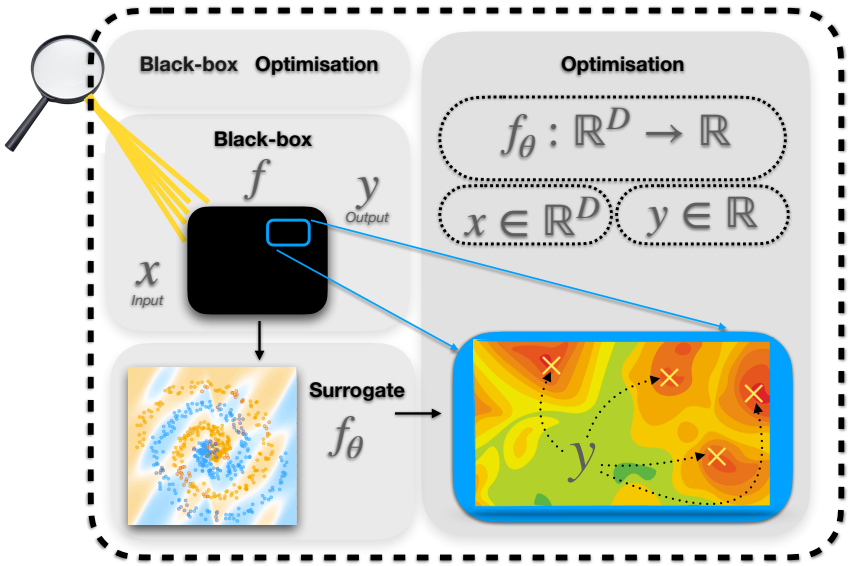


Figure 1.1.: Introduction to Black-box optimisation. Generally, black-box optimisation aims to efficiently traverse an input domain to maximise, or minimise, the value output by evaluating the input in the black-box function.

Black-box optimisation (BBO) has ubiquitous applications, including but not restricted to hyperparameter tuning of deep learning models [66, 121, 249, 72], where the black-box objective is the unknown function mapping between a set of model hyperparameters $\mathbf{x}$ and the validation set performance $f(\mathbf{x})$ measure such as accuracy, negative log-likelihood or mean squared error. Additional applications include automatic chemical design [84, 134, 165, 87], where the black-box objective is the unknown function mapping between a molecule $\mathbf{x}$ and its suitability as a drug candidate $f(\mathbf{x})$. Further examples of black-box optimisation problems appear as subroutines of optimisation algorithms such as immune optimisation [262, 154], ant colony optimisation [260, 224] and genetic algorithms [176], in speech recognition [166] and more broadly across domains spanning architecture [51], chemical engineering [180] and biology [210, 164].

The NeurIPS black-box optimisation challenge is a competition that evaluates black-box optimisation algorithms on real-world score functions. The contest is constructed from automated machine learning (AutoML) tasks [94] and the Bayesmark package, both of which aim at tuning hyperparameters of models to improve validation set performance. Given the importance of black-box optimisation and correctly tuning machine learning algorithms [218], this challenge constitutes a major stepping-stone toward real-world deployment of large-scale models. The challenge also highlights the use of hyperparameter tuning tasks as a proxy for rigorous evaluation tools for black-box optimisation algorithms. Performing learning on real-world datasets with unknown noise (homoscedastic or heteroscedastic) and unknown likelihood (Gaussian or non-Gaussian), with an array of linear and non-linear learning algorithms which can be either differentiable or non-differentiable, yield a non-trivial black-box optimisation test-bed.

In general, hyperparameter tuning can be formulated as a problem of optimising a performance measure (score) for various hyperparameter configurations of ML algorithms. For instance, these parameters can correspond to learning rates, layer depths and widths, or dropout rates during a neural network training step. The 'score' that we would optimise in this instance would be the validation loss. Among various challenges, the main difficulties in optimising hyperparameters are; that performance measures are neither explicitly available in closed form, nor do they necessarily adhere to differentiability assumptions, they can typically be very computationally expensive to evaluate, and there is no precise method to perform mixed variable optimisation. Nonetheless, one can still exploit a low number of model re-training for a better exploration of the hyperparameter space. We can alternatively look at the problem of hyperparameter tuning as an instance of black-box optimisation.

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \tag{1.1}$$

with $\boldsymbol{x}$ denoting optimisation variables that correspond to mixed (continuous and discrete)

hyperparameter settings and the score function $f(\cdot)$ we wish to minimise.

It is this NeurIPS test-bed, which inspire a majority of the experimentation conducted in this thesis to advance the state-of-art in black-box optimisation. We focus our attention on Bayesian optimisation (BO) methods to tackle black-box optimisation problems due to their data efficiency and overall high-quality performance, in contrast to other black-box optimisation techniques which are typically data-hungry and can result in low-quality suggestions. This property is especially useful when evaluation of the objective $f(\boldsymbol{x})$ is costly, which is usually the case for a validation performance of current ML models and especially the case for black-box optimisation tasks such as conducting the wet lab experiments to evaluate new chemical designs [180].

1.2. Bayesian Optimisation Introduction

Bayesian optimisation [140, 163, 117] presents a sample-efficient methodology for black-box optimisation. Bayesian optimisation is of particular interest to the machine learning research community due to its ability for sample-efficient and derivative-free optimisation of objectives. Whilst machine learning models become increasingly complex, as does the difficulty in efficiently identifying hyperparameters. Within the general Bayesian optimisation framework, a crucial performance-determining subroutine is the maximisation of the acquisition function, a task complicated by the fact that acquisition functions tend to be non-convex and thus nontrivial to optimise. The two core components of the BO algorithm are a probabilistic surrogate model and an acquisition function. The probabilistic surrogate model facilitates data efficiency by making use of the full optimisation history to represent the black-box function and additionally leverages uncertainty estimates to guide exploration. Given that the true sequential risk describing the optimality of a sequence of queries is computationally intractable, an acquisition function is a heuristic which acts as a proxy to the true sequential risk. The acquisition function measures the utility of a query point $\mathbf{x}$ by its mean value under the surrogate model (exploitation) as well as its uncertainty under the surrogate model (exploration). At each round of the BO algorithm, the acquisition function is maximised to select the next query point. Gaussian process regression [188] has proven to be a powerful and efficient tool in learning probabilistic surrogate models of unknown functions from even small data - a property of utmost importance in both safe sequential decision making (safe reinforcement learning) and Bayesian optimisation. These probabilistic models place a Gaussian Process prior on f, and infer the true latent relation. They are well studied probabilistic models, with trusted aleatoric and epistemic uncertainty estimates. Clearly, maximising acquisition functions plays a crucial role in Bayesian optimisation as this step constitutes the process

by which the learner yields concrete exploratory actions to improve the guess for the global optimum. The majority of acquisition functions, however, are often intractable, posing formidable challenges during optimisation. In order to tackle these challenges, researchers have proposed a plethora of methods that can generally be categorised into three main groups. Approximation techniques, the first group, replace the quantity of interest with a more readily-computable one e.g. [54] apply expectation propagation [160, 159, 169] as an approximate integration method while [248] apply a mean field approximation to enable a Gumbel sampling approximation to their max-value entropy search acquisition function. As noted in [252], these methods tend to work well in practice but may not converge to the true value of the optimiser. On the other hand, solutions provided in the second group [44] derive near-analytic expressions in the sense that they contain terms such as low-dimensional multivariate normal cumulative density functions that cannot be computed exactly but for which high-quality estimators exist [78, 79]. As noted again by [252], these methods rarely scale to high dimensions. Finally, the third group comprises Monte Carlo (MC) methods [170, 102, 219] which provide unbiased estimators to the acquisition function. MC methods have been successfully used in the context of acquisition function maximisation to the extent that they form the backbone of modern Bayesian optimisation libraries such as BoTorch [17]. Acquisitions can also be used for other applications, such as Active Learning [73]. Herby, the acquisition function is used to help determine which next point to choose in order to reduce entropy of the surrogate model to a given system. Many successful Bayesian Optimisation algorithms work well for Black-box tasks. However, there is no clear consensus about the best surrogate model, acquisition function and method for maximising the acquisition function to select — all important questions which we attempt to answer separately throughout this thesis. Successful Bayesian Optimisation algorithms include TuRBO [65] utilises a Gaussian Process surrogate model and Thompson sampling [253] with a novel trust region method for expanding the search space for the acquisition maximisation, Hyperopt [23] uses a Tree of Parzen Estimator [22] for the surrogate and expected improvement acquisition, BOHB [67] which combines Hyperband with successive halving and Skopt [155] which utilises gradient boosting regression trees as the surrogate model and negative expected improvement for acquisition maximisation. General frameworks exist for creating black-box optimisers for specific domains, such as Emukit [172] which is not a modular library but rather specifies an API to be used with the other components, the high dimensional and asynchronous Bayesian optimiser Dragonfly [122], BoTorch [18] built on-top of GPyTorch [74] a Gaussian process library implemented in PyTorch [174]. The BO library Spearmint [218], one of the oldest open-source BO packages and is no longer maintained. GPyOpt [15] and RoBO [129] were popular frameworks, but similarly are no longer maintained. Cornell-MOE [256] is another popular BO algorithm imple-

mented in C++. In this thesis, we chose to only compare against methods in the most commonly used programming languages for Bayesian Optimisation/ Machine Learning — python. Black-box optimisers that do not execute a Bayesian Optimisation are prevalent in research, such as PySOT [63] which includes warped radial-basis function interpolation for the surrogate model and SOP [137] for next candidate selection, OpenTuner [10] a multi-armed bandit optimisation framework with a sliding window and area under the curve credit assignment and Nevergrad [186] (by default OnePlus One [206, 60, 190]) directly on black-box functions.

1.3. Thesis Outline

In Chapter 3, we aim to understand the best practice for acquisition function maximisation in Bayesian optimisation. We exploit the observation that most common acquisition functions exhibit compositional structure and hence can be equivalently reformulated in a compositional form [245]. Such a reformulation allows a broader class of optimisation techniques to be applied for acquisition function optimisation [240, 80, 247] and in practice can more often enable better numerical performance to be achieved in comparison with standard first and second-order methods. The compositional form is achieved for the expected improvement (EI), simple regret (SR), upper confidence bound (UCB) and probability of improvement (PI) acquisition functions by first exposing the finite-sum form of the re-parameterised acquisition functions derived by [252] and second introducing a deterministic outer function when considering the problem from a matrix-vector perspective. It should be noted that reformulating the acquisition function in a compositional form is distinct from the setting where the black-box function has a compositional form [13]. This compositional form allows us to benefit from the extensive literature of compositional optimisation sanctioning new solvers not attempted before. We highlight the empirical advantages of the compositional approach in 3958 individual experiments comprising both synthetic tasks and tasks from the Bayesmark package. Our results demonstrate that the adoption of compositional optimisers has the potential to yield significant performance improvements in tasks where dimensionality varies between 16 and 120. Given the generality of the acquisition function maximisation subroutine, we posit that the adoption of compositional optimisers has the potential to yield performance improvements in all domains in which Bayesian optimisation is currently being applied. In order to both improve and analyse the optimisation performance on the compositional form of the acquisition function, we introduce several algorithmic adaptations. Firstly, we present (C)L-BFGS; a modification to the L-BFGS algorithm to enable the handling of nested compositional forms. Secondly, we develop AdamOS, a variant of the Adam

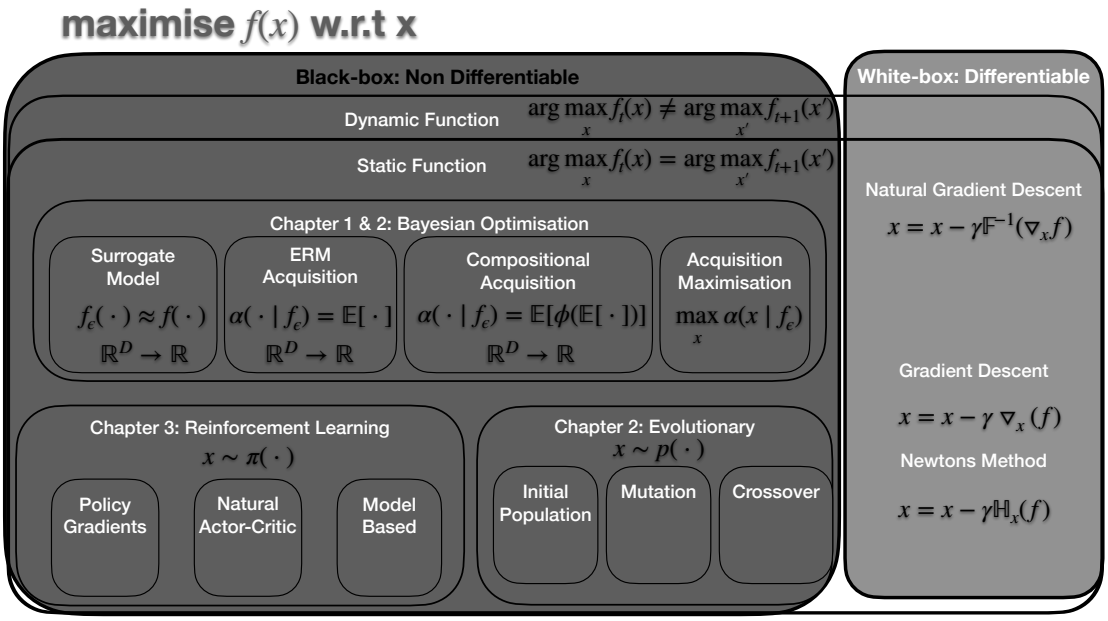


Figure 1.2.: Overview of general approaches function optimisation. Whereby we distinguish white-box/ black-box functions based on whether they are differentiable/ non-differentiable functions. For both differentiable and non-differentiable functions we have both static, e.g. doesn't change over time, to more challenging dynamic functions, which change over time, that we wish to optimise. In terms of methods we can use to optimise, we have an array of tools such as Bayesian Optimisation, Reinforcement Learning (such as Bandit style approaches), Evolutionary methods and an array of first, second order methods when functions are differentiable.

optimiser [126] which borrows the hyperparameter settings of CAdam [240] and facilitates performance comparison between compositional and non-compositional optimisers. Lastly, we formulate a generalised iterative update rule for first-order compositional optimisers and show how the updates of a number of first-order optimisers may be expressed in this manner. Inspired by the increasing desire to conclude what the best approach is for Black-box optimisation. In Chapter 4, we use the test-bed of efficiently tune machine learning hyperparameters to rigorously analyse conventional and non-conventional assumptions inherent to Bayesian optimisation. To that end, we undertake our evaluation in 2140 experiments from 108 real-world problems from the UCI repository [61], which also featured as a testbed in the NeurIPS 2020 black-box optimisation challenge. Across

an extensive set of experiments we conclude that: 1) the majority of hyperparameter tuning tasks exhibit heteroscedasticity and non-stationarity, 2) multi-objective acquisition ensembles with Pareto-front solutions significantly improve queried configurations, and 3) robust acquisition maximisation affords empirical advantages relative to its non-robust counterparts. We hope these findings may serve as guiding principles, both for practitioners and for further research in the field. This work lead to the creation of a new Bayesian Optimisation algorithm that won the NeurIPS 2020 black-box optimisation challenge. In Chapter 5, we apply Gaussian processes for sequential decision-making tasks where algorithms have been shown to solve games [162, 216, 161], and optimal control of simulated and real robots [178, 132, 177, 9, 204, 203], similar to methods such as PILCO [59], but we focus on safety tasks where there exists a dangerous region that an agent may wish to avoid. For example, one could imagine an autonomous vehicle agent as performing sequential decision-making to drive to the end destination, whilst maintaining safety (constraints) by avoiding collisions with other vehicles. In the context of safe Gaussian Process-based sequential decision making, we must mention [181, 182], which still relies on moment matching to approximate gradients through sequences of decisions, while our design relies purely on calculating gradients of immediate (one step) decisions. We propose `SAMBA`, a novel framework for safe sequential decision-making that combines aspects from probabilistic modelling and Bayesian optimisation. Our method builds upon PILCO to enable active exploration using a novel acquisition functions for out-of-sample Gaussian process evaluation. We evaluate our algorithm on a variety of safe dynamical system benchmarks involving both low and high-dimensional state representations. Our results show orders of magnitude reductions in samples and violations compared to state-of-the-art methods. Lastly, we provide intuition as to the effectiveness of the framework by a detailed analysis of our acquisition functions and how they relate to the safety constraints from both toy and practical examples.

1.4. Major Contributions

In summary, in this thesis we re-visit popular design and model assumptions in Bayesian Optimisation to successfully answer important questions such as what are the best choices for the surrogate model?, which is the best acquisition function to use? and what method for maximising the acquisition function should one select? By doing so, we introduced novel formulations of popular acquisition functions in a mathematically equivalent compositional framework, allowing us to bridge the well studied field of compositional optimisation together. We apply a diverse range of compositional optimisers and shown their success across commonly studied Bayesian optimisation benchmarks. In attempting to answer

find the answers to these most general Bayesian Optimisation questions, we developed a new algorithm HEBO that has successfully shown to be state-of-the-art during intrinsic evaluation against prior Bayesian optimisation algorithms, as well in extrinsic evaluation in the NeurIPS black-box optimisation competition. Finally, we study the important problem of safe sequential decision making and construct novel acquisition functions that allow our SAMBA agent to safely explore unknown regions. Our contributions have already led to works such as [236], where it was shown that our algorithm, in 128 steps, achieves a score that would take random search 15,512 steps to achieve. Overall, our algorithm HEBO is two orders of magnitude (121.188 times) more sample efficient than random search. We believe the strong follow up results highlight the significance of the contribution of our work to the machine learning community. Additionally, HEBO was used as the winning submission in the NeurIPS 2021 Machine Learning for Combinatorial Optimisation challenge [77], showing its superiority and dominance for winning two consecutive challenging optimisation challenges.

2. Background

In Section 2.1 we will first introduce the relevant mathematical background needed to understand typical Guassian Process regression (surrogate modelling) and commonly used acquisition functions. Lastly, in Section 2.2 we dive into the latter, and very important component, of acquisition functions and how we use them in a process named acquisition maximisation to generate our next query point.

2.1. Bayesian Optimisation

We consider a sequential decision-making approach to the global optimisation of smooth functions $f : \mathcal{X} \to \mathbb{R}$ over a bounded input domain $\mathcal{X} \subseteq \mathbb{R}^d$. At each decision round, i, we select an input $\mathbf{x}_i \in \mathcal{X}$ and observe the value of the *black-box* function $f(\mathbf{x}_i)$. We allow the returned value to be either deterministic i.e., $y_i = f(\mathbf{x}_i)$ or stochastic with $y_i = f(\mathbf{x}_i) + \epsilon_i$, where ϵ_i denotes a bounded-variance random variable. Our goal is to rapidly approach the maximum $\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ in terms of cumulative regret $R_T = \sum_{t=1}^{T} r_t$ where $r_t = f(\mathbf{x}^\star) - f(\mathbf{x}_t^{(\text{new})})$ is the distance between maximum function value $f(\mathbf{x}^\star)$ and function value at the algorithm's best recommendation at round t denoted as $\mathbf{x}_t^{(\text{new})}$. Since both $f(\cdot)$ and $\mathbf{x}^\star$ are unknown, solvers need to trade off exploitation and exploration during the search process.

To reason about the unknown function, typical Bayesian optimisation algorithms assume smoothness and adopt Bayesian modelling as a principle to carry out inference about the properties of $f(\cdot)$ in light of the observations. Here, one introduces a prior to encode beliefs over the smoothness properties and an observation model to describe collected data, $\mathcal{D}_i = \{\mathbf{x}_l, y_l\}_{l=1}^{n_i}$, up to the i^{th} round with n_i denoting the total acquired data so far. Using these two components in addition to Bayes rule, we can then compute a posterior $p(f(\cdot)|\mathcal{D}_i)$ to encode all knowledge of $f(\cdot)$ allowing us to account for the location of the maximum.

2.1.1. Bayesian Optimisation with Gaussian Processes

A Gaussian process (GP) offers a flexible and sample-efficient procedure for placing priors over unknown functions [188]. These models are fully specified by a mean function $m(\mathbf{x})$ and a covariance function, or kernel, $k(\mathbf{x}, \mathbf{x}')$ that encodes the smoothness assumptions on $f(\cdot)$. Given any finite collection of inputs $\mathbf{x}_{1:n_i}$, the outputs are jointly Gaussian given by

$$f(\mathbf{x}_{1:n_i})|\boldsymbol{\theta} \sim \mathcal{N}\left(m(\mathbf{x}_{1:n_i}), \mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})\right),$$

with mean vector denoted by $[m(\mathbf{x}_{1:n_i})]_k = m(\mathbf{x}_k)$, and covariance matrix $\mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:i}) \in \mathbb{R}^{n_i \times n_i}$ with its $(k,l)^{th}$ entry computed as $[\mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})]_{k,l} = k_{\boldsymbol{\theta}}(\mathbf{x}_k, \mathbf{x}_l)$. Here, parameterised kernel represented as $k_{\boldsymbol{\theta}}(\cdot,\cdot)$ with unknown hyperparameters $\boldsymbol{\theta}$ corresponding to lengthscales or signal amplitudes for example. For ease of presentation following [188], we use a zero-mean prior in our notation here. In terms of the choice of Gaussian process kernel, there are a wide array of options which encode prior modelling assumptions about the latent function. Two of the most commonly-encountered kernels in the Bayesian optimisation literature are the squared exponential (SE) and Matérn$(5/2)$ kernels

$$[\mathbf{K}_{\boldsymbol{\theta}}^{\text{SE}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})]_{k,l} = k_{\boldsymbol{\theta}}^{\text{SE}}(\mathbf{x}_k, \mathbf{x}_l) = \exp\left(-\frac{1}{2}r^2\right),$$
$$[\mathbf{K}_{\boldsymbol{\theta}}^{\text{Matérn}(5/2)}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})]_{k,l} = k_{\boldsymbol{\theta}}^{\text{Matérn}(5/2)}(\mathbf{x}_k, \mathbf{x}_l) = \exp\left(-\sqrt{5}r\right)\left(1 + \sqrt{5}r + \frac{5}{3}r^2\right),$$

with d-dimensional hyperparameters denoted by $r = \sqrt{(\mathbf{x}_k - \mathbf{x}_l)^{\mathsf{T}} \operatorname{diag}(\boldsymbol{\theta}^2)^{-1} (\mathbf{x}_k - \mathbf{x}_l)}$ and $\boldsymbol{\theta} \in \mathbb{R}^d$ with $\boldsymbol{\theta}^2$ executed element-wise. As noted in [188], both kernels are suited for situations where little is known about the latent function in question. The Matérn kernel, however, is arguably suitable for a broader class of real-world Bayesian optimisation problems as it imposes less restrictive smoothness assumptions on $f(\cdot)$ [228]. Following initial experimentation with linear, cosine, squared exponential and various Matérn kernels, we chose the Matérn$(5/2)$ kernel to perform all experiments with.

Given the data $\mathcal{D}_i$, and assuming Gaussian-corrupted observations $y_i = f(\mathbf{x}_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, we can write the joint distribution over the data and an arbitrary evaluation input $\mathbf{x}$ as

$$\begin{bmatrix} \mathbf{y}_{1:n_i} \\ f(\mathbf{x}) \end{bmatrix} \Bigg| \boldsymbol{\theta} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I} & \mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}) \\ \mathbf{k}_{\boldsymbol{\theta}}^{(i),\mathsf{T}}(\mathbf{x}) & k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}) \end{bmatrix}\right),$$

with covariance matrix denoted by $\mathbf{K}_{\boldsymbol{\theta}}^{(i)} = \mathbf{K}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x}_{1:n_i})$ and $\mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}) = \mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_{1:n_i}, \mathbf{x})$. With the above joint distribution derived, we can now easily compute the predictive posterior through marginalisation [188] leading us to $f(\mathbf{x})|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\mu}_i(\mathbf{x}; \boldsymbol{\theta}), \sigma_i(\mathbf{x}; \boldsymbol{\theta})^2\right)$ with

$$\begin{aligned}\mu_i(\mathbf{x}; \boldsymbol{\theta}) &= \mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x})^\mathsf{T}(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{y}_{1:n_i}, \\ \sigma_i(\mathbf{x}; \boldsymbol{\theta})^2 &= k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x})^\mathsf{T}(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}).\end{aligned}$$

Of course, the above can be generalised to the case when a predictive posterior over q arbitrary evaluation points $\mathbf{x}_{1:q}^\star$ needs to be computed as is the case in batched adaptations of Bayesian optimisation. In such a setting $\mathbf{f}(\mathbf{x}_{1:q}^\star)|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{x}_{1:q}^\star; \boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q}^\star; \boldsymbol{\theta}))$ with

$$\begin{aligned}\boldsymbol{\mu}_i(\mathbf{x}_{1:q}^\star; \boldsymbol{\theta}) &= \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:n_i})(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{y}_{1:n_i}, \\ \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q}^\star; \boldsymbol{\theta}) &= \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:q}^\star) - \mathbf{K}_{\boldsymbol{\theta}}^{(i)}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:n_i})(\mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{\boldsymbol{\theta}}^{(i),\mathsf{T}}(\mathbf{x}_{1:q}^\star, \mathbf{x}_{1:n_i}).\end{aligned}$$

The remaining ingredient needed in a GP pipeline is a process to determine the unknown hyperparameters $\boldsymbol{\theta}$ given a set of observation $\mathcal{D}_i$. In standard GPs [188], $\boldsymbol{\theta}$ are fit by minimising the negative log marginal likelihood (NLML) leading us to the following optimisation problem

$$\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2}\det\left(\mathbf{C}_{\boldsymbol{\theta}}^{(i)}\right) + \frac{1}{2}\mathbf{y}_{1:n_i}^\mathsf{T}\mathbf{C}_{\boldsymbol{\theta}}^{(i),-1}\mathbf{y}_{1:n_i} + \frac{n_i}{2}\log 2\pi, \text{ with } \mathbf{C}_{\boldsymbol{\theta}}^{(i)} = \mathbf{K}_{\boldsymbol{\theta}}^{(i)} + \sigma^2\mathbf{I}. \quad (2.1)$$

The objective in Equation (2.1) represents a non-convex optimisation problem making GPs susceptible to local minima. Various off-the-shelf optimisation solvers ranging from first-order [126, 31] to second-order [263, 7] methods have been rigorously studied in the literature. In our experiments, we made use of a set of implementations provided in GPyTorch [75] that relied on a scipy [243] implementation of L-BFGS-B [263] for determining $\boldsymbol{\theta}$. It is also worth noting that gradients of the loss in Equation (2.1) require inverting an $n_i \times n_i$ covariance matrix leading to an order of $\mathcal{O}(n_i^3)$ complexity in each optimisation step. In large data regimes, variational GPs have proved to be a scalable methodology through the usage of $m << n_i$ inducing points [233, 103].

In Bayesian optimisation however, data is typically sparse due to the expense of evaluating even one query of the black-box function, which makes the application of sparse GPs less attractive in these scenarios. While other scalable surrogate models such as Bayesian neural networks (BNNs) and Random Forest have featured in the literature [221, 110], each come with disadvantages. Many BNN-based approaches rely on approximate inference, and hence uncertainty estimates may deteriorate in quality relative to exact GPs while the Random-Forest-based SMAC algorithm is not amenable to gradient-based

optimisation due to a discontinuous response surface [109, 212]. As such, we restrict our focus to exact GPs and direct the reader to external sources for discussion on alternative surrogate models such as sparse GPs [158], BNNs [222, 225, 104], neural processes [125] as well as heteroscedastic GPs [39, 88].

2.1.2. Acquisition Functions

Having introduced a distribution over latent black-box functions and specified mechanisms for updating hyperparameters, we now discuss the process by which novel query points are suggested for collection in order to improve the surrogate model's best guess for the global optimiser $\mathbf{x}^\star$. In Bayesian optimisation, proposing novel query points is performed through maximising an acquisition function $\alpha(\cdot|\mathcal{D}_i)$ that trades off exploration and exploitation by utilising statistics from $p(f(\cdot)|\mathcal{D}_i)$, i.e., $\mathbf{x}_{i+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}|\mathcal{D}_i)$. Acquisition functions can be taxonomised into myopic and non-myopic forms [86]. The former class involves integrals defined in terms of beliefs over unknown outcomes from the black-box function, while the latter class constitutes more complicated nested integrals. In this thesis, we focus on representative examples of standard myopic acquisitions whilst considering entropy search as a widely-used non-myopic acquisition. We detail these acquisitions next.

Expected Improvement: One of the most popular acquisition functions is expected improvement [163, 117], which determines new query points by maximising expected gain relative to the function values observed so far. Formally, denote by $\mathbf{x}_i^+$ an input point in $\mathcal{D}_i$ for which $f(\cdot)$ is maximised, i.e., $\mathbf{x}_i^+ = \arg\max_{\mathbf{x}\in\mathbf{x}_{1:n_i}} f(\mathbf{x})$. Given $\mathbf{x}_i^+$, we define an expected improvement acquisition to compute the expected positive gain in function value compared to the best incumbent point in $\mathcal{D}_i$ as

$$\alpha_{\text{EI}}(\mathbf{x}|\mathcal{D}_i) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max\{(f(\mathbf{x}) - f(\mathbf{x}_i^+)), 0\}\right] = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\text{ReLU}(f(\mathbf{x}) - f(\mathbf{x}_i^+))\right],$$

with ReLU represents a rectified linear unit with $\text{ReLU}(a) = \max\{0, a\}$. The above can be generalised to support a batch form generating query points $\mathbf{x}_{1:q}$ as introduced in [81]. Here, we first compute the multi-dimensional predictive posterior $f(\mathbf{x}_{1:q})|\mathcal{D}_i, \boldsymbol{\theta}$ as described in Section 2.1.1 and then define the maximal gain across all q-batches as

$$\alpha_{\text{q-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\{\text{ReLU}(\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q)\}\right], \tag{2.2}$$

with a q-dimensional vector of ones denoted $\mathbf{1}_q$ and as such, the $\text{ReLU}(\cdot)$ is to be executed element-wise. In words, Equation (2.2) simply computes the expected maximal improvement across all q-dimensional predictions compared to the best incumbent point in $\mathcal{D}_i$.

This form of acquisition is termed joint parallel acquisition function maximisation in [252] (other forms being greedy and incremental) and is chosen for the experiments in this thesis due to its usage in the BoTorch library [17]. In joint parallel acquisition function maximisation, each query point is treated as a dimension of the acquisition surface and the set of batch points is optimised on this surface cf. figure 2 of [252] for an illustration.

Probability of Improvement: Another commonly-used acquisition function in Bayesian optimisation is the probability of improvement criterion which measures the probability of acquiring gains in the function value compared to $f(\mathbf{x}_i^+)$ [140]. Such a probability is measured through an expected Heaviside step function as follows

$$\alpha_{\text{PI}}(\mathbf{x}|\mathcal{D}_i) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\mathbb{1}\{f(\mathbf{x}) - f(\mathbf{x}_i^+)\}\right],$$

with the Heavy side step function $\mathbb{1}\{f(\mathbf{x}) - f(\mathbf{x}_i^+)\} = 1$ if $f(\mathbf{x}) \geq f(\mathbf{x}_i^+)$ and zero otherwise. Analogous to expected improvement, we can extend the acquisition function $\alpha_{\text{PI}}(\mathbf{x}|\mathcal{D}_i)$ to a batch form by generalising the step function to support-vectored random variables in addition to adopting maximal gain across all batches as an improvement metric

$$\alpha_{\text{q-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\left\{\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q\}\right\}\right], \tag{2.3}$$

with a q-dimensional binary vector $\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q\}$ with $[\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\}]_j = 1$ if $[\mathbf{f}(\mathbf{x}_{1:q})]_j \geq [f(\mathbf{x}_i^+)\mathbf{1}_q]_j$ and zero otherwise for all $j \in \{1, \ldots, q\}$.

Simple Regret: In simple regret, new query points are determined by maximising expected outcomes, i.e., $\alpha_{\text{SR}}(\mathbf{x}|\mathcal{D}_i) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}[f(\mathbf{x})]$. This acquisition function can also be generalised to a batch mode by considering the maximal improvement across all q batches leading to

$$\alpha_{\text{q-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\{\mathbf{f}(\mathbf{x}_{1:q})\}\right].$$

Upper Confidence Bound: In this type of acquisition, the learner trades off the mean and variance of the predictive distribution to gather new query points for function evaluation [227]. In the standard form, an upper-confidence bound acquisition can simply be written as: $\alpha_{\text{UCB}}(\mathbf{x}|\mathcal{D}_i) = \mu_i(\mathbf{x};\boldsymbol{\theta}) + \sqrt{\beta}\sigma_i(\mathbf{x};\boldsymbol{\theta})$ with $\beta \in \mathbb{R}$ being a free tuneable hyperparameter. Although widely used, such a form of the upper-confidence bound is not

directly amendable to parallelism. To circumvent this problem, the authors in [252] have shown an equivalent form for the expectation by exploiting reparameterisation leading to

$$\alpha_{\text{UCB}}(\mathbf{x}|\mathcal{D}_i) = \mu_i(\mathbf{x};\boldsymbol{\theta}) + \sqrt{\beta}\sigma_i(\mathbf{x};\boldsymbol{\theta}) = \mathbb{E}_{f(\mathbf{x})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\mu_i(\mathbf{x};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\gamma_i(\mathbf{x};\boldsymbol{\theta})|\right],$$

with variance $\gamma_i(\mathbf{x};\boldsymbol{\theta}) = f(\mathbf{x}) - \mu_i(\mathbf{x};\boldsymbol{\theta})$. Given such a formulation, we can now follow similar reasoning to previous generalisations of acquisition functions and consider a batched version by taking the maximum over all q query points

$$\alpha_{\text{q-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\boldsymbol{\gamma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})|\right\}\right],$$

with variance $\boldsymbol{\gamma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) = \mathbf{f}(\mathbf{x}_{1:q}) - \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$.

Entropy Search: In [102] the authors introduce an information-theoretic approach to select novel query points based on an approximation of the posterior entropy for the global optimiser $\mathbf{x}^\star$. The next point $\mathbf{x}_{i+1}$ is chosen to minimise the posterior entropy $\mathbb{E}_{f(\mathbf{x}|\mathcal{D}_i,),\boldsymbol{\theta}}\left[\mathbb{H}[p(\mathbf{x}^*|\mathcal{D}_i \cup \{\mathbf{x}, f(\mathbf{x})\})]\right]$ and hence minimises the uncertainty over the location of $\mathbf{x}^+$. In [252] a parallel implementation is introduced via a $q-$batch form for the entropy search acquisition function

$$\alpha_{\text{q-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = -\mathbb{E}_{\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i,\boldsymbol{\theta}}\left[\text{H}\left[\mathbb{E}_{f\left(\mathbf{x}^{(\text{g})}_{1:u}\right)|\mathcal{D}_i\cup\{\mathbf{x}_{1:q};\mathbf{f}(\mathbf{x}_{1:q})\},\boldsymbol{\theta}}\left[\mathbb{1}\{\mathbf{f}(\mathbf{x}^{(\text{g})}_{1:u}) - \max_{j\in 1:u} f(\mathbf{x}^{(\text{g})}_j)\mathbf{1}_u\}\right]\right]\right],$$

with a grid of u discrete locations $\mathbf{x}^{(\text{g})}_{1:u}$ sampled from the input domain $\mathcal{X}$ according to a discretisation measure $\mathcal{U}(\cdot|\mathcal{D}_i)$, $\text{H}[\cdot]$ is the Shannon entropy and a u-dimensional binary vector $\mathbb{1}\{\mathbf{f}(\mathbf{x}^{(\text{g})}_{1:u}) - \max_{j\in 1:u} f(\mathbf{x}^{(\text{g})}_j)\mathbf{1}_u\}$ with $[\mathbb{1}\{\mathbf{f}(\mathbf{x}^{(\text{g})}_{1:u}) - \max_{j\in 1:u} f(\mathbf{x}^{(\text{g})}_j)\mathbf{1}_u\}]_\ell = 1$ if $f(\mathbf{x}^{(\text{g})}_\ell) = \max_{j\in 1:u} f(\mathbf{x}^{(\text{g})}_j)$ and zero otherwise for all $\ell \in \{1,\ldots,u\}$.

Following the introduction of GP surrogate models and acquisition functions, we are now ready to present a canonical template for the Bayesian optimisation algorithm. The main steps are summarised in the pseudocode of Algorithm 1.

First, a GP model is fit to the available data (see line 3 of Algorithm 1) enabling the computation of the predictive distribution needed to maximise the acquisition function. Having acquired new query points, the learner then updates the dataset $\mathcal{D}_i$ after which the above process repeats until a total number of iterations N is reached. At the end of the main loop, Algorithm 1 outputs $\mathbf{x}^\star$, the best performing input from all acquired data $\mathcal{D}_N$.

Clearly, maximising acquisition functions plays a crucial role in Bayesian optimisation as this step constitutes the process by which the learner yields concrete exploratory actions

Algorithm 1 General algorithm detailing Batched Bayesian Optimisation with Gaussian Process's. Firstly; we collect (or begin) with an initial dataset of input-output pairs, fit a surrogate model to this dataset (such as a Gaussian Process), maximise an acquisition function in order to get new query points for evaluation in the black-box, evaluate said query points and append the new input-output pairs to our current dataset and repeat. After executing this for N steps, we then return the best-performing query point from our dataset.

Inputs: Total number of outer iterations N, initial randomly-initialised dataset $\mathcal{D}_0 = \{\mathbf{x}_l, y_l \equiv f(\mathbf{x}_l)\}_{l=1}^{n_0}$, batch size q, acquisition function type
for $i = 0 : N - 1$:
Fit the GP model to the current dataset $\mathcal{D}_i$ by $\min_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$ from Equation (2.1)
Find q points by solving $\mathbf{x}_{1:q}^{(\text{new})} = \arg\max_{\mathbf{x}_{1:q}} \alpha_{\text{q-type}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$)
Evaluate new inputs by querying the black-box to acquire $\mathbf{y}_{1:q}^{(\text{new})} = f(\mathbf{x}_{1:q}^{(\text{new})})$
Update the dataset creating $\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{\mathbf{x}_l^{(\text{new})}, y_l^{(\text{new})}\}_{l=1}^{q}$
end for
Output: Return the best-performing query point from the data $\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{D}_N} f(\mathbf{x})$

to improve the guess for the global optimum $\mathbf{x}^\star$. The majority of acquisition functions, however, are often intractable, posing formidable challenges during the optimisation step in Algorithm 1. In order to tackle these challenges, researchers have proposed a plethora of methods that can generally be categorised into three main groups. Approximation techniques, the first group, replace the quantity of interest with a more readily-computable one e.g. [54] apply expectation propagation [160, 159, 169] as an approximate integration method while [248] apply a mean field approximation to enable a Gumbel sampling approximation to their max-value entropy search acquisition function. As noted in [252], these methods tend to work well in practice but may not converge to the true value of the optimiser. On the other hand, solutions provided in the second group [44] derive near-analytic expressions in the sense that they contain terms such as low-dimensional multivariate normal cumulative density functions that cannot be computed exactly but for which high-quality estimators exist [78, 79]. As noted again by [252], these methods rarely scale to high dimensions. Finally, the third group comprises Monte Carlo (MC) methods [170, 102, 219] which provide unbiased estimators to $\alpha(\cdot|\mathcal{D}_i)$. MC methods have been successfully used in the context of acquisition function maximisation to the extent that they form the backbone of modern Bayesian optimisation libraries such as BoTorch [17]. As such, given their prevalence in present-day implementations, we restrict our attention to MC techniques and note three classes of widely-used optimisers. Zeroth-order

procedures [101, 72], such as evolutionary algorithms [194, 28], only use function value information for determining the maximum of the acquisition. First-order methods [126, 31], on the other hand, utilise gradient information during the ascent step, while second-order methods exploit (approximations to) Hessians [38, 263, 34, 239, 238] in their update. During the implementation of first and second-order optimisers, one realises the need for differentiating through an MC estimator with respect to the parameters of the generative distribution $\mathcal{P}(\cdot)$. As described in [252], this can be achieved through reparameterisation in two steps: 1) reparameterising samples from $\mathcal{P}(\cdot)$ as draws from a simpler distribution $\hat{\mathcal{P}}(\cdot)$, and 2) interchanging integration and differentiation by exploiting sample-path derivatives. After reparameterisation, the designer faces two implementation choices which we refer to as ERM-BO and FSM-BO akin to the distinction between empirical risk minimisation [85] and finite sum [200] optimisation forms[1]. In an ERM-BO construction, samples from $\hat{\mathcal{P}}(\cdot)$ are acquired at every iteration of the optimisation algorithm as needed. In contrast, in an FSM-BO setting, all samples from $\hat{\mathcal{P}}(\cdot)$ are obtained upfront and mini-batched during gradient computations. Due to memory consideration, especially in high-dimensional scenarios, the ERM-BO version has been mostly preferred and studied in the literature [131, 17].

In this thesis, however, we are interested in both views and desire to shed light on best practices when optimising acquisition functions. To accomplish such a goal, we carefully probe both settings and realise that an FSM-BO implementation enables a novel connection to a compositional (nested expectation) formulation that sanctions new compositional solvers not previously attempted. Next, we derive such a connection, present memory-efficient optimisation algorithms for FSM-BO, and demonstrate empirical gains in large-scale experiments. For ease of exposition, we summarise the acquisition function derivations of coming work in Figure 2.1 to demonstrate the three steps of reparameterisation, Monte-Carlo estimation for finite-sum forms, and matrix-vector considerations for compositional objectives.

2.2. Acquisition Function Maximisation

The first step in investigating different implementations of BO is to derive relevant reparameterised forms of the acquisition functions in Section 2.1.2. When reparameterising one reinterprets samples $\mathbf{y}_k \sim \mathcal{P}(\mathbf{y}; \boldsymbol{\theta})$ as a deterministic map $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\cdot)$ of a simpler random variable $\mathbf{z}_k \sim \hat{\mathcal{P}}(\mathbf{z})$, that is $\mathbf{y} = \lambda_{\boldsymbol{\theta}}(\mathbf{z})$. Under these conditions, the expectation of some loss $\mathcal{L}(\cdot)$ under $\mathbf{y}$ can be rewritten in terms of $\hat{\mathcal{P}}(\mathbf{z})$ as $\mathbb{E}_{\mathbf{y}\sim\mathcal{P}(y;\boldsymbol{\theta})}[\mathcal{L}(\mathbf{y})] = \mathbb{E}_{\mathbf{z}\sim\hat{\mathcal{P}}(\mathbf{z})}[\mathcal{L}(\lambda_{\boldsymbol{\theta}}(\mathbf{z}))]$ al-

[1] Of course, an empirical risk and a finite sum formulation become equivalent as samples grow large. In reality, infinite samples cannot possibly be acquired hence our two-class categorisation.

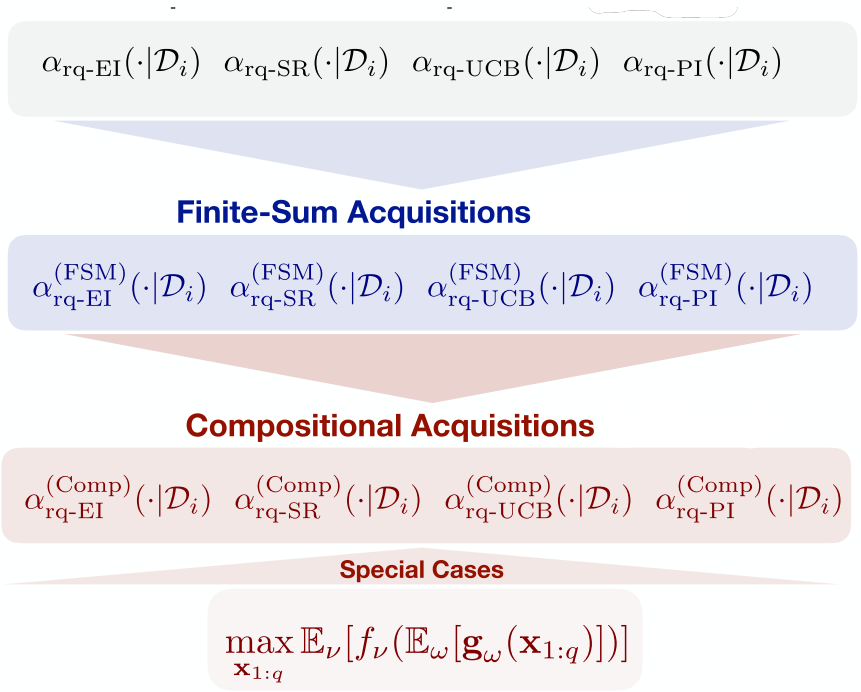


Figure 2.1.: Hierarchy of acquisition function derivations in this thesis. Acquisitions lower in the hierarchy may be written as instances of those above them in the hierarchy. We have the top, most encompassing, form of acquisition function studied in this work being Reparametrised Acquisition Function (Section 2.2), which encompasses Finite-Sum Acquisitions (Section 3.1.1), the middle form, which lastly encompasses Compositional Acquisitions (Section 3.1.1), the last form which is introduced by this thesis.

lowing us, under further technical conditions [252], to push gradients inside expectations when needed.

Before diving into ascent direction computation, we first present reparameterised acquisition formulations as derived in [252]. First, we realise that all batched acquisition functions in Section 2.1.2 involve an expectation over the GP's predictive posterior $\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}))$. Second, we recall that if a random variable is Gaussian distributed, one can reparameterise by choosing $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then applying $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\mathbf{z}) = \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}$ with $\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{L}_i^{\mathsf{T}}(\mathbf{x}_{1:q};\boldsymbol{\theta}) = \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$. Using such a deterministic transformation $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\mathbf{z})$, the original random variable's distribution remains unchanged indicating a mean $\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$ and covariance $\boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$. Now, we can easily replace $\boldsymbol{\lambda}_{\boldsymbol{\theta}}(\mathbf{z})$ in each of the expected improvement, simple regret, upper confidence bound, and entropy search acquisitions leading us to the following batch-reparameterised formulations

$$\alpha_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z} - f(\mathbf{x}_i^+)\mathbf{1}_q\right)\right\}\right], \tag{2.4}$$

$$\alpha_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}\right\}\right], \tag{2.5}$$

$$\alpha_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}|\right\}\right]. \tag{2.6}$$

When it comes to probability of improvement, the direct insertion of $\lambda_{\boldsymbol{\theta}}(\mathbf{z})$ into Equation (2.3) is difficult due to the discrete nature of the utility measure that violates differentiablity assumptions in reparameterisation [114]. To overcome this issue, we follow [252] and adopt the concrete (continuous to discrete) approximation to replace the discontinuous mapping [153] such that transformed and original variables are close in distribution. Sticking to the formulation presented [252], we loosen the indicator part of $\alpha_{\text{q-PI}}(\cdot)$ from Equation (2.3) and write

$$\max_{j\in 1:q}\left\{\mathbb{1}\{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q\}\right\} \approx \max_{j\in 1:q}\left\{\text{Sig}\left(\frac{\mathbf{f}(\mathbf{x}_{1:q}) - f(\mathbf{x}_i^+)\mathbf{1}_q}{\tau}\right)\right\},$$

with the component-wise sigmoid function $\text{Sig}(\cdot)$ and temperature parameter $\tau \in \mathbb{R}_+$. It yields an exact approximation as $\tau \to 0$. Given the approximation above and using a multivariate standard normal (instead of a uniform, see [153]) as $\hat{\mathcal{P}}(\mathbf{z})$, we derive the following reparameterised form for the probability of improvement acquisition

$$\alpha_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\max_{j\in 1:q}\left\{\text{Sig}\left(\frac{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z} - f(\mathbf{x}_i^+)\mathbf{1}_q}{\tau}\right)\right\}\right]. \tag{2.7}$$

Finally, for the entropy search acquisition function, the above reparametrisation trick should be applied twice: for the outer posterior distribution $\mathbf{f}(\mathbf{x}_{1:q})|\mathcal{D}_i, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}), \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}))$ and for the inner posterior distribution $\mathbf{f}\left(\mathbf{x}^{(\mathrm{g})}_{1:u}\right)|\mathcal{D}_i \cup \{\mathbf{x}_{1:q};\mathbf{f}(\mathbf{x}_{1:q})\}, \boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\mathbf{f}(\mathbf{x}_{1:q}),\boldsymbol{\theta}), \boldsymbol{\Sigma}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\theta}))$ with

$$\boldsymbol{\mu}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\mathbf{f}(\mathbf{x}_{1:q}),\boldsymbol{\theta}) = \underbrace{\mathbf{K}^{(i)}_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathrm{g})}_{1:u}, \mathcal{D}_i \cup \mathbf{x}_{1:q})}_{\mathbf{K}^{(\mathrm{g}),(i)}_{\boldsymbol{\theta}}} \begin{bmatrix} \mathbf{K}^{(i)}_{\boldsymbol{\theta}} + \sigma^2\mathbf{I} & \mathbf{K}^{(i)}_{\boldsymbol{\theta}}(\mathcal{D}_i, \mathbf{x}_{1:q}) \\ \mathbf{K}^{(i),\mathsf{T}}_{\boldsymbol{\theta}}(\mathcal{D}_i, \mathbf{x}_{1:q}) & \mathbf{K}^{(i)}_{\boldsymbol{\theta}}(\mathbf{x}_{1:q}, \mathbf{x}_{1:q}) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y}_{1:n_i} \\ \mathbf{f}(\mathbf{x}_{1:q}) \end{bmatrix},$$

$$\boldsymbol{\Sigma}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\theta}) = \mathbf{K}^{(i)}_{\boldsymbol{\theta}}(\mathbf{x}^{(\mathrm{g})}_{1:u}, \mathbf{x}^{(\mathrm{g})}_{1:u}) - \mathbf{K}^{(\mathrm{g}),(i)}_{\boldsymbol{\theta}} \begin{bmatrix} \mathbf{K}^{(i)}_{\boldsymbol{\theta}} + \sigma^2\mathbf{I} & \mathbf{K}^{(i)}_{\boldsymbol{\theta}}(\mathcal{D}_i, \mathbf{x}_{1:q}) \\ \mathbf{K}^{(i),\mathsf{T}}_{\boldsymbol{\theta}}(\mathcal{D}_i, \mathbf{x}_{1:q}) & \mathbf{K}^{(i)}_{\boldsymbol{\theta}}(\mathbf{x}_{1:q}, \mathbf{x}_{1:q}) \end{bmatrix}^{-1} \mathbf{K}^{(\mathrm{g}),(i),\mathsf{T}}_{\boldsymbol{\theta}}.$$

Due to the nested expectation structure of the entropy search acquisition function $\alpha_{\text{q-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, in order to rewrite it in the reparametrised form we consider two deterministic transformations. The two deterministic transformations we consider are $\boldsymbol{\lambda}^{(i)}_{\boldsymbol{\theta}}(\mathbf{z}) = \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}$ with Cholesky decomposition $\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{L}^{\mathsf{T}}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) = \boldsymbol{\Sigma}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})$ and $\boldsymbol{\varrho}^{(i)}_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \boldsymbol{\mu}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\mathbf{f}(\mathbf{x}_{1:q}),\boldsymbol{\theta}) + \mathbf{L}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\theta})\boldsymbol{\omega}$ with Cholesky decomposition $\mathbf{L}^{(\mathrm{g})}_i(\mathbf{x}_{1:u};\boldsymbol{\theta})\mathbf{L}^{(\mathrm{g}),\mathsf{T}}_i(\mathbf{x}_{1:u};\boldsymbol{\theta}) = \boldsymbol{\Sigma}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\theta})$. Choosing random vectors $\mathbf{z} \sim \mathcal{N}(\mathbf{0}_q, \mathbf{I}_{q\times q})$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$[2] in the above transformations $\boldsymbol{\lambda}^{(i)}_{\boldsymbol{\theta}}(\cdot)$, $\boldsymbol{\varrho}^{(i)}_{\boldsymbol{\theta}}(\cdot)$ respectively, and applying the following smooth approximation for the step function

$$\mathbb{1}\{\mathbf{f}(\mathbf{x}^{(\mathrm{g})}_{1:u}) - \max_{j\in 1:u} f(\mathbf{x}^{(\mathrm{g})}_j)\mathbf{1}_u\} \approx \mathrm{SM}\left(\frac{\boldsymbol{\mu}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\lambda}^{(i)}_{\boldsymbol{\theta}}(\mathbf{z}),\boldsymbol{\theta}) + \mathbf{L}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)$$

with softmax function $\mathrm{SM}(\cdot)$ and a temperature parameter $\tau \in \mathbb{R}_+$ controlling the approximation accuracy. Using the above, we now arrive to the batch-reparametrised form for the entropy search acquisition function

$$\alpha_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = -\mathbb{E}_{\mathbf{z}}\left[\mathrm{H}\left[\mathbb{E}_{\boldsymbol{\omega}}\left[\mathrm{SM}\left(\frac{\boldsymbol{\mu}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\lambda}^{(i)}_{\boldsymbol{\theta}}(\mathbf{z}),\boldsymbol{\theta}) + \mathbf{L}^{(\mathrm{g})}_i(\mathbf{x}^{(\mathrm{g})}_{1:u};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)\right]\right]\right], \quad (2.8)$$

Given reparameterised acquisitions, we now turn our attention to ERM- and FSM-BO depicting both implementations. We present novel compositional procedures that are sample and memory efficient.

[2] Here $\mathbf{0}_a$ and $\mathbf{I}_{a\times a}$ denote $a-$dimensional vector of zeros and a by a identity matrix respectively

2.2.1. ERM-BO using Stochastic Optimisation

Mainstream implementations of BO cast the inner optimisation problem (in Algorithm 1) in an empirical risk form $\max_{\mathbf{x}_{1:q}} \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\mathcal{L}(\mathbf{x}_{1:q};\mathbf{z})]$ with $\mathcal{L}(\mathbf{x}_{1:q};\mathbf{z})$ dependent on the acquisition's type, e.g., $\max_{j\in 1:q}\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}\}$ in the simple regret case. Such a connection enables tractable optimisation through the usage of numerous zero, first, and second-order optimisers developed in the literature [194, 32, 229]. Since such an implementation is fairly common in practice [131, 17] and not to burden the reader with unnecessary notation, we defer the exact details of the optimisers used in our experiments to [91]. Here, we briefly mention that we surveyed three zero-order optimisers, eight first-order algorithms and one well-known approximate second-order method.

Zeroth-Order Optimisers in ERM-BO: Zeroth-order methods optimise objectives based on function value information and have emerged from many different fields. In the online learning literature, for example, development of zeroth-order methods is mostly theoretical aiming at efficient and optimal regret guarantees [101, 142, 72] – a challenging topic in itself. Empirical successes of such procedures have been achieved in isolated instances [213, 242, 50, 43, 35, 12, 98]. Mainstream implementation of zeroth-order optimisers for BO, however, are of the evolutionary type updating generations of $\mathbf{x}$ through a process of adaptation and mutation [20].

In our experiments, we used three such strategies, varying from simple to advanced. The most simple among the three was random search (RS) which acts as a low-memory, low-compute baseline. The second, corresponds to a covariance matrix evolutionary strategy (CMA-ES) that generates updates of the mean and covariance of a multivariate normal based on average sample ranks gathered from function value information [100, 194]. The third and final algorithm was differential evolution (DE) which is widely considered a go-to in evolutionary optimisation [185, 16], e.g., NSGA I and II [58] as implemented in [28]. DE continuously updates a population of candidate solutions via component-wise mutation performing selection according to a mutation probability p_{mutation}. More details are available in [91].

First-Order Optimisers in ERM-BO: First-order optimisation techniques rely on gradient information to compute updates of $\mathbf{x}$. They are iterative in nature running for a total

of T iterations and executing a variant of the following rule at each step[3]

$$\underbrace{\mathbf{x}_{1:q,t+1} = \delta_t \mathbf{x}_{1:q,t} + \eta_t \frac{\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \left\{\beta_k^{(1)}\right\}_{k=0}^{t}\right)}{\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \left\{\beta_k^{(2)}\right\}_{k=0}^{t}, \epsilon\right)}}_{\text{(General update)}} \tag{2.9}$$

where δ_t is a weighting that depends on the type of algorithm used, η_t is a typically decaying learning rate, $\boldsymbol{\phi}_t^{(1)}(\cdot)$ and $\boldsymbol{\phi}_t^{(2)}(\cdot)$ are history-dependent mappings that vary between algorithms with the ratio executed element-wise, $\left\{\beta_k^{(1)}\right\}_{k=0}^{t}$ and $\left\{\beta_k^{(2)}\right\}_{k=0}^{t}$ are history-weighting parameters, and ϵ a small positive constant used to avoid division by zero. Additionally, $\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}$ represent sub-sampled gradient estimators that are acquired using Monte-Carlo samples of $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. It is also worth noting that differentiating through the $\max$ operator that appears in all acquisitions can be performed either using sub-gradients or by propagating through the max value of the corresponding vector.

To elaborate our generalised form, we realise that one can easily recover Adam's [126] update equation by setting $\delta_1 = \cdots = \delta_T = 1$, $\beta_1^{(1)} = \cdots = \beta_T^{(1)} = \beta_1$, $\beta_1^{(2)} = \cdots = \beta_T^{(2)} = \beta_2$, and $\boldsymbol{\phi}_t^{(1)}$ and $\boldsymbol{\phi}_t^{(2)}$ to

$$\boldsymbol{\phi}_t^{(1)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}, \beta_1\right) = \frac{1-\beta_1}{1-\beta_1^t}\sum_{k=0}^{t}\beta_1^k \overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\overline{\nabla\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)}^2, \ldots, \overline{\nabla\alpha(\mathbf{x}_{1:q,t}|\mathcal{D}_i)}^2, \beta_2, \epsilon\right) = \sqrt{\frac{1-\beta_2}{1-\beta_2^t}\sum_{k=0}^{t}\beta_2^k \overline{\nabla\alpha(\mathbf{x}_{1:q,t-k}|\mathcal{D}_i)}^2} + \epsilon.$$

Of course, Adam is yet another special case of Equation (2.9). For notational convenience, we defer the detailed derivations of other optimisers including SGA [196], RProp [193], RMSprop [106], AdamW [151], AdamOS (an Adam adaptation with new hyperparameters that we propose in this thesis), AdaGrad [62], and AdaDelta [261] to [91].

[3]For simplicity in the notation for acquisition functions $\alpha(\mathbf{x}_{1:q,0}|\mathcal{D}_i)$ we drop the subscript with the type.

3. CompBO: Compositional Bayesian Optimisation

In this chapter we will detail our first contribution, a new family of compositional acquisition functions [91]. Firstly, in Section 3.1 we will introduce the compositional form of popular acquisition functions, building on the background introduced in Section 2.1. In Section 3.2 we analyse experimental results from compositional vs non-compositional acquisition functions and lastly summarise the work in the conclusions in Section 3.3.

3.1. Introduction

Bayesian optimisation is a method for optimising black-box objective functions [140, 163, 117]. The black-box optimisation (BBO) problem describes the search for the global maximiser $\mathbf{x}^*$ of an unknown objective function $f(\mathbf{x})$. The objective function is unknown in the sense that an analytical form is unavailable. However, the objective may still be evaluated pointwise at arbitrary query locations within the bounds of the design space. A further characteristic of the BBO problem is that each query is expensive in terms of time, and as such, it is desirable to query as few points as possible in the search for the global maximiser.

Real world examples of BBO problems are ubiquitous. Illustrative examples include hyperparameter tuning in machine learning [66, 121, 249, 72], where the black-box objective is the mapping between a set of model hyperparameters $\mathbf{x}$ and the validation set performance $f(\mathbf{x})$, as well as automatic chemical design [84, 134, 165, 87], where the black-box objective is the mapping between a molecule $\mathbf{x}$ and its suitability as a drug candidate $f(\mathbf{x})$. Further examples of BBO problems appear as subroutines of optimisation algorithms such as immune optimisation [262, 154], ant colony optimisation [260, 224] and genetic algorithms [176], in reinforcement learning when accounting for safety [53, 2], in multi-agent systems to compute Nash equilibria [257, 11], in speech recognition [166] and more broadly across domains spanning architecture [51], chemical engineering [180] and biology [210, 164].

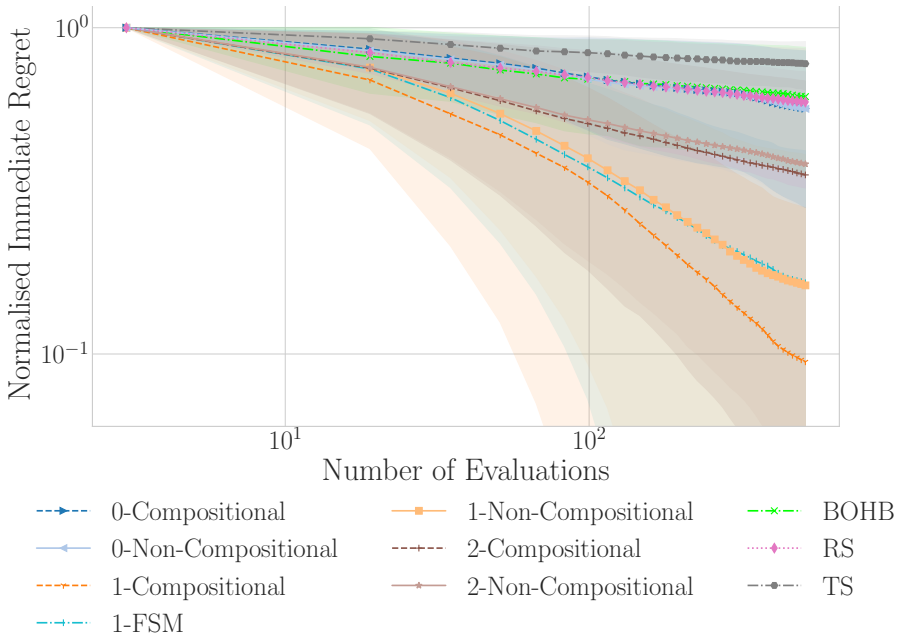


Figure 3.1.: Summary plot for 3100 synthetic BBO experiments. We average for each seed over all optimisers within that class and plot the normalised regret across each seed for each class. Note for certain optimisers that are not an nth-order compositional/ non-computational optimiser, for any n, we allow them to represent themselves (e.g. BOHB, TS, RS etc). Lower regret is indicative of better performance. This experiment shows that first-order compositional optimisers outperform others, whilst first-order non-compositional and BOHB perform similarly well, with the worst performing method being TS.

Various strategies exist for optimising black-box objective functions including zero-order methods [241, 90, 72], resource allocation methods [145, 66] and surrogate model-based methods [219, 212, 71]. In this paper, we focus on Bayesian optimisation, a sequential, data-efficient, surrogate model-based approach that is particularly effective when function evaluations are costly. The two core components of the Bayesian optimisation algorithm are a probabilistic surrogate model and an acquisition function. The probabilistic surrogate model facilitates data efficiency by making use of the full optimisation history to represent the black-box function and additionally leverages uncertainty estimates to guide exploration. Given that the true sequential risk describing the optimality of a sequence of queries is computationally intractable, an acquisition function is a myopic heuristic which

acts as a proxy to the true sequential risk. The acquisition function measures the utility of a query point $\mathbf{x}$ by its mean value under the surrogate model (exploitation) as well as its uncertainty under the surrogate model (exploration). At each round of the Bayesian optimisation algorithm, the acquisition function is maximised to select the next query point.

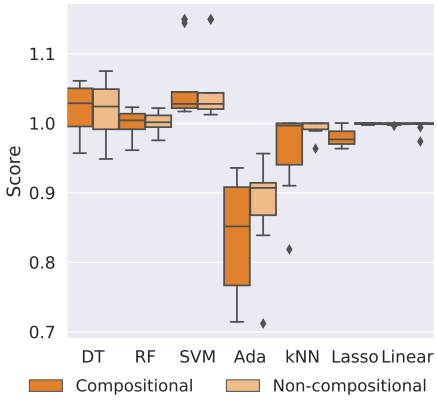


Figure 3.2.: Bayesmark regression summary amalgamating the results from 54 Bayesmark regression tasks, we compare compositional and non-compositional optimisers. Higher score is better. Boxplots show median, lower and upper quartiles of the scores. Results show compositional optimisers outperforming non-compositional optimisers on half the tasks and vice versa for non-compositional optimisers.

It has been argued that maximisation of the acquisition function is an important, yet neglected determinant of the performance of Bayesian optimisation schemes [252]. The vast majority of acquisition functions however, constitute a serious challenge from the standpoint of optimisation; a characteristic exacerbated in the batch setting, where acquisition functions are routinely non-convex, high-dimensional and intractable [252]. Many strategies exist for optimising acquisition functions including gradient-based methods [62, 106, 126], evolutionary methods [111, 116, 99] as well as variations of random search [207, 202, 21]. In this work, we choose to focus on gradient-based methods which were recently shown to be highly effective for optimising a wide class of Monte Carlo acquisition functions [252].

The most commonly-used acquisition functions in practical applications [219] are Monte Carlo acquisition functions in the sense that they are formulated as integrals with respect to the current probabilistic belief over the unknown function f [212, 252]; these integrals are typically intractable and as such are approximated by the corresponding Monte Carlo

(MC) estimate. In order to admit gradient-based optimisation, a reparametrisation trick [127, 192], introduced first as infinitesimal perturbation analysis [42, 82], is applied to facilitate differentiation through the MC estimates with respect to the parameters of the surrogate model. It was shown in [252] that acquisition functions estimated via MC integration are consistently amenable to gradient-based optimisation via standard first and second-order methods including SGA [31], Adam [126], RMSprop [106], AdaGrad [62] and L-BFGS-B [263].

In this work, we exploit the observation that most common acquisition functions exhibit compositional structure and hence can be equivalently reformulated in a compositional form [245]. Such a reformulation allows a broader class of optimisation techniques to be applied for acquisition function optimisation [240, 80, 247] and in practice can more often enable better numerical performance to be achieved in comparison with standard first and second-order methods. The compositional form is achieved for the expected improvement (EI), simple regret (SR), upper confidence bound (UCB) and probability of improvement (PI) acquisition functions by first exposing the finite-sum form of the reparameterised acquisition functions derived by [252] and second introducing a deterministic outer function when considering the problem from a matrix-vector perspective. It should be noted that reformulating the acquisition function in a compositional form is distinct from the setting where the black-box function has a compositional form [13]. In order

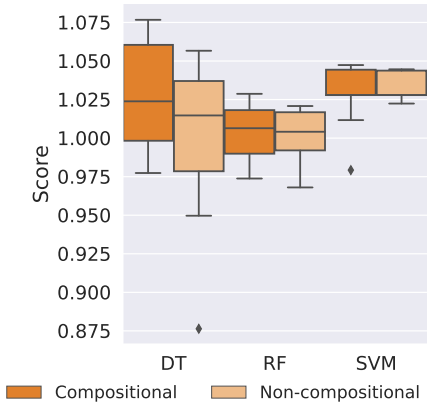


Figure 3.3.: Bayesmark classification summary amalgamating the results from 54 Bayesmark classification tasks, we compare compositional and non-compositional optimisers. Higher score is better. Boxplots show lower, median and upper quartiles of the data. Results show compositional optimisers outperforming non-compositional optimisers across all tasks.

to both improve and analyse the optimisation performance on the compositional form of the acquisition function, we introduce several algorithmic adaptations. Firstly, we present (C)L-BFGS; a modification to the L-BFGS algorithm to enable the handling of nested compositional forms. Secondly, we develop AdamOS, a variant of the Adam optimiser [126] which borrows the hyperparameter settings of CAdam [240] and facilitates performance comparison between compositional and non-compositional optimisers. Lastly, we formulate a generalised iterative update rule for first-order compositional optimisers and show how the updates of a number of first-order optimisers may be expressed in this manner. In our empirical study, we seek to identify the most effective means of optimising the acquisition function under a range of experimental conditions including input dimensionality, presence or absence of observation noise and choice of acquisition function. We investigate twenty-eight optimisation schemes, spanning zeroth, first and second-order optimisers as well as both compositional and non-compositional methods. Additionally, we seek to answer the following questions: Are there benefits to the finite-sum formulation of the reparameterised acquisition functions compared to the more frequently-encountered empirical risk minimisation formulation? Are compositional or non-compositional approaches to optimisation more effective and if so, under what conditions are they more effective? What are the performance-related trade-offs in memory-efficient implementations of compositional acquisition functions? How does the wall-clock time of compositional optimisation methods compare to non-compositional optimisation methods, and how does this vary with the dimensionality of the input space? How do compositional optimisers fare when faced with noisy observations?

In order to answer these questions, we first perform a set of experiments across five noiseless synthetic function tasks. Using this set of noiseless experiments as a filter for the most effective optimisers, we then perform a second set of experiments on the Bayesmark datasets which are noisy and bear a closer resemblance to real-world problems than the synthetic tasks. Our results for the synthetic experiments are summarised in Figure 3.1 whilst our results for the Bayesmark datasets are summarised in Figure 3.2 and Figure 3.3 for the regression and classification challenges respectively. In sum total, our empirical study comprises 3958 individual experiments.

The paper is organised as follows: First, we introduce the necessary background on the Bayesian optimisation framework. Second, we hone in on the acquisition function maximisation subroutine of Bayesian optimisation with the intent to understand the efficacy of compositional optimisation schemes. We provide a general overview of compositional optimisation and derive compositional forms for the four most popular myopic acquisition functions. Third, we discuss state-of-the-art compositional solvers, namely CAdam, NASA, SCGA and ASCGA. Fourth, we detail our experimental setup and present the empirical results. Fifth, we analyse the experimental results, draw conclusions and

indicate avenues for future work as well as descriptions of open problems in acquisition function maximisation.

3.1.1. FSM-BO & Connections to Compositional Optimisation

Rather than considering the problem of acquisition function maximisation as an instance of empirical risk minimisation, we can follow an alternative route and focus on finite sum approximations. To do so, imagine we acquire M independent and identically-distributed samples from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, $\{\mathbf{z}_m\}_{m=1}^M$, upfront before the beginning of any acquisition function optimisation step. Assuming fixed samples for now, we can write finite-sum forms of the reparameterised acquisition functions (those from Section 2.2) using a simple Monte Carlo estimator as follows

$$\alpha_{\text{rq-EI}}^{(\text{FSM})}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right)\right\}, \quad (3.1)$$

$$\alpha_{\text{rq-SR}}^{(\text{FSM})}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m\right\}, \quad (3.2)$$

$$\alpha_{\text{rq-UCB}}^{(\text{FSM})}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m|\right\}, \quad (3.3)$$

$$\alpha_{\text{rq-PI}}^{(\text{FSM})}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\text{Sig}\left(\frac{\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q}{\tau}\right)\right\}. \quad (3.4)$$

As for the entropy search acquisition function given in Equation (2.8), due to its nested expectation form simply replacing both expectations with their corresponding MC estimates leads to a biased estimate of $\alpha_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$. Instead, we use a collection of independent random vectors $\{\mathbf{z}_m\}_{m=1}^M$ sampled from $\mathcal{N}(\mathbf{0}_q, \mathbf{I}_{q\times q})$ to construct a Monte Carlo estimate for the outer expectation

$$\alpha_{\text{rq-ES}}^{(\text{FSM})}(\mathbf{x}_{1:q}|\mathcal{D}_i) = -\frac{1}{M}\sum_{m=1}^{M}\text{H}\left[\mathbb{E}_{\boldsymbol{\omega}}\left[\text{SM}\left(\frac{\boldsymbol{\mu}_i^{(\text{g})}(\mathbf{x}_{1:u}^{(\text{g})};\boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}_m),\boldsymbol{\theta}) + \mathbf{L}_i^{(\text{g})}(\mathbf{x}_{1:u}^{(\text{g})};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right)\right]\right]. \quad (3.5)$$

At this stage, we can execute any off-the-shelf optimiser to maximise the finite sum version of the acquisitions as shown in Section 2.2.1. Contrary to ERM-BO which samples new $\mathbf{z}$ vectors at each iteration, the FSM formulation fixes $\{\mathbf{z}_m\}_{m=1}^M$ and mini-batches from this

fixed pool to compute necessary gradients and Hessian estimates for first and second-order methods respectively. At first sight, one might believe that ERM and FSM are the only plausible approximation forms of acquisition functions in BO. Upon further investigation, however, we realise that finite sum myopic acquisitions adhere to yet another configuration that is still to be (well-) explored in the literature. Not only does this new form allow for novel solvers not yet attempted in acquisition function maximisation, but also seems to significantly outperform both ERM-and FSM-BO in practice, cf. Section 3.2.

Comp-BO: A Compositional Form for Myopic Acquisition Functions

Recently, the optimisation community has displayed an increased interest in developing specialised algorithms for compositional (or nested) objectives due to their prevalence in subfields of machine learning, e.g., in model-agnostic-meta-learning [240], semi-implicit variational inference [259], dynamic programming and reinforcement learning [247]. In each of these examples, compositional solvers have demonstrated efficiency advantages when compared to other algorithms which begs the question as to whether these improvements can be ported to Bayesian optimisation.

From a definition perspective, compositional problems involve maximising an objective that consists of a non-linear nesting of expectations of random variables:

$$\max_{\mathbf{x}_{1:q}} \mathbb{E}_{\nu}[f_{\nu}(\mathbb{E}_{\omega}[\mathbf{g}_{\omega}(\mathbf{x}_{1:q})])], \tag{3.6}$$

with (not necessarily iid) random variables ν and ω are sampled from $\mathcal{P}_{\nu}(\cdot)$ and $\mathcal{P}_{\omega}(\cdot)$, respectively [246], stochastic function $f_{\nu}(\cdot)$, and stochastic map $\mathbf{g}_{\omega}(\cdot)$. Hence to benefit from such techniques, our first step consists of transforming the finite-sum versions of the acquisition functions above into a composed (or nested) form that abides by the structure in Equation (3.6). Interestingly, this can easily be achieved if we look at the problem from a matrix-vector perspective. To illustrate, consider $\alpha^{(\text{FSM})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$ and define $\mathbf{g}^{(\text{EI})}_{\omega}(\mathbf{x}_{1:q})$ to be a $q \times M$ matrix such that the ω^{th} column is set to $\mathbf{v}^{(\text{EI})}_{\omega} = \text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_{\omega} - f(\mathbf{x}_i^+)\mathbf{1}_q\right) \in \mathbb{R}^q$ with ω uniformly distributed in $[1:M]$, and set the other columns to $\mathbf{0}_q$

$$\mathbf{g}^{(\text{EI})}_{\omega}(\mathbf{x}_{1:q}) = [\mathbf{0}_q, \ldots, \mathbf{v}^{(\text{EI})}_{\omega}, \ldots, \mathbf{0}_q].$$

Clearly, if we consider the expectation with respect to $\omega \sim \text{Uniform}([1:M])$, we arrive at the following matrix that sums all information across $\{\mathbf{z}_m\}_{m=1}^M$

$$\mathbb{E}_{\omega}[\mathbf{g}^{(\text{EI})}_{\omega}(\mathbf{x}_{1:q})] = \frac{1}{M}[\mathbf{v}^{(\text{EI})}_1, \ldots, \mathbf{v}^{(\text{EI})}_m, \ldots, \mathbf{v}^{(\text{EI})}_M],$$

with a q-dimensional vector $\mathbf{v}_m^{(\text{EI})} = \text{ReLU}\left(\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right)$. To attain the original form of $\alpha_{\text{rq-EI}}^{(\text{FSM})}(\cdot)$, we further introduce a deterministic outer function $f^{(\text{EI})}: \mathbb{R}^{q\times M} \to \mathbb{R}$ as follows

$$\alpha_{\text{rq-EI}}^{(\text{Comp})}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}_\omega^{(\text{EI})}(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\mathbf{v}_m^{(\text{EI})} = \alpha_{\text{rq-EI}}^{(\text{FSM})}.$$

Importantly, the above shows that a finite-sum expected improvement acquisition can be written in a compositional (nested) form with $\alpha_{\text{rq-EI}}^{(\text{FSM})} = f(\mathbb{E}_\omega[\mathbf{g}_\omega(\mathbf{x})])$. In our derivations, we have considered a deterministic outer function $f(\cdot)$ leading us to a special case of Equation (3.6) where $P_\nu(\cdot)$ is Dirac. Such a consideration is mostly due to the fact that q is typically in the order of tens or hundreds in BO allowing for exact outer summations. In the case of large batch sizes, our formulation can easily be generalised to a stochastic setting exactly matching a compositional form as shown in [91].

Following the same strategy above, we can now reformulate all other acquisition functions as instances of compositional optimisation. Next, we list these results and refer the reader to [91] for a detailed exposition. First, we choose $\omega \sim \text{Uniform}([1:M])$ and then consider the following inner matrix mappings

$$\begin{aligned}
\mathbf{g}_\omega^{(\text{PI})}(\mathbf{x}_{1:q}) &= [\mathbf{0}_q, \ldots, \mathbf{v}_\omega^{(\text{PI})}, \ldots, \mathbf{0}_q] \in \mathbb{R}^{q\times M},\\
\mathbf{g}_\omega^{(\text{SR})}(\mathbf{x}_{1:q}) &= [\mathbf{0}_q, \ldots, \mathbf{v}_\omega^{(\text{SR})}, \ldots, \mathbf{0}_q] \in \mathbb{R}^{q\times M},\\
\mathbf{g}_\omega^{(\text{UCB})}(\mathbf{x}_{1:q}) &= [\mathbf{0}_q, \ldots, \mathbf{v}_\omega^{(\text{UCB})}, \ldots, \mathbf{0}_q] \in \mathbb{R}^{q\times M},
\end{aligned}$$

and for the entropy search acquisition $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$

$$\mathbf{g}_{\boldsymbol{\omega}}^{(\text{ES})}(\mathbf{x}_{1:q}) = [\mathbf{v}_{1,\boldsymbol{\omega}}^{(\text{ES})}, \mathbf{v}_{2,\boldsymbol{\omega}}^{(\text{ES})}, \ldots, \mathbf{v}_{M,\boldsymbol{\omega}}^{(\text{ES})}] \in \mathbb{R}^{u\times M}.$$

Here the $q-$dimensional vectors $\mathbf{v}_m^{(\text{PI})}, \mathbf{v}_m^{(\text{SR})}$, and $\mathbf{v}_m^{(\text{UCB})}$ are defined as (for $m \in [1:M]$)

$$\begin{aligned}
\mathbf{v}_m^{(\text{PI})} &= \frac{1}{\tau}\left[\boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m - f(\mathbf{x}_i^+)\mathbf{1}_q\right],\\
\mathbf{v}_m^{(\text{SR})} &= \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m,\\
\mathbf{v}_m^{(\text{UCB})} &= \boldsymbol{\mu}_i(\mathbf{x}_{1:q};\boldsymbol{\theta}) + \sqrt{\beta\pi/2}\,\left|\mathbf{L}_i(\mathbf{x}_{1:q};\boldsymbol{\theta})\mathbf{z}_m\right|.
\end{aligned}$$

and $u-$dimensional vector $\mathbf{v}_{m,\boldsymbol{\omega}}^{(\text{ES})}$ is defined as (for $m \in [1:M]$ and $\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}_u, \mathbf{I}_{u\times u})$)

$$\mathbf{v}_{m,\boldsymbol{\omega}}^{(\text{ES})} = \text{SM}\left(\frac{\boldsymbol{\mu}_i^{(\text{g})}(\mathbf{x}_{1:u}^{(\text{g})};\boldsymbol{\lambda}_{\boldsymbol{\theta}}^{(i)}(\mathbf{z}_m),\boldsymbol{\theta}) + \mathbf{L}_i^{(\text{g})}(\mathbf{x}_{1:u}^{(\text{g})};\boldsymbol{\theta})\boldsymbol{\omega}}{\tau}\right).$$

Now, properly selecting the outer functions $f^{(\text{PI})}(\cdot)$, $f^{(\text{SR})}(\cdot)$, and $f^{(\text{UCB})}(\cdot)$ gives us

$$\alpha^{(\text{Comp})}_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\text{PI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{PI})}_\omega(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\text{Sig}\left(\mathbf{v}^{(\text{PI})}_m\right)\right\} = \alpha^{(\text{FSM})}_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i),$$

$$\alpha^{(\text{Comp})}_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\text{SR})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{SR})}_\omega(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\mathbf{v}^{(\text{SR})}_m\right\} = \alpha^{(\text{FSM})}_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i),$$

$$\alpha^{(\text{Comp})}_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\text{UCB})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{UCB})}_\omega(\mathbf{x}_{1:q})]) = \frac{1}{M}\sum_{m=1}^{M}\max_{j\in 1:q}\left\{\mathbf{v}^{(\text{UCB})}_m\right\} = \alpha^{(\text{FSM})}_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i).$$

Finally, properly selecting the stochastic outer function $f^{(\text{ES})}_\nu(\cdot)$ with $\nu \sim \text{Uniform}([1:M])$ gives us

$$\alpha^{(\text{Comp})}_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_\nu\left[f^{(\text{ES})}_\nu\left(\mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{g}^{(\text{ES})}_{\boldsymbol{\omega}}(\mathbf{x}_{1:q})\right]\right)\right] = -\frac{1}{M}\sum_{m=1}^{M}\text{H}\left[\mathbb{E}_{\boldsymbol{\omega}}\left[\mathbf{v}^{(\text{ES})}_{m,\boldsymbol{\omega}}\right]\right] = \alpha^{(\text{FSM})}_{\text{rq-ES}}(\mathbf{x}_{1:q}|\mathcal{D}_i).$$

Clearly, the results above recover the formulations of the acquisition functions given in Equations 3.2 - 3.4 while making them amenable to compositional solvers, a new class of optimisers not yet well-studied in the Bayesian optimisation literature. We detail such compositional optimisers next.

Zeroth-Order Compositional Solvers for BO: Of course, the compositional forms presented above are still suitable for zeroth-order methods (Section 2.2.1). The distinguishing factor from non-compositional forms is the evaluation process of nested objectives which requires careful consideration. In the case of $\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, for example, the inner expectation $\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x})]$ in Equation 3.6 can be evaluated using a Monte Carlo approximation

$$\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})] \approx \frac{1}{K}\sum_{m=1}^{K}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q}), \quad \text{with } K < M \text{ being a mini-batch of } \{\mathbf{z}_m\}_{m=1}^{M}.$$

Furthermore, the outer function is estimated by $f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]) \approx f^{(\text{EI})}\left(\frac{1}{K}\sum_{m=1}^{K}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q})\right)$, where such an estimate asymptotically ($K \to \infty$) converges to the true expectation due to the continuity of $f^{(\text{EI})}(\cdot)$

$$\lim_{K\to\infty} f^{(\text{EI})}\left(\frac{1}{K}\sum_{m=1}^{K}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q})\right) = f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]).$$

Clearly, this observation allows us to straightforwardly apply any of the three considered zero-order methods (CMA-ES, DE, and RS) for determining updates of $\mathbf{x}_{1:q}$. Certainly, such Monte Carlo approximations are not distinctive for $\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, allowing us to follow the same scheme for $\alpha^{(\text{Comp})}_{\text{rq-PI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, $\alpha^{(\text{Comp})}_{\text{rq-SR}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, and $\alpha^{(\text{Comp})}_{\text{rq-UCB}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$.

First-Order Compositional Solvers for BO: In contrast to zeroth-order compositional methods, where the only difference between them and their non-compositional counterparts is in the evaluation of the objective function, first-order compositional optimisers require more sophisticated techniques due to the difficulty associated in acquiring unbiased gradients of nested objectives. To elaborate, let us carry on with our running example and consider the gradient of $\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = f^{(\text{EI})}(\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})])$. Using the chain rule, we can easily see that such a gradient involves a product of the Jacobian of $\mathbf{g}_\omega(\mathbf{x}_{1:q})$ with the gradient of $f^{(\text{EI})}(\cdot)$ that is to be evaluated around the inner mapping[1]

$$\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i) = \mathbb{E}_\omega[\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]^\intercal \nabla_{\boldsymbol{\zeta}} f^{(\text{EI})}(\boldsymbol{\zeta}) \mid_{\boldsymbol{\zeta}=\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]},$$

with an unrolled vector across all dimensions d and batch sizes q denoted by $\text{vec}(\mathbf{x}_{1:q}) \in \mathbb{R}^{dq}$. When attempting to acquire an unbiased estimate of $\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}_{\text{rq-EI}}(\mathbf{x}_{1:q}|\mathcal{D}_i)$, we realise that the first term can be approximated by simple Monte Carlo

$$\mathbb{E}_\omega[\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})] \approx \frac{1}{K_1}\sum_{m=1}^{K_1} \nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}^{(\text{EI})}_{\omega_m}(\mathbf{x}_{1:q}),$$

with batch size $K_1 < M$. The second part, however, is tougher to estimate as it involves a gradient of a non-linear nesting of an expected value, i.e., $\nabla_{\boldsymbol{\zeta}} f^{(\text{EI})}(\boldsymbol{\zeta}) \mid_{\boldsymbol{\zeta}=\mathbb{E}_\omega[\mathbf{g}^{(\text{EI})}_\omega(\mathbf{x}_{1:q})]}$. To resolve this problem, in the compositional optimisation literature [245, 240], typically an auxiliary variable $\mathbf{u}$ is introduced and an exponentially-weighted average of $\boldsymbol{\zeta}$ is used, resulting in asymptotically-vanishing biases. To acquire such behaviour, not only do we need to update $\mathbf{x}_{1:q}$ but we also need to modify $\mathbf{u}$ and our estimation of $\boldsymbol{\zeta}$. As such, most compositional solvers execute three subroutines (main $\mathbf{x}_{1:q}$, auxiliary $\mathbf{u}$ and $\boldsymbol{\zeta}$) between iterations t and $t+1$ – the first to generate $\mathbf{x}_{1:q,t+1}$, the second for $\mathbf{u}_{t+1}$ and the third for $\boldsymbol{\zeta}_{t+1}$. Rather than presenting every subroutine for all utilised algorithms across all acquisition functions, here we keep the exposition general and provide a set of unifying update rules,

[1] Of course, a simple solution corresponds to a Nested Monte Carlo approach that approximates both inner and outer mappings with samples from ω and ν and then executes standard off-the-shelf algorithms. In our experiments, we make use of such a technique which we refer to as Adam-Nested (see Section 3.2) but realise that dedicated first-order compositional solvers tend to outperform such a scheme.

deferring exact details to [91]. To that end, we introduce four history-dependent mappings $\phi_t^{(1)}(\cdot)$, $\phi_t^{(2)}(\cdot)$, $\phi_t^{(3)}(\cdot)$ and $\phi_t^{(4)}(\cdot)$. $\phi_t^{(1)}(\cdot)$ and $\phi_t^{(2)}(\cdot)$ act on sub-sampled gradient histories, and their corresponding squares, for updating $\mathbf{x}_{1:q,t}$ as follows

$$\mathbf{x}_{1:q,t+1} = \mathbf{x}_{1:q,t} + \eta_t \frac{\phi_t^{(1)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t}, \left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right)}{\phi_t^{(2)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t}, \left\{\gamma_k^{(2)}\right\}_{k=0}^{t}, \epsilon\right)}, \quad (3.7)$$

with learning rate η_t, history-dependent weightings that vary across algorithms $\{\gamma_k^{(1)}\}_{k=0}^t$ and $\{\gamma_k^{(2)}\}_{k=0}^t$. In Equation (3.7), we also denote compositional gradient estimates by $\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}$ that can be written as

$$\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)} = \left[\frac{1}{K_1}\sum_{m=1}^{K_1}\nabla_{\text{vec}(\mathbf{x}_{1:q})}\mathbf{g}_{\omega_m}^{(\text{type})}(\mathbf{x}_{1:q,k})\right]^{\mathsf{T}}\nabla_{\boldsymbol{\zeta}} f^{(\text{type})}(\boldsymbol{\zeta}_k), \quad (3.8)$$

with inner and outer mapping of a compositional formulation denoted by $\mathbf{g}_{\omega_m}^{(\text{type})}$ and $f^{(\text{type})}$ where type $\in \{\text{EI}, \text{PI}, \text{SR}, \text{UCB}\}$. With $\mathbf{x}_{1:q,t+1}$ computed, the next step is to update $\mathbf{u}_t$ and $\boldsymbol{\zeta}_t$ which can be achieved through $\phi_t^{(3)}(\cdot)$ and $\phi_t^{(4)}(\cdot)$ in the following manner

$$\mathbf{u}_{t+1} = \phi_{t+1}^{(3)}\left(\mathbf{x}_{1:q,0}, \ldots, \mathbf{x}_{1:q,t+1}, \{\beta_k\}_{k=0}^t\right), \quad (3.9)$$

$$\boldsymbol{\zeta}_{t+1} = \phi_{t+1}^{(4)}\left(\overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_1)}, \ldots, \overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_{t+1})}, \{\beta_k\}_{k=0}^t, \boldsymbol{\zeta}_0, \mathbf{u}_0\right), \quad (3.10)$$

with a set of free parameters[2] $\{\beta_k\}_{k=0}^t$, initialisations $\boldsymbol{u}_0$ and $\boldsymbol{\zeta}_0$ that in turn depend on $\mathbf{x}_{1:q,0}$. Furthermore, in Equation (3.10) for Monte Carlo estimate of the inner mapping we use $\overline{\mathbf{g}^{(\text{type})}(\cdot)}$

$$\overline{\mathbf{g}^{(\text{type})}(\cdot)} = \frac{1}{K_2}\sum_{m=1}^{K_2}\mathbf{g}_{\omega_m}^{(\text{type})}(\cdot),$$

[2]It is worth noting that in [91] we provide a complete set of all hyperparameters used across all 28 optimisers.

with batch size $K_2 < M$ and type $\in \{\text{EI}, \text{PI}, \text{SR}, \text{UCB}\}$. As an illustrative example, we note that one can recover CAdam [240] by instantiating the above as follows

$$\boldsymbol{\phi}_t^{(1)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}\right\}_{k=0}^{t}, \left\{\gamma_k^{(1)}\right\}_{k=0}^{t}\right) = \sum_{k=0}^{t}(1-\gamma_k^{[1]})\prod_{j=k+1}^{t}\gamma_j^{[1]}\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)},$$

$$\boldsymbol{\phi}_t^{(2)}\left(\left\{\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}^2\right\}_{k=0}^{t}, \left\{\gamma_k^{(2)}\right\}_{k=0}^{t}, \epsilon\right) = \sqrt{\sum_{k=0}^{t}(1-\gamma_k^{[2]})\prod_{j=k+1}^{t}\gamma_j^{[2]}\overline{\nabla_{\text{vec}(\mathbf{x}_{1:q})}\alpha^{(\text{Comp})}(\mathbf{x}_{1:q,k}, \boldsymbol{\zeta}_k|\mathcal{D}_i)}^2} + \epsilon,$$

$$\boldsymbol{\phi}_t^{(3)}\left(\mathbf{x}_{1:q,0}, \ldots, \mathbf{x}_{1:q,t}, \{\beta_k\}_{k=0}^{t-1}\right) = (1-\beta_{t-1}^{-1})\mathbf{x}_{1:q,t-1} + \beta_{t-1}^{-1}\mathbf{x}_{1:q,t},$$

$$\boldsymbol{\phi}_t^{(4)}\left(\overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_1)}, \ldots, \overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_t)}, \{\beta_k\}_{k=0}^{t-1}, \boldsymbol{\zeta}_0, \mathbf{u}_0\right) = \sum_{k=1}^{t}\beta_{k-1}\prod_{j=k}^{t-1}(1-\beta_j)\overline{\mathbf{g}^{(\text{type})}(\mathbf{u}_k)}.$$

Of course, CAdam is just an instance of the generic update rules presented in Equations 3.7- 3.10. Other first-order compositional methods, such as NASA [80], ASCGA [245], SCGA [245] and Adam applied to a nested Monte Carlo objective can all be derived from our general form as demonstrated in [91].
We will now discuss memory efficient implementations of all compositional acquisition functions introduced.

Memory-Efficient Implementations for Comp-BO

Although the ERM-BO and FSM-BO strategies discussed in Sections 2.2.1 and 3.1.1 share commonalities such as the sampling of the reparametrisation variable $\mathbf{z} \in \mathbb{R}^q$ and the use of Monte Carlo estimates, one important difference between the approaches is memory complexity - the total amount of space in storage (be that disk or cloud) needed for the complete execution of an optimisation method. It is worthwhile mentioning that the key difference between memory and time resources is that the former can be erased and reused multiple times while the latter cannot, and this distinction plays an important role in the analysis of applied optimisation algorithms.

For ERM-BO methods, the total amount of required memory is defined by the size of the largest mini-batch sampled during the execution and the memory needed for the iterative update. Since in all ERM-BO algorithms we use mini-batches of a constant size $K = 128$,

and at each iteration t we store only the current iterative value $\mathbf{x}_{1:q,t} \in \mathbb{R}^{dq}$ the overall memory complexity is therefore bounded by $\mathcal{O}(Kq + dq)$.

Similarly to empirically-founded techniques, in FSM-BO methods we also store at each step t the current value of the iterate $\mathbf{x}_{1:q,t} \in \mathbb{R}^{dq}$ and utilise a mini-batch of samplings of size $K \ll M$. However in contrast to the ERM-BO case, the upfront sampling of M reparameterisation random variables $\mathbf{z}$ used in the FSM-BO scenario leads to an $\mathcal{O}(Mq+dq)$ bound for the overall memory capacity. On one hand, large values of M are preferable as they provide a better approximation to the true acquisition functions given in Equations 2.4 - 2.7, yet on the other hand, such values of M make finite-sum methods memory stringent.

To remedy this problem, we propose memory-efficient adaptations of compositional methods: CAdam-ME, NASA-ME and Nested-MC-ME. In a nutshell, all these methods exploit the observation that at any given iteration, stochastic compositional optimisers only require uniform sub-sampling from the fixed collection of M reparametrisation variables $\mathbf{z}$. Hence instead of storing M samples upfront, one can draw K of them from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ at each iteration resulting in an overall memory complexity given by $\mathcal{O}(Kq + dq)$. For a detailed description of the memory-efficient methods CAdam-ME, NASA-ME and Nested-MC-ME, we refer the reader to [91].

3.2. Experiments & Results

Having presented a comprehensive set of optimisation techniques suitable for maximising acquisition functions, we now wish to systematically evaluate their empirical performance. Specifically, we design our experimental setup with the intention of answering the following questions:

1. Do Finite-Sum Minimisation acquisition functions provide any benefits compared to the more frequently-used Empirical Risk Minimisation versions?

2. Do compositional optimisers provide any advantages over non-compositional optimisers?

3. What are the practical savings for using memory-efficient implementations of compositional acquisition functions?

4. Are compositional methods more computationally expensive than non-compositional optimisation methods and how does runtime scale as a function of the input dimensionality?

5. How do compositional optimisers perform when optimising real-world black-box functions with noisy evaluations?

In order to answer Questions 1-4, we run twenty-eight optimiser variants on five synthetic, noiseless BBO problems for which the true maxima are known. Knowing the true maxima allows for exact computation of the normalised immediate regret

$$r_t = \frac{|f(\tilde{\mathbf{x}_\mathbf{t}}) - f(\mathbf{x}^*)|}{|f(\tilde{\mathbf{x}_\mathbf{0}}) - f(\mathbf{x}^*)|}, \tag{3.11}$$

with the function value at the global optimiser $\mathbf{x}^*$ denoted by $f(\mathbf{x}^*)$, $\tilde{\mathbf{x}_\mathbf{t}}$ is the algorithm's recommendation at round t and $f(\tilde{\mathbf{x}_\mathbf{0}})$ is the regret upon initialisation at round 0. The use of analytic functions also facilitates the treatment of input dimensionality as an experiment variable. In order to answer question 5, we focus on the tasks from Bayesmark. These tasks possess noise in the evaluations and are more representative of real-world BBO problems. For these latter experiments we take forward the best-performing optimisers observed in the synthetic function experiments. A pictorial summary of the experimental setup is provided in Figure 3.4.

Surrogate Model: For all tasks, we use a GP with constant mean function set to the empirical mean of the data, and a Matérn$(5/2)$ kernel with lengthscale parameter $\boldsymbol{\theta}$. At each acquisition step k, the hyperparameters of the GP kernel are estimated based on the current observed input-output pairs $\mathcal{D}_k$ by optimising the negative log marginal likelihood with a *Gamma* prior over $\boldsymbol{\theta}$. To facilitate the fitting procedure of the surrogate model, we standardise the outputs and apply an affine transformation to the inputs so that the search domain lies in $[0, 1]^d$. At the beginning of each experiment, three points are drawn uniformly at random within the search domain to initialise the surrogate model.

Additionally, in order to provide some indication as to how the GP-based surrogate model schemes, endowed with compositional optimisation of the acquisition function, perform against other surrogates, we also compare against the BOHB algorithm [66], a hybrid approach based on Bayesian optimisation and the Hyperband algorithm [145]. BOHB has recently been demonstrated to outperform Bayesian optimisation across a range of problems in the multi-fidelity setting, that is where multiple objective functions exist possessing varying degrees of accuracy and cost associated with querying them [223]. In order to enable comparison in the single-fidelity contexts considered in our experiments, we simply ignore the budget handling from Hyperband.

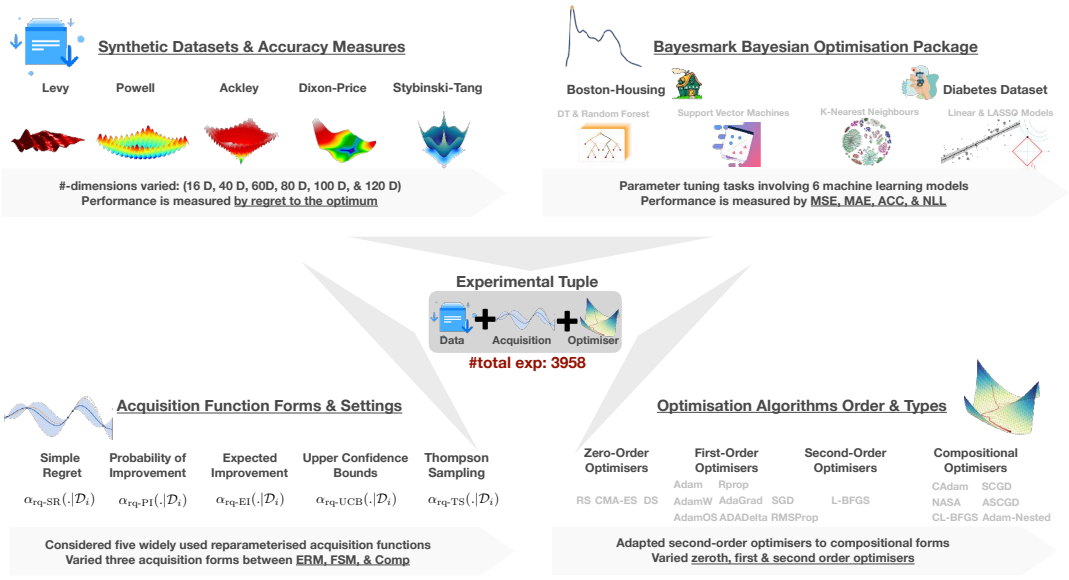


Figure 3.4.: Experiment Overview: **Top Left**: Synthetic functions (noiseless). **Top Right**: Bayesmark data (noisy). **Bottom Left**: Five classes of acquisition function in ERM, Finite-Sum, and Compositional forms. **Bottom Right**: Four classes of optimiser. Each experiment tuple comprises a dataset, an acquisition function and an optimiser. The study comprises 3958 experiments in total.

Acquisition Functions: We consider the batched versions of each acquisition function presented in Section 2.1.2, namely EI, PI, SR and UCB under ERM, FSM and compositional forms. Additionally, we employ Thompson sampling [232] as a baseline in order to provide an indication as to how the compositionally-optimised acquisition functions perform against another popular batch acquisition function.

Optimisers: Acquisition function maximisation is carried out using the zero-order optimisers RS, CMA-ES and DE from the PYMOO library [28], the non-compositional first-order optimisers Adadelta, Adagrad, Adam, AdamW, RMSprop, Rprop and SGA taken from PYTORCH [174], the second-order optimiser L-BFGS-B from the SCIPY library [243], as well as the compositional optimisers ASCGA, CAdam, Nested-MC, NASA and SCGA that we implemented on top of the BOTORCH library [17]. Except when using non-memory-efficient compositional methods, we used quasi-MC normal Sobol sequences [171] instead of *i.i.d.* normal samples in order to obtain lower variance estimates of the value and gradient

of the acquisition function as recommended by [17]. For the L-BFGS-B optimiser, the minibatch of samples was fixed in all cases. To ensure fairness in performance comparison, the same number of optimisation steps T (set to 64) and minibatch size m (set to 128), is used for each method at each acquisition step. As acquisition function maximisation is a non-convex problem, it is sensitive to the initialisation set. As such, we use multiple restart points [244] that we first obtain by drawing 1024 batches uniformly at random in the modified search space $[0, 1]^{q\times d}$, and second using the default heuristic from [17] to select only 32 promising initialisation batches. Consequently, at each inner optimisation step of BO, the Random Search optimisation strategy is granted $32 \times T \times m$ evaluations of the acquisition function at random batches. Similarly, CMA-ES and DE are run for 64 evolution and mutation steps, and the aforementioned initialisation strategy is used to generate the 32 members of the initial population.

It is known that first-order stochastic optimisers can be very sensitive to the choice of hyperparameter settings [17, 201]. Therefore, to limit the effect of choice of hyperparameter settings for the different optimisers, we conducted each experiment in two phases. An experiment in this instance is characterised by the 3-tuple consisting of a black-box function, an acquisition function and an optimiser.

In the first phase, we ran BO hyperparameter tuning to identify the best optimiser hyperparameters, in the sense that these hyperparameters provide the lowest final regret for the given task. This first phase allows us to compare optimisers in their most favourable settings, and therefore we hope that under-performance cannot be the result of a poor choice of hyperparameters but would reflect a real weakness of the considered method in tackling BO's inner optimisation problem.

In the second phase, we ran the black-box maximisation task using the acquisition function and optimiser with hyperparameters fixed to be the best ones identified during the first phase. The set and range of the considered hyperparameters for non-compositional optimisers and compositional optimisers are detailed in [91].

3.2.1. FSM vs. ERM

In the following experiment, we consider five non-separable, non-convex, synthetic black-box functions chosen to have a variety of optimisation landscapes and that are commonly-used benchmarks for optimisation algorithms [113, 141]. We include the unimodal functions *Dixon-Price* and *Powell* as well as the multimodal *Levy*, *Ackley* and *Styblinski-Tang* functions. We run experiments for (negative) versions of these functions with search domain specified as in [113, 141]. We consider optimisation problems across dimensionalities in the set (16D, 40D, 60D, 80D, 100D and 120D) in order to observe the impact of the input space dimension on the optimisers' performance. At each acquisition step, a batch

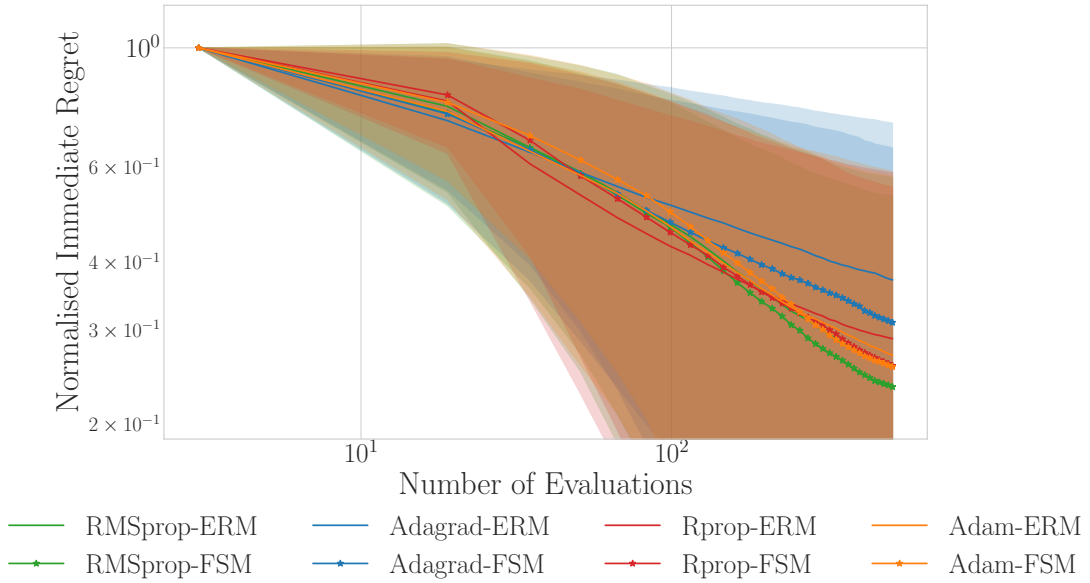


Figure 3.5.: Summary plot comparing the evolution of the normalised immediate regret averaged over all tasks when using first-order methods with either the ERM or FSM formulation of the acquisition function. The results of 960 experiments are summarised. We observe a small advantage of the FSM formulation over the ERM formulation across every optimiser. Statistical significance is discussed in [91].

of $q = 16$ points is acquired as a result of batch acquisition function maximisation. We run each BO algorithm with 32 acquisition steps and observe the normalised immediate regret from Equation 3.11 as the performance metric.

Results Summary: Figure 3.5 aggregates by optimiser category, (zero-order non-compositional, first-order compositional,...), the results of 960 experiments involving each combination of optimisation task, acquisition function and optimiser. The best performances obtained inside each category are accounted for. Specifically, given a category and an acquisition step, the lowest normalised immediate regrets obtained at this step by an optimiser belonging to this category are included and the average and standard deviation obtained over all optimisation tasks and all acquisition functions, are reported.

In light of these results we will now answer Question 1:

Question 1

Do Finite-Sum Minimisation acquisition functions provide any benefits compared to the more frequently-used Empirical Risk Minimisation versions?

When looking at the top four first-order non-compositional optimisers, Figure 3.5 shows in all cases that the FSM version outperforms the ERM version when averaging the normalised immediate regret scores over all optimisation tasks and acquisition functions. This can be seen in an un-aggregated breakdown in Figure 3.6. FSM outperforming ERM is an interesting discovery, and to the best of our knowledge, we are the first to observe this. We now proceed to our second question.

3.2.2. Compositional vs. Non-Compositional Optimisation

To synthesise the results obtained over all combinations of synthetic function (Levy, Ackley, Powell, Dixon-Price, Styblinski-Tang), input dimensionality (16D, 40D, 60D, 80D, 100D and 120D), and acquisition function (EI, PI, SR, UCB), we show in Figure 3.1 the evolution of the normalised immediate regret for each category of optimiser. We confirm the observation of [252] that gradient-based approaches outperform zero-order methods. Evolutionary strategies perform comparably to Random Search (which we exclude from its category as a global baseline). The poor performance of zero-order methods can be explained by the dimensionality of the acquisition function domain, ranging from 16×16 to 16×120 and the strict limitation on the number of optimisation steps. Results obtained with BOHB are also similar to Random Search, although it is worth mentioning that the experimental setting is single-fidelity and not multi-fidelity where BOHB has been observed to perform well. The performance of Thompson sampling (TS) coincides with the observation in the literature that TS has difficulty scaling beyond 8-10 dimensions [254]. We run GPflow [57] implementations of function-space, weight-space and decoupled TS with the default hyperparameters from [254]. We report these results in our summary plots and note that scaling such information-based acquisition functions constitutes an important direction for future work, see Section 3.3.

On examining gradient-based methods, we observe that quasi-Newton (C)L-BFGS-B is consistently outperformed by first-order methods, which was not observed in [17] where only a small-dimensional experiment with no batch acquisition (i.e. $q = 1$) was presented. From this global summary, our results favour first-order optimisers, with a relative advantage being given to compositional methods associated with the FSM approximation. On the other hand, non-compositional optimisers do not seem to be amenable to ERM or FSM formulation.

To show a breakdown of all experiments, we present in Figure 3.6 the best performances yielded by each category of optimiser for each input dimensionality and acquisition function considered. Each row corresponds to a domain dimension (16D, 40D, 60D, 80D, 100D and 120D) and each column is associated with an acquisition function (EI, PI, SR and UCB). Relative improvements yielded by BOHB and TS are also reported (there is no variation across columns as they do not depend on the acquisition function), leading the number of experiments aggregated on this figure to be 3100. On each row, the graph corresponding to the acquisition function that achieved the lowest regret for the given input dimension has a thick grey border. In 40, 60, 80 and 120 dimensions, the best performance is achieved using UCB with a first-order optimiser, while in 16 and 100 dims, it is SR with a first-order compositional optimiser that led to the largest relative improvement. From this figure, we can first observe that the dimensionality of the BO problem does not seem to have a significant impact on the relative performances between the different types of methods, that is, for any dimension, the best first-order gradient method outperforms the second-order methods, which achieve lower regret than zero-order ones. Aside from this trend at the level of the optimiser order, we do not notice any lower-level trend that may be driven by the input dimensionality.

Moreover, Figure 3.6 provides some insight into the comparatively better performance of first-order compositional optimisers observed in the global summary Figure 3.1. Lower regrets are obtained when the PI acquisition function is used. Nevertheless, the shading of the graphs corresponding to the best acquisition function for each dimensionality indicates that PI yields consistently higher regrets than UCB or SR, which encourages the use of these alternative acquisition functions in place of PI with a first-order compositional optimiser.

Returning to our second question:

> Question 2
>
> Do compositional optimisers provide any advantages over non-compositional optimisers?

The global summary Figure 3.1 in addition to Figure 3.6 indicate that there are a significant number of optimisation task and acquisition function pairs where a compositional optimiser is preferable and as such, compositional schemes warrant much more attention than they are currently receiving in the Bayesian optimisation community. We will now proceed to answer our third question.

3.2.3. Memory Efficiency

Compositional acquisition function maximisation requires considerably larger memory relative to ERM. However, by introducing a simple trick whereby we do not store all the auxiliary variables and adopt an alternative sampling scheme, we can dramatically reduce the memory requirements to be equivalent to those of ERM. In answer to question 3:

> Question 3
>
> What are the practical savings for using memory-efficient implementations of compositional acquisition functions?

Figure 3.7, which aggregates results obtained on tasks in 80, 100 and 120 dimensions using both memory-efficient and standard versions of CAdam, NASA and Nested-MC to maximise the acquisition function, shows that CAdam is negatively impacted by the ME implementation, whereas NASA and Nested-MC are positively impacted by memory efficiency. In all cases, the impact on going from standard to memory-efficient implementations is minor enough that we believe it warrants the use of the ME implementation as the de facto standard. We now proceed to answer Question 4:

3.2.4. Runtime Efficiency

Runtime efficiency is of great importance for many applications. As such, we wish to see how the execution time required for a single acquisition function optimisation varies across compositional optimisers and input dimensionality. We fix the acquisition function to UCB as this choice has negligible effect on overall timings and we run the BO algorithm for 32 acquisition steps on two black-box maximisation tasks using all available optimisers, repeating each experiment five times. In answer to Question 4:

> Question 4
>
> Are compositional methods more computationally expensive than non-compositional optimisation methods and how does runtime scale as a function of the input dimensionality?

There is a marked difference between the execution times reported in Figure 3.8a for compositional and non-compositional methods with compositional methods being slower relative to non-compositional. Additionally, ME methods are faster than standard compositional methods. We can also see that as the input dimensionality increases, a steeper incline in the execution time for compositional methods relative to non-compositional

methods may be observed; a feature to be expected given the extra backward passes required by compositional optimisers. Due to these additional backward passes, compositional methods are 1.5-2 times slower per iteration in terms of wall-clock time. It should be noted that compositional optimisers may require fewer iterations in total to converge to a specified accuracy and in this case overall wall-clock time could be comparatively better for them. Finally, if the black-box system evaluation wall-clock time is factors larger than the optimisation wall-clock time, which is the case in many real-world problems such as molecule synthesis where a single query can take 2-3 weeks [231], then the differences in runtime between compositional and non-compositional schemes becomes negligible. We now proceed to answer our final question.

3.2.5. Real-World Problems: Noisy Evaluations

We now examine the performance of optimisers on Bayesmark tasks. All tasks involve hyperparameter tuning for machine learning models. In contrast to the synthetic functions, the Bayesmark datasets possess noise in the evaluations of the black-box function, a feature inherent in the vast majority of real-world BBO problems. As such, these experiments are designed to assess whether the observations derived from the synthetic experiments are relevant for noisy problems.

Hyperparameter Tuning Tasks: The Bayesmark tasks consist of both regression and classification tasks on the Boston and Diabetes UCI datasets [61] respectively. In terms of hyperparameter tuning the following six models are considered: Decision Tree (DT), Random Forest (RF), K-Nearest Neighbours (kNN), Support Vector Machine (SVM), Linear and Lasso models. the dimensionality of each task varies from 2 to 9. In contrast to the synthetic functions, we only have access to noisy evaluation of the black-box functions in this instance. We apply Bayesian optimisation using 16 iterations of 8-batch acquisition steps, to optimise the validation loss, mean-squared error (MSE), mean absolute error (MAE), negative log likelihood (NLL) or accuracy depending on the task, plotting the normalised validation loss score (Eq 3.12) for performance comparison. We ran all six models on regression tasks (both MAE and MSE objectives) and we run three models (DT, RF and SVM) on classification tasks (both NLL and accuracy objectives) due to a limited computation budget. The score achieved after t acquisition steps is given by:

$$\textbf{score}_t = \frac{\mathcal{L}_t - \mathcal{L}^*}{\mathcal{L}_t^{\text{rand}} - \mathcal{L}^*} \tag{3.12}$$

where $\mathcal{L}_t$ is the best-achieved loss at batch t. $\mathcal{L}^*$ is the estimated optimal loss for the task and $\mathcal{L}^{\text{rand}}$ is the mean loss (across multiple runs) acquired from random search at batch t.

Experiment Setup: The top three non-compositional optimisers (Adam, RMSprop, Rprop) were selected for performance comparison against compositional optimisers (NASA, CAdam, Nested-MC). We show results for the four top-performing acquisition functions (SR, EI, PI and UCB) from the synthetic function experiments. We use the same GP surrogate model as in Sec 3.2.1, with rounding of integer values when either integer or categorical variables are present. Although more sophisticated methods exist to deal with categorical/integer variables [198, 56, 76] we do not consider them here as we are interested in solely in performance on acquisition function maximisation. We sample $2 \times D$ points uniformly at random to initialise the model. We run the same form of hyperparameter tuning for the initialisation as in the synthetic experiments, repeating each experiment 5 times in order to compute the variance for individual tasks.

Results Summary: In answer to our final question:

> Question 5
>
> How do compositional optimisers perform when optimising real-world black-box functions with noisy evaluations?

Figure 3.2 shows a high-level breakdown of compositional and non-compositional optimiser performance on the Bayesmark regression tasks. The best final scores for the model undergoing tuning are pooled across optimisers, tasks, loss functions and acquisition functions. We observe that compositional and non-compositional optimisers perform comparably, with compositional methods performing slightly better for DT, RF and SVM. We see that the mean scores are roughly equivalent for optimiser classes across the kNN, Lasso, linear and AdaBoost models. In an analogous fashion, Figure 3.3 pools the scores for all classification experiments. For the DT, and RF models, compositional methods achieve higher mean scores wheraeas comparable performance is observed when tuning the SVM model. In conclusion, compositional vs. non-compositional optimiser performance appears to vary depending on both the model class undergoing tuning as well as the performance metric.

Detailed Results: Figure 3.9 depicts a finer-grained breakdown of the pooled results for the Bayesmark regression tasks. Pooling in this case is carried out using the best, median and average optimiser performances across all intra-class optimisers and acquisition functions, where for example the best compositional optimiser for a given model would be the top-scoring optimiser-acquisition pair. For DT and RF, the best results are produced from compositional optimisers, whereas for SVM, AdaBoost, kNN and the linear model,

non-compositional methods exhibit better performance. For compositional optimisation of the Lasso model we observe better median performance for a higher number of black-box function evaluations, but deteriorating performance under the best grouping. Figure 3.10 similarly shows a finer-grained breakdown of the Bayesmark classification tasks. We observe that for certain models, such as RF, compositional methods perform better in each of best, median and average groupings at all steps in the optimisation, namely 8, 16 and 128 evaluations of the black-box system. In the DT experiments we again observe that compositional optimisers perform better in the latter optimisation steps (16 & 128 evaluations), but worse in the initial stages of the optimisation (8 evaluations). In summary, compositional methods yield better performance in two-thirds of the cases considered in Figure 3.10.

3.3. Future Work

In this paper, we presented an in-depth study of acquisition function maximisation in Bayesian optimisation. Apart from conventional forms typically used in literature, we demonstrated that acquisition functions adhere to a compositional structure enabling numerous new algorithms that led to favourable empirical results. We verified our claims in a rigorous experimental study involving 3958 tasks and twenty-eight optimisers. We used both synthetic and real-world data gathered from Bayesmark. We demonstrated that compositional optimisers outperform traditional solvers in 67 % of the time. In the future, we plan to extend our analysis to cover non-myopic acquisition functions, constrained and safe BO, high-dimensional BO [92] as well as to investigate compositional structures of causal BO.

			Dim.	16		40		60		80		100		120		**Tot.**	
				#Best	NFR	#Best	NFR	#Best	NFR	#Best	NFR	#Best	NFR	#Best	NFR	#Best	NFR
	Order	Optimiser	Ref.	(%)		(%)		(%)		(%)		(%)		(%)		(%)	
Non-Comp	0	RS	[91]	0	.33	0	.51	0	.60	0	.64	0	.68	0	.75	0	.59
		CMA-ES	[91]	0	.30	0	.49	0	.76	0	.80	0	.81	0	.85	0	.67
		DE	[91]	0	.29	0	.45	0	.61	0	.66	0	.66	0	.70	0	.56
		Subtot.		0	.31	0	.48	0	.66	0	.70	0	.72	0	.77	0	.61
	1	SGA	Details [91]	0	.18	0	.28	0	.33	0	.42	0	.35	0	.48	0	.34
		Adagrad	Details [91]	5	.36	5	.55	5	.66	5	.75	5	.87	10	.89	6	.68
		RMSprop	Details [91]	10	.29	5	.45	15	.47	0	.58	0	.53	**15**	.64	8	.49
		Adam	Details [91]	5	.35	15	.46	5	.51	5	.53	**20**	.61	10	.70	10	.52
		Adadelta	Details [91]	0	.20	0	.44	5	.32	0	.46	0	.45	0	.48	1	.39
		Rprop	Details [91]	0	.36	0	.49	10	.57	5	.61	0	.59	10	.66	4	.55
		AdamW	Details [91]	0	.18	0	.24	5	.22	5	.22	5	.25	5	.23	3	.22
		Adamos	Details [91]	0	.17	0	.26	0	.26	0	.28	5	.30	5	.34	2	.27
		Subtot.		20	.26	25	.40	45	.42	20	.48	35	.49	**55**	.55	33	.43
	2	L-BFGS-B	Details [91]	0	.19	0	.29	0	.39	0	.45	0	.45	0	.51	0	.38
		Subtot.		0	.19	0	.29	0	.39	0	.45	0	.45	0	.51	0	.38
	Tot.			20	.27	25	.41	45	.48	20	.53	35	.55	55	.60	33	.47
Comp	0	CMA-ES	Details [91]	0	.30	0	.49	0	.76	0	.82	0	.83	0	.87	0	.68
		DE	Details [91]	0	.30	0	.46	0	.61	0	.64	0	.67	0	.71	0	.57
		Subtot.		0	.30	0	.47	0	.69	0	.73	0	.75	0	.79	0	.62
	1	SCGA	Details [91]	10	.12	0	.18	0	.33	0	.44	0	.52	0	.62	2	.37
		ASCGA	Details [91]	5	.11	5	.17	0	.34	0	.48	0	.53	0	.60	2	.37
		CAdam	Details [91]	20	.09	25	.12	**35**	.19	**25**	.14	**20**	.14	10	.22	**22**	.15
		NASA	Details [91]	**45**	.08	**35**	.21	15	.31	20	.39	10	.40	5	.55	**22**	.32
		Nested-MC	Details [91]	0	.17	10	.22	5	.23	5	.26	5	.29	0	.38	4	.26
		CAdam-ME	Details [91]	-	-	-	-	-	-	20	.14	15	.16	20	.24	18	.18
		NASA-ME	Details [91]	-	-	-	-	-	-	10	.35	10	.40	5	.52	8	.43
		Nested-MC-ME	Details [91]	-	-	-	-	-	-	0	.28	5	.29	5	.32	3	.29
		Subtot.		**80**	.12	**75**	.18	**55**	.28	**80**	.31	**65**	.34	45	.43	**67**	.28
	2	CL-BFGS-B	Details [91]	0	.20	0	.28	0	.34	0	.36	0	.44	0	.50	0	.35
		Subtot.		0	.20	0	.28	0	.34	0	.36	0	.44	0	.50	0	.35
	Tot.			**80**	.17	**75**	.27	**55**	.39	**80**	.39	**65**	.43	45	.50	**67**	.36

Table 3.1.: Marginal results over acquisition functions and synthetic black-box optimisation tasks (i.e., 20 tasks per dimension).

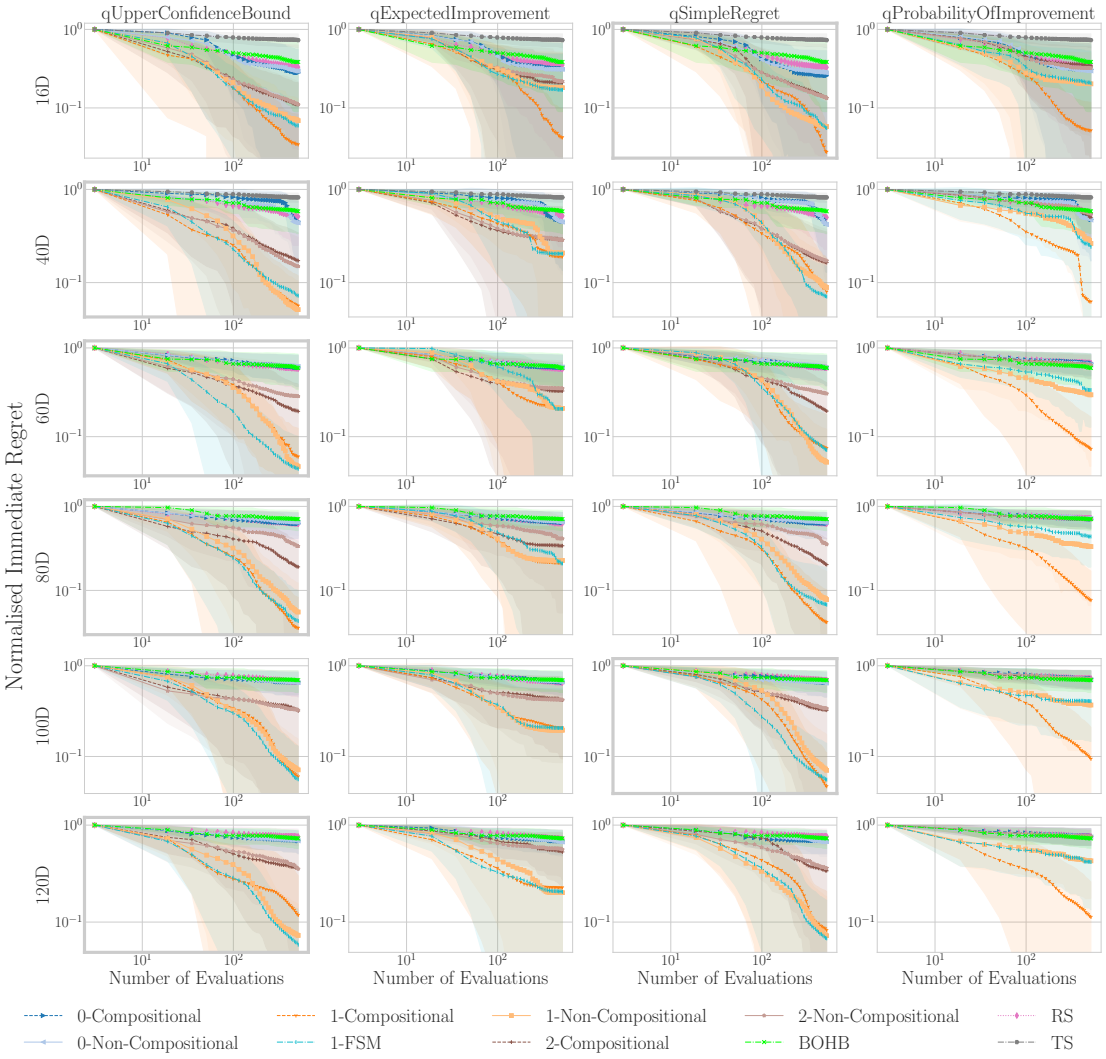


Figure 3.6.: Each row corresponds to a domain dimension (16D, 40D, 60D, 80D, 100D and 120D) and each column is associated with an acquisition function (EI, PI, SR and UCB). On each row, the graph corresponding to the acquisition function that achieved the lowest regret for the given input dimension has a thick grey border. In 40, 60, 80 and 120 dimensions, the best performance is achieved using UCB with a first-order optimiser, while in 16 and 100 dims, it is SR with a first-order compositional optimiser that led to the largest relative improvement.

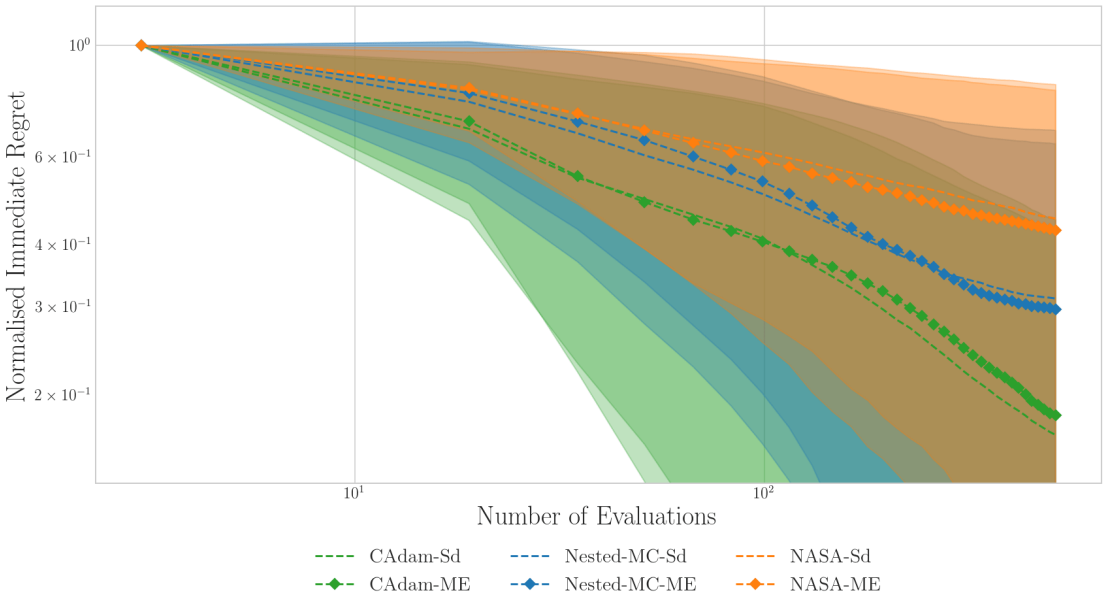


Figure 3.7.: (3.7) Summary plot comparing the evolution of the normalised immediate regret averaged over all considered acquisition functions, and optimisation tasks in 80, 100 and 120 dimensions, when using standard (Sd) and memory-efficient (ME) compositional first-order optimisers. From this figure, aggregating the results of 360 experiments, we can see that memory-efficient optimiser versions perform comparably to standard optimisers, thus making it worthwhile to use memory-efficient implementations due to the large memory savings.

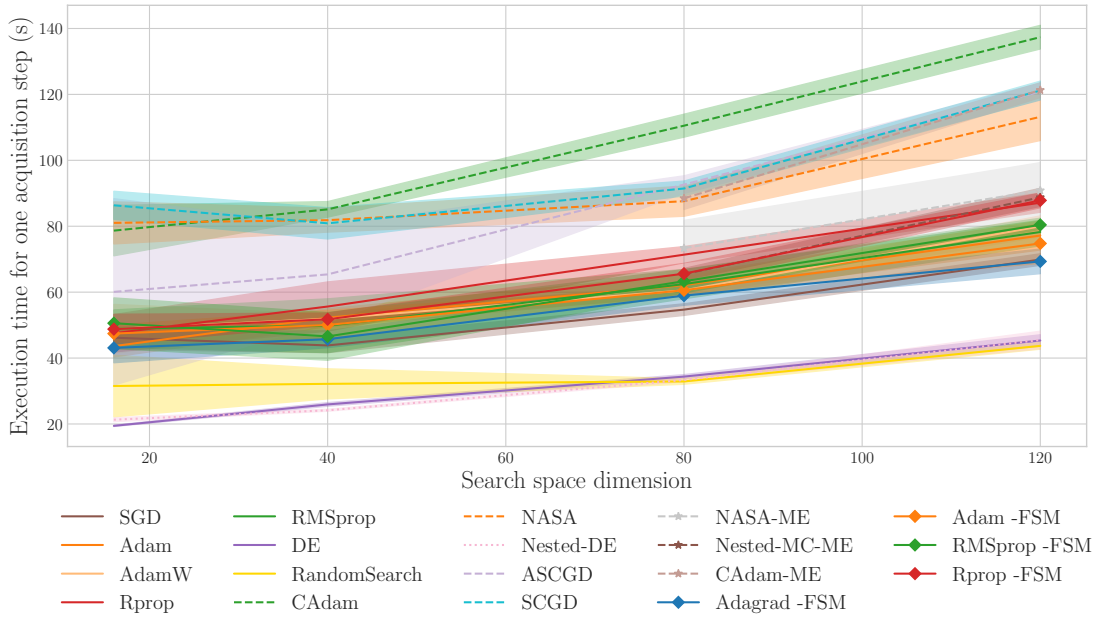


Figure 3.8.: (3.8a) Execution time of UCB maximisation run on 4 CPUs. We report the time it takes an optimiser to carry out a single UCB maximisation, and we show the mean and standard deviation observed over 5 seeds, 32 acquisition steps and 2 synthetic black-box functions in 16, 40, 80 and 120 dims. From this figure, aggregating results of 152 experiments, we observe that compositional methods take about 1.5-2x the CPU time taken by non-compositional methods. We do not report the execution times measured for (C)L-BFGS-B and CMA-ES as they are an order of magnitude greater than those observed for non-compositional, first-order methods. We provide complementary results in [91].

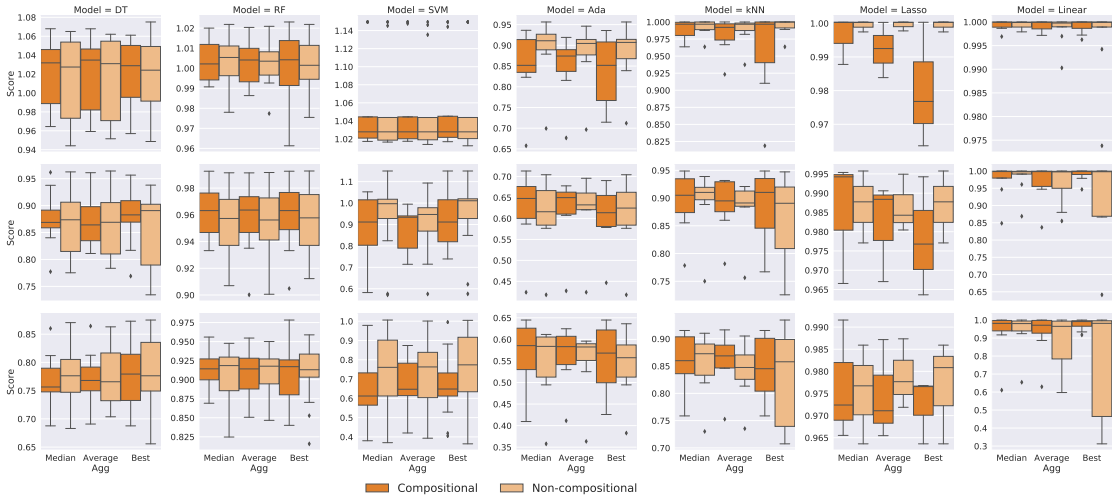


Figure 3.9.: The boxplot shows the quartiles of compositional and non-compositional optimiser performance on the regression hyperparameter tuning task, where the performance metrics are MAE and MSE. For each model, we show a further split of the optimiser class for different aggregation methods. This plot summarises all 672 experiments conducted on regression tasks on the Bayesmark dataset. We observe performance benefits for DT, RF and AdaBoost when using a compositional optimiser, with SVM and kNN showing performance benefits when using a non-compositional optimiser. When to use compositional and when to use non-compositional?

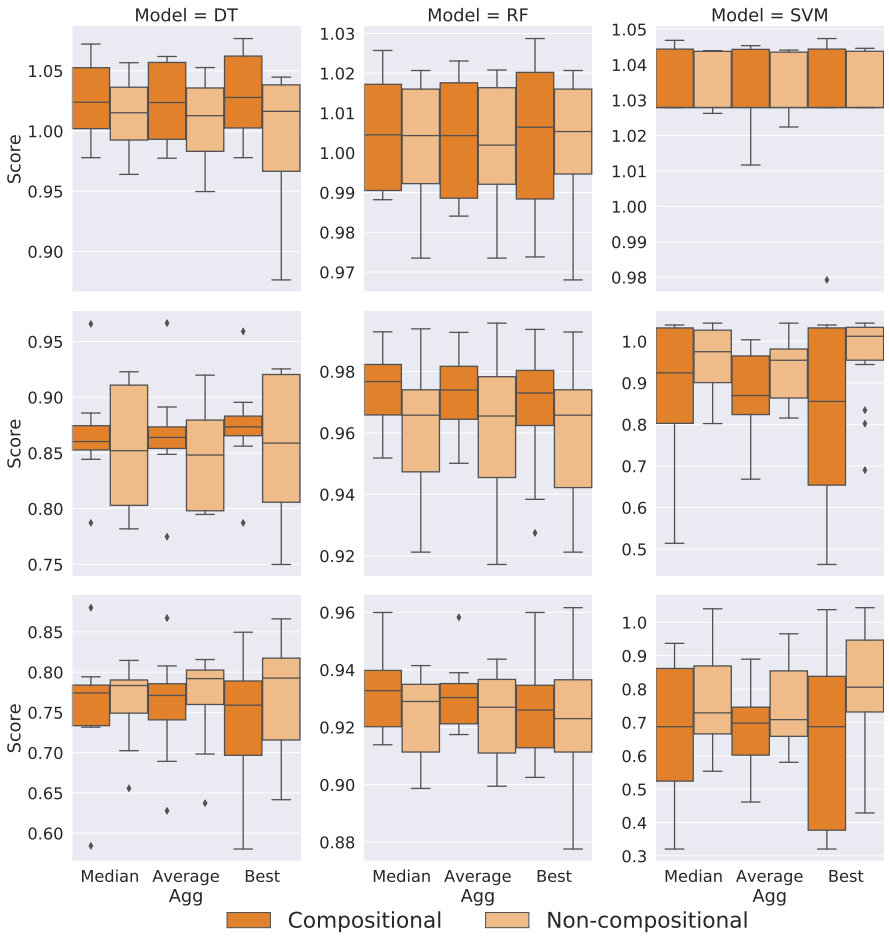


Figure 3.10.: The boxplot shows the quartiles of compositional and non-compositional optimiser performance on the classification hyperparameter tuning task, where the performance metrics are NLL and accuracy. For each model, we show a further split of the optimiser class against different aggregation methods. This plot summarises all 288 experiments conducted on classification tasks for the Bayesmark datasets. We observe that for the DT and RF models, compositional optimisers offer modest performance gains relative to non-compositional optimisers, yet non-compositional optimisers perform better on SVM hyperparameter tuning.

4. HEBO: Pushing the Limit of Sample-Efficient Hyper-parameter Optimisation

In this chapter we will conduct an empirical study of assumptions in Bayesian Optimisation applied to hyperparameter tuning tasks, and through this we will uncover key algorithmic components that lead to a SOTA algorithm, named Heteroscedastic Evolutionary Bayesian Optimisation [52] (HEBO). Firstly, in Section 4.1 we will introduce the motivation for black-box optimisation for hyperparameter tuning. In Section 4.3 we introduce common design choices and assumptions in black-box optimisation and in Section 4.4 we detail the research questions we wish to answer around these common design choices and assumptions. We introduce improvements to alleviate the limitations of existing assumptions in Section 4.5 and discuss their impact in the experimental Section 4.6. We discuss related work in Section 4.2 and draw conclusions on the improvements in Section 4.8.

4.1. Introduction

Although achieving significant success across numerous applications [29, 148, 68, 120, 53], the performance of machine learning models chiefly depends on the correct setting of hyperparameters. As models grow larger and more complex, efficient and autonomous hyperparameter tuning algorithms become crucial determinants of performance. A variety of methods from black-box and multi-fidelity optimisation [119, 208] have been adopted for hyperparameter tuning with varying degrees of success. Techniques such as Bayesian optimisation (BO), for example, enable sample efficiency (in terms of black-box evaluations) at the expense of high computational demands, while "unguided" bandit-based approaches can fail to converge [67]. Identifying such failure modes, the authors in [67] built on [146] and proposed a combination of bandits and BO that achieves the best of both worlds; fast convergence and computational scalability. More recently in the context of the 2020 NeurIPS competition on Black-Box Optimisation, many BO variants have been convincingly demonstrated to be superior to random search

for the task of hyperparameter tuning [237]. Though impressive, such successes of BO and alternative black-box optimisers, belie a set of restrictive modelling and acquisition function assumptions. We begin by describing these assumptions.

Modelling Assumptions: A core determinant of BO performance is the set of data modelling assumptions required to specify an appropriate probabilistic model of the black-box objective (e.g., the choice of validation loss in hyperparameter tuning tasks). The model should not only provide accurate point estimates, but should also maintain calibrated uncertainty estimates to guide exploration of the objective. Amongst many possible surrogates [225, 109], Gaussian processes [250] (GPs) are the default choice due to their flexibility and sample efficiency. Growing interest in applications of Bayesian optimisation has catalysed engineering feats that enhance scalability and training efficiency of GP surrogates by exploiting graphical processing units [131, 18].

Similar to any other framework, the correct specification of a GP model is dictated by the data modelling assumptions imposed by the user. For instance, a homoscedastic GP suffers from misspecification when required to model data with heteroscedastic noise whilst stationary GPs fail to track non-stationary targets. The aforementioned shortcomings are not unnatural across a range of real-world problems [123, 89] and hyperparameter tuning of machine learning algorithms is no exception, as illustrated in our hypothesis tests of Section 4.4.2. Hence, even if one succeeds in improving computational efficiency, frequently-made assumptions such as homoscedasticity and stationarity can easily inhibit the performance of any BO-based hyperparameter tuning algorithm. Despite the importance of these assumptions in practice, GPs that presume homoscedasticity and stationarity still constitute the most common choice of surrogate.

Acquisition Function & Optimiser Assumptions: Modelling choices such as those described above are not unique to the GP fitting procedure but rather transcend to other steps in the BO algorithm. Precisely, given a model that adheres to some (or all) assumptions mentioned above, the second step involves maximising an acquisition function to query novel input locations that are then evaluated. Hence, practitioners introduce additional constraints relating to the category of optimisation variables and the choice of acquisition function. When it comes to variable categories, mainstream implementations [131, 18] assume continuous domains and employ first and second-order optimisers such as LBFGS [149] and ADAM [126] to propose query locations. Real-valued configurations cover but a subset of possible machine learning hyperparameters rendering discrete variable categories out of scope, an example being the hidden layer size in deep networks. Moreover, from the point of view of acquisition functions, libraries tend to presuppose that one unique acquisition performs best in a given task, while research

has shown that benefits that can arise from a combined solution [211, 212, 152] as we demonstrate in Section 4.6.

Contributions: Having identified important modelling choices in BO, our goal in this paper is to provide empirical insight into the impact of modelling choice on empirical performance. As a case study, we consider best practices for hyperparameter tuning. We wish for our findings to be applicable across a broad range of tasks and datasets, be attentive to the effect of random initialisation on algorithmic performance, and naturally, be reproducible. As such, we prefer to build on established benchmark packages, especially those that facilitate fast and scalable evaluations with multi-seeding protocols. To that end, we undertake our evaluation in 2140 experiments from 108 real-world problems from the UCI repository [61], which was also the testbed of choice for the NeurIPS 2020 Black-Box Optimisation challenge [237]. Our findings point towards the following conclusions:

1. hyperparameter tuning tasks exhibit significant levels of heteroscedasticity and non-stationarity.

2. Input and output warping mitigate the effects of heteroscedasticity and non-stationarity giving rise to better performing tuning algorithms with higher mean and median performance across all 108 black-box functions under examination.

3. Individual acquisition functions tend to conflict in their solution (i.e., an optimum for one acquisition function can be a sub-optimal point for another and vice versa). Using a multi-objective formulation significantly improves performance;.

To verify our principal conclusions, we conduct additional ablation studies on our proposed solution method, Heteroscedastic and Evolutionary Bayesian Optimisation (HEBO) which attempts to address the shortcomings identified in our analysis and placed first in the 2020 NeurIPS Black-Box Optimisation Challenge. We obtain a ranked order of importance for significant components of `HEBO`, finding that output warping, multi-objective acquisitions and input warping lead to the most significant improvements followed by robust acquisition function formulations.

4.2. Related Work

We introduce work on the following topics relating to modelling, acquisition and optimisers in Bayesian optimisation:

Heteroscedasticity with output transforms: Among various approaches to handling heteroscedasticity [123, 143, 138, 39, 89], transforming the output variables is a straightforward option giving rise to warped Gaussian processes [217]. More recently, output transformations have been extended to compositions of elementary functions [195] and normalising flows [191, 156]. Output transformations have not featured prominently in the Bayesian optimisation literature, perhaps due to the commonly-held opinion that warped GPs require more data relative to standard GPs in order to function as effective surrogates [167]. Rather than introduce additional hyperparameters to the GP, we enable efficient output warping through methods that only require pre-training. Recent work [64] has also investigated Gaussian copula transforms which may prove to be particularly effective in situations where there are outliers.

Non-stationarity with input warping: Many surrogate models with input warping exist for optimising non-stationary black-box objectives [220, 40, 168] and have enjoyed particular success in hyperparameter tuning where the natural scale of parameters is often logarithmic. Traditionally, a Beta cumulative distribution function is used. In this paper, we adopt the `Kumaraswamy` warping which is another instance of the generalised Beta class of distributions which we have observed to achieve superior performance [220] [1]; confirming results reported in [18].

Multi-objective acquisition ensembles: Multi-objective acquisition ensembles were first proposed in [152] and are closely related to portfolios of acquisition functions [107, 211, 18]. In this form, the optimisation problem involves at least two conflicting and expensive black-box objectives and as such, solutions are located along the Pareto-efficient frontier. The multi-objective acquisition ensemble employs these ideas to find a Pareto-efficient solution amongst multiple acquisition functions. Although we utilised the multi-objective acquisition ensemble, we note that our framework is solver agnostic in so far as any multi-objective optimiser [1] may be applied.

Robustness of Acquisitions: Methods achieving robustness with respect to either surrogates [173] or the optimisation process [30, 27] have been previously proposed. Most relevant to our setting, is the approach of [30] that introduces robustness to BO by solving a $\max\min$ objective to determine optimal input perturbations. Their method, however, relies on gradient ascent-descent-type algorithms that require real-valued variables and are not guaranteed to converge in the general non-convex, non-concave setting [147]. On the other hand, our solution possesses two advantages: 1) simplicity of implementation as

[1] For clarity we note that the input warping function used in [220] is the same one used in this work.

we merely require random perturbations of acquisition functions to guarantee robustness, and 2) support for mixed variable solutions through the use of evolutionary solvers.

4.3. Standard Design Choices in BO

As discussed earlier, the problem of hyperparameter tuning can be framed as an instance of black-box optimisation

$$\arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \tag{4.1}$$

with $\boldsymbol{x}$ denoting a configuration choice, $\mathcal{X}$ a (potentially) mixed design space, and $f(\boldsymbol{x})$ a validation accuracy we wish to maximise. In this paper, we focus on BO as a solution concept for black-box problems of the form depicted in Equation (4.1). BO considers a sequential decision approach to the global optimisation of a black-box function $f : \mathcal{X} \to \mathbb{R}$ over a bounded input domain $\mathcal{X}$. At each decision round, i, the algorithm selects a collection of q inputs $\boldsymbol{x}_{1:q}^{(\text{new})} \in \mathcal{X}^q$ and observes values of the *black-box* function $\boldsymbol{y}_{1:q}^{(\text{new})} = f(\boldsymbol{x}_{1:q}^{(\text{new})})$. The goal is to rapidly approach the maximum $\boldsymbol{x}^\star = \arg\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. Since both $f(\cdot)$ and $\boldsymbol{x}^\star$ are unknown, solvers need to trade off exploitation and exploration during this search process.

To achieve this goal, BO algorithms operate in two steps. In the first, a Bayesian model is learned, while in the second an acquisition function determining new query locations is maximised. Next, we survey frequently-made assumptions in mainstream BO implementations and contemplate their implications for performance.

4.3.1. Modelling Assumptions

When black-boxes are real-valued, Gaussian processes [188] are effective surrogates due to their flexibility and ability to maintain calibrated uncertainty estimates. In established implementations of BO, designers place GP priors on latent functions, $f(\cdot)$, which are fully specified through a mean function, $m(\boldsymbol{x})$, and a covariance function or kernel $k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}')$ with $\boldsymbol{\theta}$ representing kernel hyperparameters. The model specification is completed by defining a likelihood. Here, practitioners typically assume that observations y_l adhere to a Gaussian noise model such that $y_l = f(\boldsymbol{x}_l) + \epsilon_l$ where $\epsilon_l \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. This assumption generates a Gaussian likelihood of the form $y_l|\boldsymbol{x}_l \sim \mathcal{N}(f_l, \sigma^2_{\text{noise}})$ where we use f_l to denote $f(\boldsymbol{x}_l)$ with $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}'))$. Additionally, a further design choice commonly made by practitioners is that the GP kernel is stationary, depending only on the norm between $\boldsymbol{x}$ and $\boldsymbol{x}'$, $||\boldsymbol{x} - \boldsymbol{x}'||$. From this exposition, we conclude two important modelling assumptions stated as *data stationarity* and *homoscedasticity of the noise distribution*. If

the true latent process does not adhere to these assumptions, the resultant model will be a poor approximation to the black-box. Realising the potential empirical implications of these modelling choices, we identify the first two questions addressed by this paper

Q.I. Are hyperparameter tuning tasks stationary?

Q.II. Are hyperparameter tuning tasks homoscedastic?

Q.III. Can acquisition function solutions conflict in hyperparameter tuning tasks?

In Section 4.4.2, we show that even amongst the simplest hyperparameter tuning tasks, the null hypothesis may be rejected in the case of statistical hypothesis tests for heteroscedasticity and non-stationarity.

4.3.2. Acquisition Function & Optimisation Assumptions

Acquisition functions trade off exploration and exploitation by utilising statistics from the posterior $p_{\boldsymbol{\theta}}(f(\cdot)|\mathcal{D})$ with $\mathcal{D}$ denoting the data (hyperparameter configurations as inputs and validation accuracy as outputs) collected so far. Under a GP surrogate with Gaussian-corrupted observations $y_\ell = f(\boldsymbol{x}_\ell) + \epsilon_\ell$ where $\epsilon_\ell \sim \mathcal{N}(0, \sigma^2)$, and given a data set $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\}$, the joint distribution of $\mathcal{D}$ and an arbitrary set of input points $\boldsymbol{x}_{1:q}$ is given by

$$\left[\begin{array}{c} \boldsymbol{y} \\ f(\boldsymbol{x}_{1:q}) \end{array} \right] \Bigg| \, \boldsymbol{\theta} \sim \mathcal{N}\left(\left[\begin{array}{c} m(\boldsymbol{x}) \\ m(\boldsymbol{x}_{1:q}) \end{array} \right], \left[\begin{array}{cc} \boldsymbol{K}_{\boldsymbol{\theta}} + \sigma^2 \boldsymbol{I} & \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) \\ \boldsymbol{k}_{\boldsymbol{\theta}}^\intercal(\boldsymbol{x}_{1:q}) & \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}, \boldsymbol{x}_{1:q}) \end{array} \right] \right),$$

where $\boldsymbol{K}_{\boldsymbol{\theta}} = \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x})$ and $\boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) = \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{x}_{1:q})$. From this joint distribution one can derive though marginalisation [188] the posterior predictive $p(f(\boldsymbol{x}_{1:q})|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}))$ with

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) &= m(\boldsymbol{x}_{1:q}) + \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q})^\top (\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - m(\boldsymbol{x})) \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}) &= \boldsymbol{K}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}, \boldsymbol{x}_{1:q}) - \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q})^\top (\boldsymbol{K}_{\boldsymbol{\theta}} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}). \end{aligned}$$

As such we note that $p(f(\boldsymbol{x}_{1:q})|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:q}))$. In this paper, we focus on three widely-used myopic acquisition functions which in a reparameterised form can be written as [251]

Expected Improvement (EI):

$$\alpha^{\boldsymbol{\theta}}_{\text{EI}}(\boldsymbol{x}_{1:q}|\mathcal{D}) = \mathbb{E}_{\text{post.}}\left[\max_{j\in 1:q}\{\text{ReLU}(f(\boldsymbol{x}_j) - f(\boldsymbol{x}^+))\}\right],$$

where the subscript 'post.' is the predictive posterior of a GP [188], $\boldsymbol{x}_j$ is the j^{th} vector of $\boldsymbol{x}_{1:q}$, and $\boldsymbol{x}^+$ is the best performing input in the data so far.

Probability of Improvement (PI):

$$\alpha^{\boldsymbol{\theta}}_{\text{PI}}(\boldsymbol{x}_{1:q}|\mathcal{D}) = \mathbb{E}_{\text{post.}}\left[\max_{j\in 1:q}\{\mathbb{1}\{f(\boldsymbol{x}_j) - f(\boldsymbol{x}^+)\}\}\right],$$

where $\mathbb{1}\{\cdot\}$ is the left-continuous Heaviside step function.

Upper Confidence Bound (UCB):

$$\alpha^{\boldsymbol{\theta}}_{\text{UCB}}(\boldsymbol{x}_j) = \mathbb{E}_{\text{post.}}\left[\max_{j\in 1:q}\left\{\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_j) + \sqrt{\beta\pi/2}|\gamma_{\boldsymbol{\theta}}(\boldsymbol{x}_j)|\right\}\right],$$

where $\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_j)$ is the posterior mean of the predictive distribution and $\gamma_{\boldsymbol{\theta}}(\boldsymbol{x}_j) = f(\boldsymbol{x}_j) - \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_j)$. When it comes to practicality, generic BO implementations make additional assumptions during the acquisition maximisation step. First, it is assumed that one of the aforementioned acquisitions works best for a specific task, and that the GP model is an accurate approximation to the black-box. However, when it comes to real-world applications, both of these assumptions are difficult to validate; the best-performing acquisition is challenging to identify upfront and GP models may easily be misspecified. With this in mind, we identify a third question that we wish to address.

In the following section, we affirm that acquisitions can conflict even on the simplest of hyperparameter tuning tasks. Moreover, we show that a robust formulation to tackle misspecification of acquisition maximisation can improve overall performance (see Section 4.5.2).

4.4. Modelling Assumption Analysis

Before discussing the improvements afforded to BO via our solution method, we detail analyses conducted to answer questions (**Q.I.**, **Q.II.**, and **Q.III.**) posed in the previous

section. Our analyses indicate

A.I.: Even simple hyperparameter tuning tasks exhibit significant heteroscedasticity.
A.II.: Even simple hyperparameter tasks exhibit significant non-stationarity.
A.III.: Acquisition functions conflict in their optima, occasionally leading to opposing solutions.

Experiment Setting: We create a wide range of hyperparameter tasks (108) across a variety of classification and regression problems. We use nine models, (e.g. multilayer perceptrons, support vector machines) and six datasets (two regression and four classification) from the UCI repository, and two metrics per dataset (such as negative log-likelihood or mean squared error). Each model possesses tuneable hyperparameters, e.g. the number of hidden units of a neural network. The goal is to fit these hyperparameters so as to maximise/minimise one of the specified metrics. Values of the black-box objective are stochastic with noise contributions originating from the train-test splits used to compute the losses. Experimentation was facilitated by the `Bayesmark`[2] package. Full hyperparameter search spaces are defined in Table 4.1 and Table 4.2 of the Appendix [3].

Statistical Hypothesis Testing for Heteroscedasticity and Non-Stationarity: We describe here the statistical hypothesis tests we use to answer **Q.II.**. GP regression typically considers a conditional normal distribution of the observations $y|\cdot \sim \mathcal{N}(f(\cdot), \sigma^2(\cdot))$ and in most cases $\sigma(\cdot)^2$ is assumed to be constant, in which case the GP is termed homoscedastic. To assess whether the homoscedasticity assumption holds for the tasks under examination, we make use of Levene's test and the Fligner-Killeen test. To run these tests on a given task, we evaluate $k = 50$ distinct sets of hyperparameters $\{x_i\}_{1\leq i\leq k}$ for $n = 10$ times and obtain scores $\{Y_{ij}\}_{1\leq i\leq k, 1\leq j\leq n}$, where Y_{ij} is the j^{th} score observed when evaluating the i^{th} configuration. For $i = 1, \ldots, k$, let σ_i^2 denote the observed variance of $y|x_i$, then both Levene's test and the Fligner-Killeen test share the same null hypothesis of homoscedasticity

$$H_0 : \sigma_1^2 = \cdots = \sigma_k^2.$$

In all 108 tests, we see a p-value significantly lower than 0.05 in 72 tasks using Levene's test, and in 73 tasks using Fligner-Killeen test. Such results (shown in detail in [52]) imply that at least 66% of the experimental tasks exhibit heteroscedastic behaviour.

[2] https://github.com/uber/bayesmark
[3] It is these search spaces that are used by the random search baseline.

Table 4.1.: Search spaces for hyperparameter tuning on classification tasks. We specify the variable type of each hyperparameter (with $\mathbb{R}$ for real-valued and $\mathbb{Z}$ for integer- valued) as well as the search domain. We specify $\log -\mathcal{U}$ (resp. logit $-\mathcal{U}$) to indicate that a $\log$ (resp. logit) transformation is applied to the optimisation domain.

Model	Parameter	Type	Domain
kNN	n_neighbors	$\mathbb{Z}$	$\mathcal{U}(1, 25)$
	p	$\mathbb{Z}$	$\mathcal{U}(1, 4)$
Support Vector Machine	C	$\mathbb{R}$	$\log -\mathcal{U}(1, 10^3)$
	gamma	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{-3})$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
Decision Tree	max_depth	$\mathbb{Z}$	$\mathcal{U}(1, 15)$
	min_samples_split	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_samples_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	min_weight_fraction_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	max_features	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_impurity_decrease	$\mathbb{R}$	$\mathcal{U}(0, 0.5)$
Random Forest	max_depth	$\mathbb{Z}$	$\mathcal{U}(1, 15)$
	max_features	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_samples_split	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.99)$
	min_samples_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	min_weight_fraction_leaf	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.01, 0.49)$
	min_impurity_decrease	$\mathbb{R}$	$\mathcal{U}(0, 0.5)$
MLP-Adam	hidden_layer_sizes	$\mathbb{Z}$	$\mathcal{U}(50, 200)$
	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{1})$
	batch_size	$\mathbb{Z}$	$\mathcal{U}(10, 250)$
	learning_rate_init	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	validation_fraction	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.1, 0.9)$
	beta_1	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.5, 0.99)$
	beta_2	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.9, 1 - 10^{-6})$
	epsilon	$\mathbb{R}$	$\log -\mathcal{U}(10^{-9}, 10^{-6})$
MLP-SGD	hidden_layer_sizes	$\mathbb{Z}$	$\mathcal{U}(50, 200)$
	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{1})$
	batch_size	$\mathbb{Z}$	$\mathcal{U}(10, 250)$
	learning_rate_init	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	power_t	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.1, 0.9)$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	momentum	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.001, 0.999)$
	validation_fraction	$\mathbb{R}$	$\text{logit} - \mathcal{U}(0.1, 0.9)$
AdaBoost	n_estimators	$\mathbb{Z}$	$\mathcal{U}(10, 100)$
	learning_rate	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{1})$
Lasso	C	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	intercept_scaling	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
Linear	C	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	intercept_scaling	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$

Table 4.2.: Models and search spaces for hyperparameter tuning on regression tasks. Models having the same search spaces for classification and regression tasks are omitted (cf. Table 4.1).

Model	Parameter	Type	Domain
AdaBoost	n_estimators	$\mathbb{Z}$	$\mathcal{U}(10, 100)$
	learning_rate	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{1})$
Lasso	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	fit_intercept	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	normalize	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	max_iter	$\mathbb{Z}$	$\log -\mathcal{U}(10, 5000)$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-5}, 10^{-1})$
	positive	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
Linear	alpha	$\mathbb{R}$	$\log -\mathcal{U}(10^{-2}, 10^{2})$
	fit_intercept	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	normalize	$\mathbb{Z}$	$\mathcal{U}(0, 1)$
	max_iter	$\mathbb{Z}$	$\log -\mathcal{U}(10, 5000)$
	tol	$\mathbb{R}$	$\log -\mathcal{U}(10^{-4}, 10^{-1})$

Levene's Test Levene's test statistic is defined as

$$W = \frac{N-k}{k-1} \cdot \frac{\sum_{i=1}^{k} n(\bar{Z}_{i\cdot} - \bar{Z}_{\cdot\cdot})^2}{\sum_{i=1}^{k}\sum_{j=1}^{n}(Z_{ij} - \bar{Z}_{i\cdot})^2},$$

where $N = k \times n$, $Z_{ij} = |Y_{ij} - \frac{1}{n}\sum_{j=1}^{n} Y_{ij}|$, $\bar{Z}_{i\cdot} = \frac{1}{n}\sum_{j=1}^{n} Z_{ij}$ and $\bar{Z}_{\cdot\cdot} = \frac{1}{k}\sum_{i=1}^{k} \bar{Z}_{i\cdot}$, for all $i = 1, \ldots, k$, $j = 1, \ldots, n$. Levene's test rejects the null hypothesis of homoscedasticity H_0 if

$$W > F_{\alpha, k-1, N-k},$$

where $F_{\alpha,k-1,N-k}$ is the upper critical value at a significance level α of the F distribution with $k-1$ and $N-k$ degrees of freedom. The Fligner-Killeen test is an alternative to Levene's test that is particularly robust to outliers.

Fligner-Killeen Test: Computation of the Fligner-Killeen test involves ranking all the absolute values $\{|Y_{ij} - \tilde{Y}_i|\}_{1\le i\le k, 1\le j\le n}$, where $\tilde{Y}_i$ is the median of $\{Y_{ij}\}_{1\le j\le n}$. Increasing scores $a_{N,r} = \Phi^{-1}\left(\frac{1+\frac{r}{N+1}}{2}\right)$ are associated with each rank $r = 1, \ldots, N$, where $N = kn$ and $\Phi(\cdot)$ is the cumulative distribution function for a standard normal random variable.

We denote the rank score associated with Y_{ij} as r_{ij}. The Fligner-Killeen test statistic is given by

$$\chi_o^2 = \frac{\sum_{i=1}^{k} n\left(\bar{A}_i - \bar{a}\right)^2}{V^2},$$

where $\bar{A}_i = \frac{1}{n}\sum_{j=1}^{n} a_{N,r_ij}$, $\bar{a} = \frac{1}{N}\sum_{r=1}^{N} a_{N,r}$ and $V^2 = \frac{1}{N-1}\sum_{r=1}^{N}(a_{N,r} - \bar{a})^2$. As χ_0 has an asymptotic $\mathcal{X}^2$ distribution with $(k-1)$ degrees of freedom, the test rejects the null hypothesis of homoscedasticity H_0 if

$$\chi_0 > \mathcal{X}^2_{\alpha,k-1}$$

with the upper critical value at a significance level α of the $\mathcal{X}^2$ distribution with $k-1$ degrees of freedom denoted by $\mathcal{X}^2_{\alpha,k-1}$.

Table 4.3.: Hypothesis Testing for 108 tasks. We find that output transformations which tackle heteroscedasticity significantly improve GP modelling capabilities. Similarly, input transformations which tackle non-stationarity significantly improve GP modelling capabilities.

	Better	Sig. Better	Worse	Sig. Worse
Heteroscedasticity (Output Transform)	70 (65%)	58 (54%)	38 (35%)	25 (23%)
Non-Stationarity (Input Warping)	106 (98%)	79 (73%)	2 (2%)	0 (0%)

4.4.1. Answer A.I.: Simple Hyperparameter Tuning Tasks are Non-Stationary

To assess the impact of the extent of non-stationarity on BO performance, we conduct probabilistic regression experiments to gauge the predictive performance of a stationary GP on the hyperparameter tuning tasks with and without input warping transformations which correct for non-stationarity. We contend that the quality of the surrogate model is a good proxy for BO performance. We first run a two-sided paired t-test for each of the 108 tasks where the null hypothesis is that the application of the input warping yields no difference in the log probability metric. In Table 4.3 significance tests show that in 106/108 tasks, the log probability metric is more favourable when input warping is applied. In 79/108 tasks, the gain is significant at the 95% level of confidence (p-value < 0.025). We thus conclude that non-stationarity is an important consideration for BO performance due to the observed effect on the log probability metric.

4.4.2. Answers A.II.: Simple Hyperparameter Tuning Tasks are Heteroscedastic

We perform an analogous hypothesis test as in Section 4.4.1, assessing a vanilla GP's performance with and without output transformations (Box-Cox/ Yeo-Johnson). We run a two-sided paired t-test for each of the 108 tasks where the null hypothesis is that the application of the output transform yields no difference in the log probability metric. In Table 4.3 significance tests show that in 70/108 tasks, the log probability metric is more favourable when output transformations are applied. In 58/108 tasks, the gain is significant at the 95% level of confidence (p-value < 0.025). We thus conclude that heteroscedasticity is an important consideration for BO performance due to its impact on the log probability metric.

Furthermore, to gauge the level of heteroscedasticity in the underlying data, we use the Fligner-Killeen [70] and Levene [144] tests. For both tests, the null hypothesis is that the underlying black-box function noise process is homoscedastic. In all 108 tests, we see a p-value significantly lower than 0.05 in 72 tasks using Levene's test, and in 73 tasks using Fligner-Killeen. Such results (shown in detail in [52]) imply that at least 66% of the experimental tasks exhibit heteroscedastic behaviour.

4.4.3. Answer A.III.: No Clear Winner

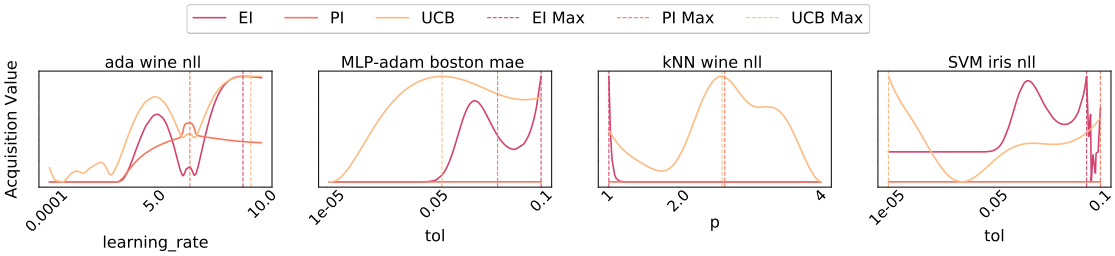


Figure 4.1.: Examples depicting conflicting acquisitions across data sets (Wine, Boston Housing, and Iris) and models (AdaBoost, Multilayer perceptron, K-Nearest neighbours, and support vector machines).

It has previously been observed that acquisition functions can conflict in their optima [211]. To provide further support for the answer to **Q.III.**, we collect 128 samples from each task by evaluating various hyperparameter configurations across metrics. We then assemble a data set $\mathcal{D} = \{\textbf{hyper-param}_i, y_i\}_{i=1}^{32}$, where $\textbf{hyper-param}_i$ is a vector with dimensionality dependent on the number of hyperparameters in a given model, and y_i is an evaluation metric, (e.g., mean squared error) We subsequently fit a GP surrogate

model and consider each of the three acquisition functions from Section 4.3.2. Given the difficulty involved in the graphical depiction of an acquisition function conflict in more than two dimensions, we examine a simple, two-dimensional illustrative example. From Figure 4.1, it is apparent that even in the simplest 2D case, many examples of conflicting acquisitions exist. Thus, in higher dimensions this behaviour will also occur.

4.5. Optimising Bayesian Optimisation

In this section we describe the component design choices that may mitigate for heteroscedastic and non-stationary aspects of commonly-encountered BO problems. Input and output transformations as well as multiobjective acquisition functions have been introduced in isolation previously, whilst acquisition function robustness is unique to this work. We synthesise the design choices in a single solution method which we term Heteroscedastic and Evolutionary Bayesian Optimisation (HEBO).

4.5.1. Tackling Heteroscedasticity and Non-Stationarity

To parsimoniously handle heteroscedasticity and non-stationarity, we leverage ideas from the warped GP literature [217] where output transformations facilitate the modelling of complex noise processes. We observe that the well-known `Box-Cox` [33] and `Yeo-Jonhson` [258] output transformations in conjunction with the `Kumaraswamy` [139] input transformation, offer a balance between simplicity of implementation and empirical performance. In our ablation study (Section 4.6), we demonstrate that the addition of these two modelling components alone yields large performance gains.

Output Transformation for Heteroscedasticity: We consider the `Box-Cox` transformation most frequently used as a corrective mapping for non-Gaussian data. The transform depends on a tuneable parameter ζ and applies the following map to each of the labels: $\mathrm{T}_\zeta(y_l) = y_l^\zeta - 1/\zeta$ for $\zeta \neq 0$ and $\mathrm{T}_\zeta(y_l) = \log y_l$ if $\zeta = 0$, where in our case y_l denotes the validation accuracy of the l^{th} hyperparameter configuration. ζ must be fit based on the observed data such that the distribution of the transformed labels closely resembles a Gaussian distribution. The transform is achieved by minimising the negative `Box-Cox` likelihood function

$$\log \left[\sum_{l=1}^{n} \frac{(\mathrm{T}_\zeta(y_l) - \overline{\mathrm{T}}_\zeta(\boldsymbol{y}))^2}{n} \right]^{\frac{n}{2}} + \sum_{l=1}^{n} \log \left[\mathrm{T}_\zeta(y_l) \right]^{(1-\zeta)},$$

where n is the number of datapoints and $\overline{\mathrm{T}_\zeta(\boldsymbol{y})}$ is the sample mean of the transformed labels. `Box-Cox` transforms only consider strictly positive (or strictly negative) labels y_l.

When labels take on arbitrary values, we use the `Yeo-Johnson` transform in place of the `Box-Cox` transform. The `Yeo-Johnson` transform is defined as follows

$$\mathtt{Y.J.}_\zeta(y_l) = \begin{cases} \frac{(y_l+1)^\zeta - 1}{\zeta} & \text{if } \zeta \neq 0,\ y_l \geq 0, \\ \log(y_l + 1) & \text{if } \zeta = 0,\ y_l \geq 0, \\ \frac{(1-y_l)^{2-\zeta} - 1}{\zeta - 2} & \text{if } \zeta \neq 2,\ y_l < 0, \\ -\log(1 - y_l) & \text{if } \zeta = 2,\ y_l < 0. \end{cases}$$

In an analogous fashion to the `Box-Cox` transform, the `Yeo-Johnson`'s parameter is fit based on the observed data through solving the following 1-dimensional optimisation problem

$$\max_\zeta -\frac{n}{2}\log\left[\frac{\sum_{j=1}^n(\mathtt{Y.J.}_\zeta(y_l) - \overline{\mathtt{Y.J.}_\zeta(\boldsymbol{y})})^2}{n-1}\right] + (\zeta - 1)\sum_{i=1}^n \left[\mathrm{sign}(y_l)\log(|y_l| + 1)\right],$$

with $\overline{\mathtt{Y.J.}_\zeta(\boldsymbol{y})}$ the sample average computed after applying the `Yeo-Johnson` transformation.

Input Transformations for Non-Stationarity: As a general solution concept for correcting for non-stationarity, we consider input warping see [218]. We rely on the `Kumaraswamy` input warping transform as used in [218], which operates as follows for each input dimension

$$[\mathtt{Kumaraswamy}_{\boldsymbol{\gamma}}(\boldsymbol{x}_l)]_k = 1 - (1 - [\boldsymbol{x}_l]_k^{a_k})^{b_k} \ \ \forall k \in [1:d],$$

where d is the dimensionality of the decision variable (i.e. the number of free hyperparameters), a_k and b_k are tuneable warping parameters for each of the dimensions, and $\boldsymbol{\gamma}$ is a vector concatenating all free parameters, i.e., $\boldsymbol{\gamma} = [a_{1:d}, b_{1:d}]^\mathsf{T}$. $\boldsymbol{\gamma}$ is fit based on the observed data. Similar to [18], we optimise $\boldsymbol{\gamma}$ under the marginal likelihood objective used to fit the GP surrogate.

All Modelling Improvements Together: Combining the above corrective measures for heteroscedasticity and non-stationarity leads us to an improved GP surrogate with more flexible modelling capabilities. The implementation of such a model is relatively simple

and involves maximising a new marginal likelihood which may be written as

$$\max_{\boldsymbol{\theta},\boldsymbol{\gamma}} -\frac{1}{2}\mathrm{T}_{\zeta^\star}(\boldsymbol{y})^\intercal(\boldsymbol{K}_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}}+\sigma_{\text{noise}}\boldsymbol{I})^{-1}\mathrm{T}_{\zeta^\star}(\boldsymbol{y})-\frac{1}{2}|\boldsymbol{K}_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}}+\sigma_{\text{noise}}^2\boldsymbol{I}|-\text{const},$$

with GP hyperparameters $\boldsymbol{\theta}$, $\boldsymbol{\gamma}$ indicates the use of non-stationary transformations, and $\zeta^\star$ denotes the solution to a `Box-Cox` likelihood objective. It is worth noting that we use `Box-Cox` as a running example but as mentioned previously we interchange `Box-Cox` with `Yeo-Johnson` transforms based on the properties of the label y_l. We use $\boldsymbol{K}_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}} \in \mathbb{R}^{n\times n}$ to represent a matrix such that each entry depends on both $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$, where $k_{\boldsymbol{\theta}}^{\boldsymbol{\gamma}}(\boldsymbol{x},\boldsymbol{x}') = k_{\boldsymbol{\theta}}(\texttt{Kumaraswamy}_{\boldsymbol{\gamma}}(\boldsymbol{x}),\texttt{Kumaraswamy}_{\boldsymbol{\gamma}}(\boldsymbol{x}'))$.

4.5.2. Tackling Acquisition Conflict & Robustness

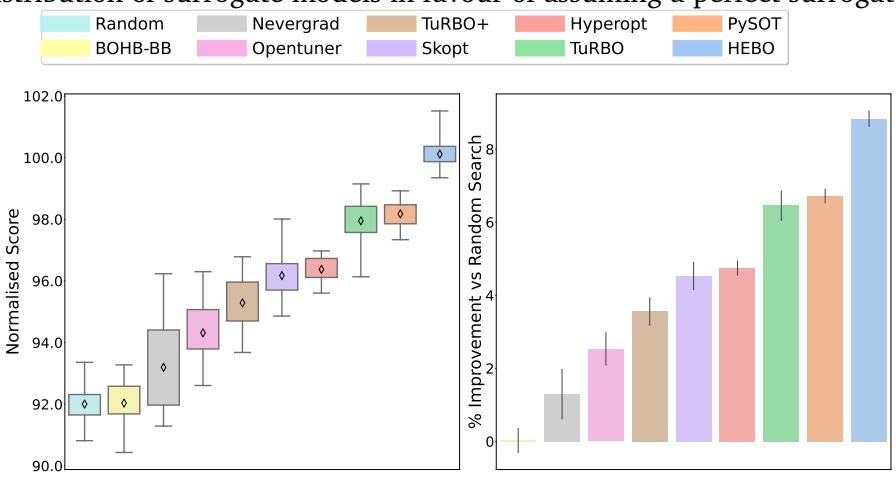
Having proposed modifications to the surrogate model component of the Bayesian optimisation scheme, we now turn our attention to the acquisition maximisation step. In particular, we focus on two considerations, the first related to the assumption of a perfect GP surrogate, and the second centred on conflicting acquisitions.

A Robust Acquisition Objective

As mentioned in Section 4.3.2, the acquisition maximisation step assumes that an adequate surrogate model is readily available. During early rounds of training especially, where data is scarce, such a property is often violated, leading to potentially severe model misspecification. One way to tackle such model misspecification is to adopt a robust formulation [128, 130] which attempts to identify the best-performing query location under the worst-case GP model, i.e., solving $\max_{\boldsymbol{x}} \min_{\boldsymbol{\theta}} \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D})$. Though such a formulation admits a solution $\boldsymbol{x}^\star$ that is robust to worst-case misspecification in $\boldsymbol{\theta}$, having a $\max\min$ acquisition is problematic for several reasons. From a conceptual perspective $\max\min$ formulations are known to lead to very conservative solutions if not correctly constrained or regularised since the optimiser possesses the power to impair the GP fit while updating $\boldsymbol{\theta}$[4]. From the perspective of implementation, one encounters two further issues. First, no global convergence guarantees are known for the non-convex, non-concave case that we [147], and second, ensuring gradients can propagate through the computation graph restricts surrogates and acquisition functions to be within the same programming framework.

[4]One can make a case for augmenting the objective with a constraint such that updates for $\boldsymbol{\theta}$ remain close to $\boldsymbol{\theta}^\star$ of the marginal likelihood. The ideal enforced proximity value however remains unclear in the robust acquisition literature to date [3, 128].

To avoid worst-case solutions and engender independence between acquisition functions and surrogate models, we leverage ideas from domain randomisation [234] and consider an expected formulation instead: $\max_{\boldsymbol{x}} \alpha_{\text{rob.}}(\cdot) \equiv \max_{\boldsymbol{x}} \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\sigma_\epsilon^2\boldsymbol{I})}\left[\alpha^{\boldsymbol{\theta}+\epsilon}(\boldsymbol{x}|\mathcal{D})\right]$. Importantly, this problem seeks to find new query locations that perform well on average over a distribution of surrogate models in favour of assuming a perfect surrogate.



(a) Empirical Performance Gain

Figure 4.2.: Analysis of the results on 108 tuning tasks. (Left) Normalised score comparison demonstrating that HEBO (i.e., BO with improvements from Section 4.5) outperforms competitor algorithms. We observe a 5% relative improvement to SOTA optimisers such as TuRBO. (Right) HEBO yields an 8% improvement compared to random search.

Multi-Objective Acquisition functions

As a final component of our general framework, we propose the use of multi-objective acquisitions seeking a Pareto-front solution. This formulation facilitates the process of "hedging" between different acquisitions such that no single acquisition dominates the solution [152]. Formally, we solve

$$\max_{\boldsymbol{x}} \left(\bar{\alpha}^{\boldsymbol{\theta}}_{\text{EI}}(\boldsymbol{x}|\mathcal{D}), \bar{\alpha}^{\boldsymbol{\theta}}_{\text{PI}}(\boldsymbol{x}|\mathcal{D}), \bar{\alpha}^{\boldsymbol{\theta}}_{\text{UCB}}(\boldsymbol{x}|\mathcal{D})\right), \tag{4.2}$$

with a robust acquisition of type $\in \{\text{EI}, \text{PI}, \text{UCB}\}$ denoted by $\bar{\alpha}^{\boldsymbol{\theta}}_{\text{type}}(\boldsymbol{x}|\mathcal{D})$, as introduced in the previous section. We also note that our formulation is designed to admit the use of a robust objective value of $\overline{\alpha}^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) = \alpha^{\boldsymbol{\theta}}(\boldsymbol{x}|\mathcal{D}) + \eta_k\sigma_n$ with η_k being a sample from $\mathcal{N}(0,1)$ at each iteration of the evolutionary solver.

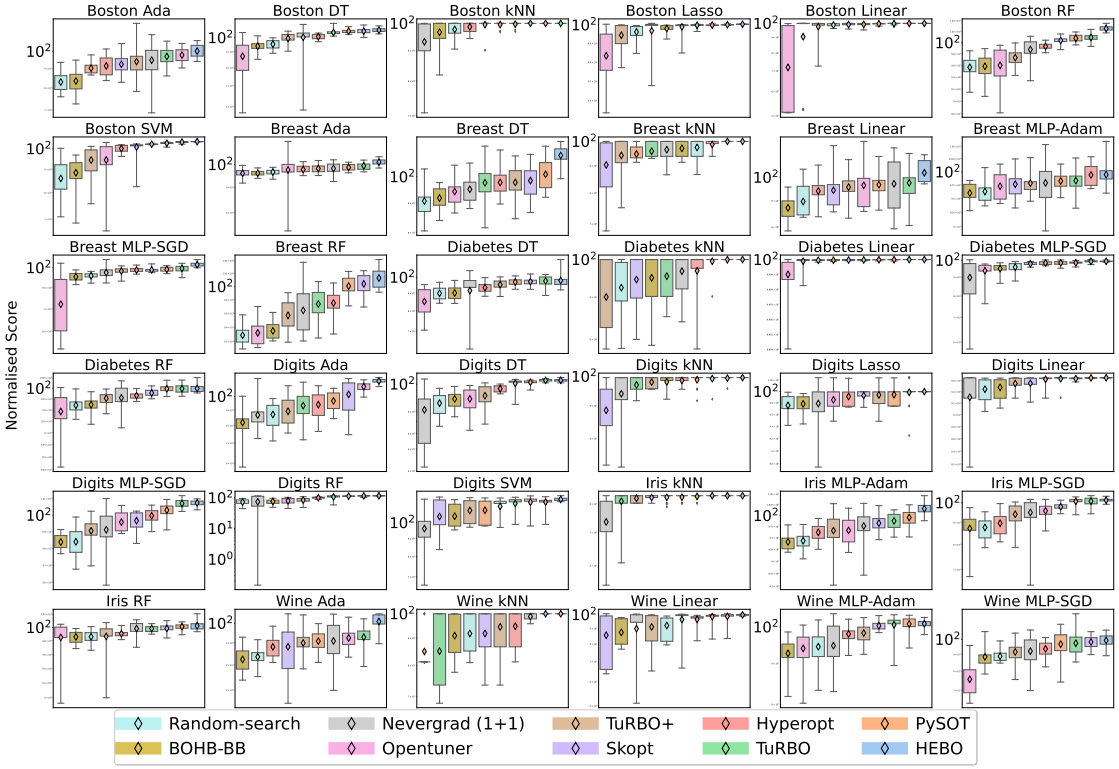


Figure 4.3.: `HEBO` compared against all baselines for 16 iterations and a batch size of 8 query points per iteration. Each experiment is repeated with 20 random seeds. We average each seed over both metrics in all tasks and display a subset of 36 summary plots for the 108 black-box functions. `HEBO` achieves the highest normalised mean score in 68.5% of the 108 black-box functions.

Although solving the problem in Equation (4.2) is a formidable challenge, we note the existence of many mature multi-objective optimisation algorithms. These range from first-order [127] to zero-order [150, 72] and evolutionary methods [99, 58]. Due to the discrete nature of hyperparameters in machine learning tasks, we advocate the use of evolutionary solvers that naturally handle categorical and integer-valued variables. In our experiments, we employ the non-dominated sorting genetic algorithm II (`NSGA-II`) which allows for mixed variable crossover and mutation to optimise real-valued and integer-valued inputs [58]. We use the implementation of `NSGA-II` found in the `Pymoo` [28] library. Alternatively, one may use the GP Hedge acquisition as used in `Dragonfly` [122] in

[108] or in SkOpt to select between acquisitions. We however, observed this formulation to perform poorly when compared against individual acquisitions.

4.6. Experiments and Results

In this section, we continue our empirical evaluation and validate gains (if any) that arise from the improvements proposed in Section 4.5. The experimental setup remains as described in Section 4.4. To assess performance, we use the normalised task score[5]. We run experiments on either 16 iterations with a batch of 8 query points per iteration or 100 iterations with 1 query point. Each experiment is repeated for 20 random seeds. We baseline against a wide range of solvers that either rely on BO-strategies or follow zero-order techniques such as differential evolution or particle swarms. These include SkOpt [175] [6] pySOT [7] a parallel global optimisation package [63], HyperOpt [23] [8], OpenTuner [9] a package for ensembling methods [10], NeverGrad [186] [10] a gradient-free optimisation toolbox (with the One Plus One optimiser), BOHB [67] [11] and Dragonfly [122] [12]. Additionally, we carried our modelling improvements to TuRBO [13] [65], augmenting the standard GP with mitigation strategies from Section 4.5 producing a new baseline that we entitle TuRBO+. Finally, we introduce Heteroscedastic Evolutionary Bayesian Optimisation (HEBO), in which we construct an optimiser with the improvements introduced in Section 4.5.

Implementation Details for BOHB: BOHB is a scalable hyperparameter tuning algorithm introduced in [67] mixing bandits and BO approaches to achieve both competitive anytime and final performances. Contrary to the other solvers considered in this paper, BOHB is specifically designed to tackle multi-fidelity optimisation and uses the Hyperband [146] routine to define the fidelity levels under which points are asynchronously evaluated. The selection of points follows a BO strategy based on the Tree Parzen Estimator (TPE) method. Given a data set $\mathcal{D}$ of observed data points and a threshold $\alpha \in \mathbb{R}$, the TPE models $p(\boldsymbol{x}|y)$,

[5] Note, we don't report the time to compute query points per algorithm as this was under 20 seconds per query point batch.

[6] https://github.com/scikit-optimize/scikit-optimize

[7] https://github.com/dme65/pySOT

[8] https://github.com/hyperopt/hyperopt

[9] https://github.com/jansel/opentuner

[10] https://github.com/facebookresearch/nevergrad

[11] https://github.com/automl/HpBandSter

[12] https://github.com/dragonfly/dragonfly

[13] https://github.com/rdturnermtl/bbo_challenge_starter_kit/

using kernel density estimates of

$$\ell(\boldsymbol{x}) = p(y < \alpha | \boldsymbol{x}, \mathcal{D}),$$
$$g(\boldsymbol{x}) = p(y \geq \alpha | \boldsymbol{x}, \mathcal{D}).$$

In the TPE algorithm, maximising the expected improvement criterion

$$\alpha_{\text{EI}}(\boldsymbol{x}) = \int \max(0, \alpha - p(y|\boldsymbol{x}))p(y|\boldsymbol{x})dy$$

is equivalent to maximising the ratio $r(x) = \frac{\ell(\boldsymbol{x})}{g(\boldsymbol{x})}$ which is carried out to select a single new candidate point at a time.

In the absence of a multi-fidelity setup in our experiments, we run a modified version of the `BOHB` algorithm implemented in the `HpBandSter` package. We leave the TPE method for modelling unchanged but ignore the fidelity level assignment from Hyperband. Moreover, as our experimental setup involves batch acquisitions, we tested two alternatives to the standard BOHB acquisition procedure to support synchronous suggestion of multiple points. In the first approach, we run q independent maximisation processes of $r(\boldsymbol{x})$ from random starting points and recover a single candidate from each process to form the q-batch suggestion. In the second approach, we obtain one point as a result of a single maximisation of $r(\boldsymbol{x})$ and we sample $q - 1$ random points to complete the q-batch suggestion. As the latter method yields better overall performance, the results reported under the `BOHB` label are obtained using the second approach.

4.6.1. Black-Box Functions

As discussed in Section 3, we evaluate black-box optimisation solvers on a large set of tasks from the `Bayesmark` package. Each task involves optimising the hyperparameters of a machine learning algorithm to minimise the cross validation loss incurred when this model is applied in a regression (reg) or a classification (clf) setting for a given data set. Thus, a task is characterised by a model, a data set and a loss function (metric) quantifying the quality of the regression or classification performance. In total, 108 distinct tasks can be defined from the valid combinations of the nine models specified in Table 4.1, the following six real-world UCI datasets [61], Boston (reg), Breast Cancer (clf), Diabetes (reg), Digits (clf), Iris (clf) and Wine (clf); the following two regression metrics, negative mean-squared error (MSE), negative mean absolute error (MAE), and two classification metrics, negative log-likelihood (NLL) and negative accuracy (ACC). The results reported in Figures 3 and 4 have been obtained by applying each black-box optimisation method using 16 iterations of 8-batch acquisition steps on all of the 108 tasks. In order to provide

a reliable evaluation of the different solvers, we repeated each run with 20 random seeds and considered the normalised score given by
with the best-achieved cross validation loss denoted by $\mathcal{L}$ at the end of the 16 acquisition steps, $\mathcal{L}^*$ is the estimated optimal loss for the task and $\mathcal{L}^{\text{rand}}$ is the mean loss (across multiple runs) obtained using random search with the same number of acquisition steps. The normalisation procedure permits aggregation of the scores across tasks despite the different cross-validation loss functions used.

4.6.2. Black-Box Optimisation Input Variables

We provide in Table 4.1 and Table 4.2 the list of the hyperparameters controlling the behaviour of each model along with their optimisation domains, which can differ depending on whether the model is used for a classification or a regression task. The search domain may include a mix of continuous and integer-valued variables (e.g. the MLP-SGD hyperparameter set includes an integer-valued hidden layer size, and a continuous-valued initial learning rate that can take on values between 10^{-5} and 10^{-1}). The dimensionality of the input space, i.e. the number of hyperparameters to tune, ranges from 2 to 9. We specify in the final column of the tables whether the search domain is modified through a standard transformation ($\log$ or logit) in order to facilitate optimisation.

Table 4.4 synthesises the performance achieved on the 108 tasks by the black-box optimisation solvers considered in our experiments. We note that the distribution of the scores attained by `HEBO` has the largest mean and the smallest standard deviation, indicating that `HEBO` significantly outperforms competitor algorithms.

Algorithm	Mean	Std	Median	40^{th} Centile	30^{th} Centile	20^{th} Centile	5^{th} Centile
`HEBO`	**100.12**	**8.70**	**100.01**	**100.00**	**99.88**	**98.64**	**85.71**
`PySOt`	98.18	9.03	100.00	99.81	98.60	95.36	80.00
`TuRBO`	97.95	10.80	100.00	99.88	98.75	95.26	78.63
`HyperOpt`	96.37	8.79	99.31	98.16	95.94	92.38	78.52
`SkOpt`	96.18	11.51	99.78	98.66	96.73	91.62	74.77
`TuRBO+`	95.29	10.93	98.97	97.60	95.27	90.92	74.77
`OpenTuner`	94.32	14.18	98.44	96.93	93.84	89.97	68.96
`NeverGrad`	93.20	17.52	99.65	97.84	94.57	88.28	55.34
`BOHB`	92.03	11.16	96.02	93.55	90.14	85.71	67.82
`Random-Search`	92.00	11.71	96.18	93.55	90.05	85.16	69.55

Table 4.4.: Mean and n-th percentile normalised scores over 108 black-box functions, each repeated with 20 random seeds. We observe significant mean improvements from `HEBO` compared to all competitor algorithms.

Figure 4.2 demonstrates gains from adopting the general `HEBO` framework. In

Table 4.5.: Number of tasks for which each optimiser performed best.

HEBO	TuRBO	PySOT	Skopt	Nevergrad (1+1)	BOHB-BO	Opentuner	Hyperopt	TuRBO+
71 (65.7%)	14 (13.0%)	7 (6.5 %)	5 (4.6 %)	4 (3.7 %)	3 (2.8%)	2 (1.9%)	1 (0.9%)	1 (0.9%)

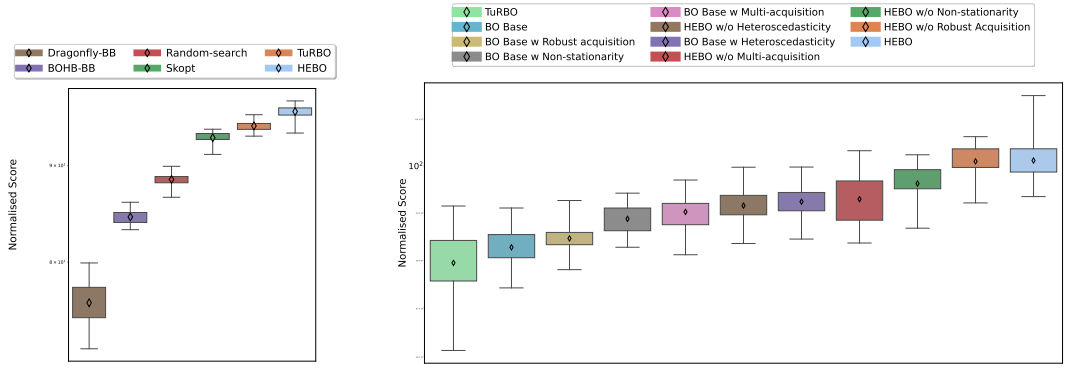
Figure 4.4, we compare HEBO against baselines and report up to an 8% performance gain relative to a random search strategy. It is also worth noting that TuRBO+ tends to underperform [14], achieving ca. 4% improvement relative to random search. We believe such a result is related to the interplay between our approach's capabilities to address heteroscedasticity and non-stationarity as well as the size of the trust regions; an interesting avenue that we plan to explore in future work. Overall, HEBO achieves the highest normalised mean scores on 71 of the 108 datasets, as shown in the summary Table 4.5. Complete results for each task can be found in [52].

Comparison to Asynchronous BO Algorithms: We perform a comparison to black-box optimisers, such as Dragonfly and BOHB, which operate in the asynchronous setting. We run each method for 100 iterations of data collection with a single query location per iteration. We label the asynchronous algorithms without their multi-fidelity components with an addition BB for black-box optimiser (Dragonfly-BB and BOHB-BB) to assess black-box optimisation performance only. The results of Figure 4.4a show that in the asynchronous setting, both Dragonfly-BB and BOHB-BB under-perform relative to other black-box optimisers, with HEBO performing best. However, this result is not surprising as asynchronous methods trade off sample efficiency with speed. Nevertheless, this experiment reveals a large gap in suggestion power between SOTA asynchronous and synchronous methods.

4.6.3. Ablation Results

To better understand the relative importance of each component of the HEBO algorithm, we conduct an ablation study by first removing each component of HEBO and testing the remaining components and second, by starting with basic BO and sequentially adding and testing each component of HEBO. The components comprise the consideration of heteroscedasticity, non-stationarity and robustness, as well as the use of a multiobjective acquisition function. We report average normalised scores in Figure 4.4b. The precedence order observed is: heteroscedasticity, multi-objective acquisition functions, non-stationarity and robustness.

[14] We believe this due to the trust region not being modelled correctly with input warping.



(a) Competitor Comparison. (b) Ablation Study

Figure 4.4.: (a) We compare `HEBO` against several popular hyperparameter tuning approaches including `BOHB-BB` and `Dragonfly-BB`, running all methods for 100 iterations with a batch size of 1 (i.e. one set of hyperparameters queried per iteration). `BOHB-BB` and `Dragonfly-BB` feature asynchronous queries, suggesting a batch of one set of hyperparameters at each iteration. We remove the multi-fidelity components from `BOHB` and `Dragonfly` to assess Black-Box optimisation alone, hence the additional BB appended to their label. (b) Ablation study where X denotes a general component of `HEBO`. `HEBO` w/o X takes one component X out at a time and BO Base w X adds one component X in at a time. We show `TuRBO` as a baseline and refer to `HEBO` with all significant components removed as `BO Base`. The ablation demonstrates that the corrections for each misspecified modelling assumption yield a tangible gain in empirical performance.

4.7. NeurIPS 2020 Black-Box Optimization Competition Results

Insights and analysis were released [237] from the organisers of the NeurIPS 2020 Black-box Optimization Challange (BBO). We highlight the strength of `HEBO` when deployed in a truly black-box task versus many existing SOTA baselines, as well as improved versions submitted to the competition by leading research labs.

Table 4.7, which is taken from [237], shows the top 20 submissions to BBO, as well as a collection of (at the time) SOTA Black-box optimizers. Firstly, in Table 4.7, we observe that `HEBO` is the best performing algorithm in the competition. This result was from evaluating all methods on a held out test set of tasks, for 20 seeds per method. Firstly, `HEBO` has the largest margin of difference for

Table 4.6.: Table from [237]. Variation in competition rankings from bootstrapping rankings. Ranking frequency shown as percentage, results demonstrate near 100% confidence in **HEBO** being the dominant algorithm. See [237] for details of how bootstrapped rankings are calculated.

Most likely ranking	2nd most likely ranking	3rd most likely ranking
Huawei Noah's Ark Lab (HEBO)	**Huawei Noah's Ark Lab (HEBO)**	**Huawei Noah's Ark Lab (HEBO)**
NVIDIA RAPIDS.AI	NVIDIA RAPIDS.AI	NVIDIA RAPIDS.AI
JetBrains Research	JetBrains Research	Duxiaoman DI
Duxiaoman DI	Optuna Developers	JetBrains Research
Optuna Developers	Duxiaoman DI	Optuna Developers
90%	5.2%	4.5%

mean normalised score out of all top 5 algorithms. Secondly, we notice `HEBO` is the only optimizer that is two orders of magnitude more efficient than random search.

[237] illustrate the relationship between the top ranking methods and baselines and random search. Specifically, they shows the mean normalised score achieved after 128 samples and time taken for random search to reach their mean normalised scores. **HEBO** is in fact the only algorithm which is two orders of magnitude more efficient than random search, taking random search >100 times more samples to reach its mean score.

4.8. Conclusion & Future Work

In this paper, we presented an in-depth empirical study of Bayesian optimisation. We demonstrated that even the simplest among machine learning problems can exhibit heteroscedasticity and non-stationarity. We also reflected on the affects of misspecified models and conflicting acquisition functions. We augmented BO algorithms with various enhancements and revealed that with a revised set of assumptions BO can in fact act as a competitive baseline in hyperparameter tuning. We hope this paper's findings can guide the community when employing BO in practice.

In the future, we wish to extend our analysis to high-dimensional domains, considering latent space optimisation [235, 93], investigate performance gains achievable through the choice of acquisition function optimiser [253, 91] and explore beyond hyperparameter tuning contexts such as molecule design [84, 87, 165, 93] and robotics [40].

Table 4.7.: Table from [237]. The top 20 results of the black-box optimisation challenge plus SOTA baselines. Results represent the 16 iterations of batches of size 8 (128 evaluations in total). RS Iters represents the number of iterations it took random search to reach the same mean score (as each method achieved in 128 evaluations). RS Efficiency is calculated through the ratio of RS Iters divided by 128.

Rank	Team	Score	Median	RS Iters.	RS Efficiency
1	**Huawei Noah's Ark Lab (HEBO)**	**93.519**	**99.166**	**15,512**	**121.188**
2	NVIDIA RAPIDS.AI	92.928	98.616	12,089	94.445
*	AutoML.org	92.551	98.693	10,353	80.883
3	JetBrains Research	92.509	99.131	10,179	79.523
4	Duxiaoman DI	92.212	99.027	9,032	70.562
5	Optuna Developers	91.806	99.156	7,698	60.141
6	Ambitious Audemer	91.107	96.668	5,899	46.086
7	jumpshot	91.089	97.056	5,861	45.789
8	KAIST OSI	90.872	98.659	5,409	42.258
9	Able Anteater	90.302	95.954	4,405	34.414
10	Oxford BXL	90.143	98.792	4,165	32.539
11	Innovatrics	90.081	97.062	4,076	31.844
12	IBM AI RBFOpt	90.050	96.117	4,032	31.500
13	Jim Liu	89.996	97.279	3,957	30.914
14	Jzkay	89.969	99.037	3,920	30.625
15	Better call Bayes	89.846	97.395	3,757	29.352
16	dannynguyen	89.706	98.800	3,581	27.977
17	AlexLekov	89.403	99.099	3,232	25.250
18	ABO	89.354	97.893	3,180	24.844
19	a2i2team	89.237	98.781	3,058	23.891
20	Tiny, Shiny & Don	89.229	96.513	3,050	23.828
-	TuRBO	88.921	98.927	2,756	21.531
-	pySOT	88.419	97.324	2,346	18.328
-	Scikit-Optimize	88.085	96.054	2,114	16.516
-	Ax	86.977	97.042	1,516	11.844
-	GpyOpt	85.384	94.443	978	7.641
-	hyperopt	82.389	93.506	477	3.727
-	Nevergrad (1+1)	80.012	92.681	288	2.250
-	pycma	78.658	95.285	220	1.719
-	OpenTuner	76.854	90.073	156	1.219
	`Random Search (RS)`	75.815	88.746	128	1.000

5. SAMBA: Safe Model-Based Active Reinforcement Learner

In this chapter we will introduce Safe Model Based Active Learner [53] (SAMBA), a model based reinforcement learning agent that is able to handle complex safety constraint efficiently. We first introduce model based/ constrained reinforcement learning in Section 5.2. In Section 5.3 we introduce the full framework, analyse the performance of the purposed framework in Section 5.4 and conclude with remarks in Section 5.5.

5.1. Introduction

Reinforcement learning (RL) has seen successes in many problems such as video and board games [162, 216, 161], and control of simulated robots [9, 204, 203]. Though successful, these applications assume idealised simulators and require tens of millions of agent-environment interactions, typically performed by randomly exploring policies. However, on the time scales of physical (i.e., real-world) systems, sample-efficiency naturally becomes a more pressing concern due to time and cost burdens. Additionally, there are often important safety considerations that one would like to integrate into the behaviour of the system. These can often be formalised through constraints on system trajectories (sequences of state and action pairs). For example, in autonomous drone navigation, one may desire to avoid sharp manoeuvres in cluttered environments, especially where there are people. A safety constraint can be integrated into the control optimisation formulation by quantifying the risk of safety violation by encoding a formal risk measure over the sequence of states (distance, velocity, acceleration, etc.), and actions (actuator control signals). Thus, taken together, the need for RL algorithms to integrate safety and sample-efficiency into a single coherent framework for real-world settings becomes clear.

To elaborate further on the issue of safety, there is a distinction to be made between requiring that a controlling agent/policy is safe (however that may be defined), and requiring that the learning process itself is safe. The latter generally requires assumptions to be made about the dynamics of the system as well as starting conditions (such as

access to a safe initial policy). For example, safety defined by constraining trajectories to safe regions of the state-action space is studied in [5] and partially-known control-affine dynamics with Lipschitz regularity conditions are assumed. In [14] and [133], strong assumptions on dynamics and initial control policies are required to give theoretical guarantees of safety. Safety in terms of Lyapunov stability [124] is studied in a model-free setting in [46, 47], which require a safe initial policy. Likewise, Lyapunov stability in a model-based setting is considered in [24], which requires Lipschitz dynamics and a safe initial policy. In [55] a decision layer is added on top of the policy in order to perturb the actions chosen by the policy so as to remain within safety constraints. The safety layer relies on the dynamics being linearisable, and to be learnt by, e.g., a neural network prior to policy training.

Whilst theoretical guarantees of safety are clearly appealing, the burden of extra assumptions and requirements needed is one we seek to avoid in this paper. As a trade-off, we are willing to tolerate some constraint violation during the learning process with the benefit of not requiring stringent assumptions on dynamics or a safe initial policy.

With regard to the form of the constraint, expectation (i.e., risk-neutral) constraints, which have a long history in the MDP literature [6], have been studied in a number of recent RL papers, e.g., [4, 55, 46, 47]. Whilst expectation constraints are sufficient for some purposes, they are somewhat limited as a risk measure. A more general risk measure, and the one we exploit in this paper, is Conditional Value-at-Risk (CVaR) [197], which is used in a number of RL papers, such as [48] (see discussion in Section 5.2.1). However, their approach is based on model-free methods for finite MDPs, and in particular, is not suitable for the continuous control use-cases considered in this paper.

Following the above discussion, in this paper we develop `SAMBA`, a model-based (deep) RL algorithm suitable for learning control in continuous state and action spaces, and which incorporates CVaR as a safety constraint. `SAMBA` exploits Gaussian processes [187] for dynamics learning, similar to PILCO [59], but does not adopt the moment-matching approach that PILCO takes in order predict forward-dynamics. Instead `SAMBA` performs policy updates by sampling trajectories from learned dynamics and applying standard deep RL techniques (policy gradient updates in an actor-critic framework). In the context of safe GP-based RL we must mention [181, 182], which still relies on moment matching to approximate the policy gradient, while our design relies on a stochastic gradient sampling strategy based upon simulations of the learnt dynamics. Furthermore, we also leverage *active* learning [73] to promote exploration near the starting state (Section 5.2.2), wherein the agent is influenced to acquire states reducing model uncertainty.

Successful application of active learning is very much predicated upon the chosen method of quantifying potential uncertainty reduction, i.e., what we refer to as the *acquisition function*. Common acquisition functions are those which identify points in

the model with large entropy or large variance, or where those quantities would be most reduced in a posterior model if the given point were added to the data set [135, 136, 69, 209]. However, our desire for safety adds a complication, since a safe learning algorithm will likely have greater uncertainty in regions where it is unsafe by virtue of not exploring those regions [59, 118]. Attacking the above challenges, we propose two novel acquisition functions for Gaussian processes that allow for exploration in novel areas while remaining close to training data, thus avoiding unsafe regions. To enable effective and grounded introduction of active exploration and safety constraints, we define a novel constrained bi-objective formulation of RL and provide a policy multi-gradient solver that is proven effective on a variety of safety benchmarks.

In short, our contributions can be stated as follows: 1) We define a novel constrained bi-objective formulation that trades off cost minimisation, exploration maximisation, and constraint feasibility. 2) We present a model-based deep RL algorithm for this constrained bi-objective formulation that is suitable for continuous state and action spaces and which exploits Gaussian processes. 3) We define safety-aware acquisition functions for exploration. We test our algorithm on three stochastic dynamical systems after augmenting these with safety regions and demonstrate a significant reduction in sample and cost complexities compared to the state-of-the-art.

As a final note, an alternative approach to using reinforcement learning would be to use model predictive control (MPC) [41] instead of computing a policy (i.e., general map from states to actions), and this is the method of some of the aforementioned papers [133, 25, 14, 118]. However, it is typically argued that an MPC controller is less robust than a policy controller [157]. In fact, robust MPC methods typically use a combination of a simple policy to ensure some form of robustness and an action plan to guarantee constraint satisfaction. Consequently, we chose to use a policy-based controller.

5.2. Background and notation

5.2.1. Reinforcement learning

We consider Markov decision processes (MDPs) with continuous states and action spaces; $\mathcal{M} = \langle \mathcal{X}, \mathcal{U}, \mathcal{P}, c, \gamma \rangle$, where $\mathcal{X} \subseteq \mathbb{R}^{d_{\text{state}}}$ denotes the state space, $\mathcal{U} \subseteq \mathbb{R}^{d_{\text{act}}}$ the action space, $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, \infty)$ is a transition density function, $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the cost function and $\gamma \in [0, 1]$ is a discount factor. At each time step $t = 0, \ldots, T$, the agent is in state $\boldsymbol{x}_t \in \mathcal{X}$ and chooses an action $\boldsymbol{u}_t \in \mathcal{U}$ transitioning it to a successor state $\boldsymbol{x}_{t+1} \sim \mathcal{P}(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t, \boldsymbol{u}_t)$, and yielding a cost $c(\boldsymbol{x}_t, \boldsymbol{u}_t)$. Given a state $\boldsymbol{x}_t$, an action $\boldsymbol{u}_t$ is sampled from a policy $\pi : \mathcal{X} \times U \to [0, \infty)$, where we write $\pi(\boldsymbol{u}_t | \boldsymbol{x}_t)$ to represent

the conditional density of an action. Upon subsequent interactions, the agent collects a trajectory $\boldsymbol{\tau} = [\boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T}]$, and aims to determine an optimal policy $\pi^\star$ by minimising total expected cost: $\pi^\star \in \arg\min_\pi \mathbb{E}_{\boldsymbol{\tau}\sim p_\pi(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau}) := \sum_{t=0}^{T} \gamma^t c(\boldsymbol{x}_t, \boldsymbol{u}_t)]$, where $p_\pi(\boldsymbol{\tau})$ denotes the trajectory density defined as: $p_\pi(\boldsymbol{\tau}) = \mu_0(\boldsymbol{x}_0) \prod_{t=0}^{T-1} \mathcal{P}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)\pi(\boldsymbol{u}_t|\boldsymbol{x}_t)$, with $\mu_0(\cdot)$ being an initial state distribution.

Constrained MDPs The above can be generalised to include various forms of constraints, often motivated by the desire to impose some form of safety measures. Examples are expectation constraints [4, 6] (which have the same form as the objective, i.e., expected discounted sum of costs), constraints on the variance of the return [184], chance constraints (a.k.a. Value-at-Risk (VaR)) [48], and Conditional Value-at-Risk (CVaR) [45, 48, 183]. The latter is the constraint we adopt in this paper for reasons that will be elucidated upon below. Adding constraints means we cannot directly apply standard algorithms like policy gradient [230], and different techniques are required, e.g., via Lagrange multipliers [26], as was done in [45, 48, 183] besides many others. Further, current methods only consider cost minimisation with no regard to exploration as we do in this paper.

Model-Based Reinforcement Learning Current solutions to the problem described above (constrained or unconstrained) can be split into model-free and model-based methods. Though effective, model-free algorithms are highly sample inefficient [105]. For sample-efficient solvers, we follow model-based strategies that we now detail. To reduce the number of interactions with the real environments, model-based solvers build surrogate models, $\mathcal{P}_{\text{surr}}$, to determine optimal policies. These methods, typically, run two main loops. The first gathers traces from the real environment to update $\mathcal{P}_{\text{surr}}$, while the second improves the policy using $\mathcal{P}_{\text{surr}}$ [59, 96]. Among various candidate models, e.g., world models [95], in this paper, we follow PILCO [59] and adopt Gaussian processes (GPs) as we believe that uncertainty quantification and sample efficiency are key for real-world considerations of safety. In this construction, one places a Gaussian process prior on a latent function f to map between input-output pairs. Such a prior is fully specified by a mean, $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$, and a covariance function $k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))]$ [187]. We write $f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$ to emphasize that f is sampled from a GP [187]. Given a data-set of n_1 input-output pairs $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n_1}$, corresponding, respectively, to state-action and successor state tuples, one can perform predictions on a query set of n_2 test data points $\{\boldsymbol{x}_\star^{(j)}\}_{j=1}^{n_2}$. Such a distribution is Gaussian with predictive mean-vectors and covariance matrices given by: $\boldsymbol{\mu}_\star = \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{y}_{n_1}$ and $\boldsymbol{\Sigma}_{n_2,n_2} = \boldsymbol{K}_{n_2,n_2} - \boldsymbol{K}_{n_2,n_1}\boldsymbol{A}_{n_1,n_1}\boldsymbol{K}_{n_1,n_2}$, where $\boldsymbol{A}_{n_1,n_1} = [\boldsymbol{K}_{n_1,n_1} + \sigma_\omega^2\boldsymbol{I}]^{-1}$ with σ_ω being the noise covariance that is assumed to be

Gaussian. In the above, we also defined $\boldsymbol{y}_{n_1}$ as a vector concatenating all training labels, $\boldsymbol{K}_{n_1,n_1} = K(\boldsymbol{X}, \boldsymbol{X}) = [k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})]_{ij}$, $\boldsymbol{K}_{n_1,n_2} = \boldsymbol{K}_{n_2,n_1}^{\mathbf{T}} = K(\boldsymbol{X}, \boldsymbol{X}_\star) = [k(\boldsymbol{x}^{(i)}, \boldsymbol{x}_*^{(j)})]_{ij}$, and $\boldsymbol{K}_{n_2,n_2}(\boldsymbol{X}_\star, \boldsymbol{X}_\star) = [k(\boldsymbol{x}_*^{(i)}, \boldsymbol{x}_*^{(j)})]_{ij}$, where $\boldsymbol{X}$ and $\boldsymbol{X}_\star$ are feature matrices with #input-dim $\times\, n_1$ and #input-dim $\times\, n_2$ sizes respectively. We executed training in GPyTorch [74], and used multi-output-GPs as defined in [255].

5.2.2. Active learning in dynamical systems

In *active learning* [69, 209], an agent chooses points to sample/query that best improve learning or model updates. Choosing the new query point is often performed by optimising an *acquisition function*, which gives some quantification of how much a model would improve if a given data point were queried, e.g., points where the model has high entropy or where variance can be most reduced. Active learning with GPs has been studied in the static case, where points can be selected at will (see, e.g., [135, 136]). In the context of dynamical systems, however, added complications arise as one is not always able to directly drive the system into a desired state. Recent work has attempted to resolve this problem, e.g., in [37] and [205], receding horizon optimisation is used to iteratively update a model, and in [37], actions are favoured that maximise the sum of differential entropy terms at each point in the mean trajectory predicted to occur by those actions. Moreover, in [205], a sum of variance terms is optimised to improve Bayesian linear regression. Again, for computational tractability, the predicted mean of states is used as propagating state distributions in the model is difficult. Different to our paper, neither of these works deal with safety, nor do they have additional objectives to maximise/minimise. In [112] a GP model that is used for MPC is updated by greedily selecting points which maximise *information gain*, i.e., reduction in entropy, as is done in [136]. Only very recently, the authors in [19] proposed an active learning approach coupled with MBRL. Similar to `SAMBA`, they use an adaptive convex combination of objectives, however their acquisition function is based on reward variance computed from a (finite) collection of models increasing the burden on practitioners who now need to predefine the collection of dynamics. They do not use GPs as we do, and do not consider safety. Compared to [19], we believe `SAMBA` is more flexible supporting model-learning from scratch and enabling principled exploration coupled with safety consideration. Further afield from our work, active learning has been recently studied in the context of GP time-series in [264], and for pure exploration in [215], which uses a finite collection of models. Our functions generalise the above to consider safe-regions and future information trade-off as we detail in Section 5.3.2.

5.3. SAMBA: Framework & solution

5.3.1. Solution Method

In designing SAMBA, we take PILCO [59] as a template and introduce two novel ingredients allowing for active exploration and safety. Following PILCO, SAMBA runs a main loop that gathers traces from the real environment and updates a surrogate model, $\mathcal{P}_{\text{GP}}(\cdot) : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, \infty)$, encoded by a Gaussian process. Given $\mathcal{P}_{\text{GP}}(\cdot)$, PILCO and other model-based methods [226] attempt to obtain a policy that minimises total-expect cost with respect to traces, $\boldsymbol{\tau}$, sampled from the learnt model by solving

$$\min_{\pi} \mathbb{E}_{\boldsymbol{\tau} \sim p_{\text{surr}}(\boldsymbol{\tau})} \left[\mathcal{C}(\boldsymbol{\tau}) \right]$$

with $\mathcal{C}(\boldsymbol{\tau}) = \sum_{t=0}^{T} \gamma^t c(\boldsymbol{x}_t, \boldsymbol{u}_t)$ and

$$p_{\text{surr}}(\boldsymbol{\tau}) = \mu_0(\boldsymbol{x}_0) \prod_{t=0}^{T-1} \mathcal{P}_{\text{GP}}(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t, \boldsymbol{u}_t) \pi(\boldsymbol{u}_t | \boldsymbol{x}_t)$$

Given a (GP) model of an environment, we formalise our problem as a constrained MDP with an additional active learning loss. We define a cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, constraint cost function (used to encode safety) $l : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, additional objective function $\zeta : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1)$.[1] In our setting, for instance, l encodes the state-action's risk by measuring the distance to an unsafe region, while $\boldsymbol{\zeta}$ denotes an acquisition function from these described in Section 5.3.2. To finalise the problem definition, we need to consider an approachable constraint to describe safety considerations. In incorporating such constraints, we are chiefly interested in those that are flexible (i.e., can support different user-designed safety criteria) and allow us to quantify events occurring in tails of cost distributions. When surveying constrained MDPs, we realise that the literature predominately focuses on expectation-type constraints [4, 189] – a not so flexible approach restricted to being safe on average. Others, however, make use of conditional-value-at-risk (CVaR); a coherent risk measure [48] that provides powerful and flexible notions of safety (i.e., can support expectation, tail-distribution, or hard – unsafe state visitation – constraints) and quantifies tail risk in the worst $(1-\alpha)$-quantile. Formally, given a random variable Z, $\text{CVaR}_\alpha(Z)$ is defined as: $\text{CVaR}_\alpha(Z) = \min_\nu [\nu + \frac{1}{1-\alpha}\mathbb{E}[(Z-\nu)^+]]$, where $(Z-\nu)^+ = \max(Z-\nu, 0)$.

With such a constraint, we can write the optimisation problem as:

$$\min_{\pi} \mathbb{E}_{\boldsymbol{\tau} \sim p_{\text{surr}}(\boldsymbol{\tau})} \left[\mathcal{C}(\boldsymbol{\tau}), \zeta(\boldsymbol{\tau}) \right]^{\mathbf{T}} \quad \text{s.t.} \quad \text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau})) \leq \xi, \tag{5.1}$$

[1] It is worth noting that the acquisition functions we develop in Section 5.3.2 map to $\mathbb{R}_+$ instead of $\mathbb{R}$.

with $\mathcal{L}(\boldsymbol{\tau}) = \sum_{t=0}^{T} \gamma^t l(\boldsymbol{x}_t, \boldsymbol{u}_t)$ being total accumulated safety cost along $\boldsymbol{\tau}$, and $\xi \in \mathbb{R}_+$ a safety threshold.

We transform this problem into a single objective one through a linear combination of $\mathcal{C}(\boldsymbol{\tau})$ and $\zeta_{\text{LOO}}(\boldsymbol{\tau})$. This relaxed yet constrained version is given by

$$\min_{\pi} \lambda_\pi \mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau})] - (1-\lambda_\pi)\mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})] \;\text{ s.t. }\; \text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau})) \leq \xi \tag{5.2}$$

where λ_π is a tuneable hyper-parameter in $[0, 1]$ and $\boldsymbol{\zeta}_{\text{LOO}}$ is an acquisition function defined in Section 5.3.2. Please note that the negative sign in the linear combination is due to the fact that we used $-\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})$.

From Constrained to Unconstrained Objectives We write an unconstrained problem using a Lagrange multiplier λ_{CVaR}:

$$\min_{\pi} \lambda^\star_\pi \mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\mathcal{C}(\boldsymbol{\tau})] - (1-\lambda^\star_\pi)\mathbb{E}_{\boldsymbol{\tau}\sim p_{\text{surr}}(\boldsymbol{\tau})}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})] + \lambda_{\text{CVaR}}[\text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau})) - \xi] \tag{5.3}$$

Due to non-convexity of the problem, we cannot assume strong duality holds, so in our experiments, we schedule λ_{CVaR} proportional to gradients using a technique similar to that in [203] that has proven effective.[2] To solve the above optimisation problem, we first fix ν and perform a policy gradient step in π.[3] To minimise the variance in the gradient estimator of the accumulated cost $\mathcal{C}(\boldsymbol{\tau})$ and the acquisition function $\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})$ (defined in Section 5.3.2), we build two neural network critics that we use as baselines. The first attempts to model the value of the standard cost, while the second learns information gain values. For the CVaR's gradient, we simply apply policy gradients. As CVaR is non-Markovian, it is difficult to estimate its separate critic. In our experiments, a heuristic where discounted safety losses as unbiased baselines was used and proved effective. In short, our main update equations when using a policy parameterised by a neural network

[2]Note that a primal dual-method as in [49] is not applicable due to non-convexity. In the future, we plan to study approaches from [83] to ease determining λ_{CVaR}.

[3]We resorted to policy gradients for two reason: 1) cost functions are not necessarily differentiable, and 2) better experimental behaviour when compared to model back-prop especially on OpenAI's safety gym tasks.

with parameters $\boldsymbol{\theta}$ can be written as:

$$\boldsymbol{\theta}^{[j][k+1]} = \boldsymbol{\theta}^{[j][k]} - \eta_k \Big(\mathbb{E}_{\boldsymbol{\tau}} \Big[\sum_{t \geq 1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_t | \boldsymbol{x}_t) \Big(\lambda_\pi^\star (Q_{\mathcal{C}}(\boldsymbol{x}_t, \boldsymbol{u}_t) - V_{\mathcal{C}}^{\phi_1}(\boldsymbol{x}_t)) \\ - (1 - \lambda_\pi^\star)(Q_{\boldsymbol{\zeta}}(\boldsymbol{x}_t, \boldsymbol{u}_t) - V_{\boldsymbol{\zeta}}^{\phi_2}(\boldsymbol{x}_t)) \Big) \Big] + \lambda_{\text{CVaR}} \nabla_{\boldsymbol{\theta}} \text{CVaR}_\alpha(\mathcal{L}(\boldsymbol{\tau})) \Big), \quad (5.4)$$

where η_k is a learning rate, and $V_{\mathcal{C}}^{\phi_1}(\boldsymbol{x}_t)$, $V_{\boldsymbol{\zeta}}^{\phi_2}(\boldsymbol{x}_t)$ are neural network critics with parameters ϕ_1 and ϕ_2. We present the main steps in Algorithm 2 and more details in the Appendix.

Algorithm 2 SAMBA: Safe Model-Based & Active Reinforcement Learning

Inputs: $\lambda_\pi^\star$, λ_{CVaR} initialisation, initial random policy π_1, $\mathcal{D}_0 = \{\}$, α-quantile, safety threshold ξ
for $j = 1 : \#$env-iterations **do:**
Sample traces from the real environment using π_j, concatenate all data and update $\mathcal{P}_{\text{GP}}^{[j]}(\cdot)$
for $k = 1 : \#$control-iterations **do:**
Sample traces from $\mathcal{P}_{\text{GP}}^{[j]}(\cdot)$ and compute $\boldsymbol{\zeta}_{\text{LOO}}^{[j][k]}(\boldsymbol{\tau})$
(§ 5.3.2) Update $\phi_1^{[j][k]}$ and $\phi_2^{[j][k]}$ with $\mathbb{E}_{\boldsymbol{\tau}}[\mathcal{C}(\boldsymbol{\tau})]$ and $\mathbb{E}_{\boldsymbol{\tau}}[\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau})]$ as targets, and $\boldsymbol{\theta}^{[j][k]}$ (§ 5.3.1)
Set $\pi_{j+1} = \pi_{\#\text{control-iterations}}$
Output: Policy $\pi_{\#\text{env-iterations}}$

The updated policy is then used to sample new traces from the real system where the above process repeats. During this sampling process, model-based algorithms consider various acquisition functions in acquiring transitions that reveal novel information, which can be used to improve the surrogate model's performance. PILCO, for instance, makes use of the GP uncertainty, while ensemble models [199, 8] explore by their aggregated uncertainties. With sufficient exploration, this allows policies obtained from surrogate-models to control real-systems. Our safety considerations mean we would prefer agents that learn well-behaving policies with minimal sampling from unsafe regions of state-action spaces; a property we achieve later by incorporating CVaR constraints as we detail in Section 5.3.1. Requiring a reduced number of visits to unsafe regions, hence, lessens the amount of "unsafe" data gathered in such areas by definition. Therefore, model entropy is naturally increased in these territories and algorithms following such exploration strategies are, as a result, encouraged to sample from hazardous states. As such, a naive adaptation of entropy-based exploration can quickly become problematic by contradicting safety requirements. To circumvent these problems, we introduce two new acquisition functions in Section 5.3.2, that assess information beyond training-data availability and consider input-output data distributions. Our acquisition functions operate

under the assumption that during any model update step, "safe" transition data (i.e., a set of state-action-successor states sampled from safe regions) is more abundant in number when compared to "unsafe" triplets. Considering such a skew between distributions, our acquisition functions yield increased values on test queries close to high-density training data. Given such an acquisition function, we enable novel model-based algorithms that attempts to minimise cost, while maximising active values and satisfying safety constraints.

Of course, the assumption of having skew towards safe regions in training data distribution is generally not true since solving the above only ensures good expected returns.

To frequently sample safe regions, we augment cost minimisation with a safety constraint that is encoded through the CVaR of a user-defined safety cost function with respect to *model* traces. Hence, SAMBA solves a constrained optimisation problem (§5.3.1) aimed at minimising cost, maximising active exploration, and meeting safety constraints.

5.3.2. Functions for safe active exploration

In general, ζ can be any bounded objective that needs to be maximised/minimised in addition to standard cost. Here, we choose one that enables active exploration in safe state-action regions. To construct ζ, we note that a feasible policy – i.e., one abiding by CVaR constraints – of the problem in Equation (5.1) samples tuples that mostly reside in safe regions. As such, the training data distribution is skewed in the sense that safe state-action pairs are more abundant than unsafe ones. Exploiting such skewness, we can indirectly encourage agents to sample safe transitions by maximising information gain functions that only grow in areas close-enough to training data.

ζ_{LOO}: Leave-One-Out acquisition Consider a GP dynamics model, $\mathcal{P}_{\text{GP}}$, that is trained on a state-action-successor-state data set $\mathcal{D} = \{\langle \tilde{\boldsymbol{x}}^{(i)}, y^{(i)} \rangle\}_{i=1}^{n_1}$ with $\tilde{\boldsymbol{x}}^{(i)} = (\boldsymbol{x}^{(i)}, \boldsymbol{u}^{(i)})$.[4] Such a GP induces a posterior allowing us to query predictions on n_2 test points $\tilde{\boldsymbol{x}}_\star = \{(\boldsymbol{x}_\star^{(j)}, \boldsymbol{u}_\star^{(j)})\}_{j=1}^{n_2}$. As noted in Section 5.2.1, the posterior is also Gaussian with the following mean vector and covariance matrix:[5] $\boldsymbol{\mu}_\star = \boldsymbol{K}_{n_2,n_1} \boldsymbol{A}_{n_1,n_1} \boldsymbol{y}_{n_1}$ and $\boldsymbol{\Sigma}_{n_2,n_2} = \boldsymbol{K}_{n_2,n_2} - \boldsymbol{K}_{n_2,n_1} \boldsymbol{A}_{n_1,n_1} \boldsymbol{K}_{n_1,n_2}$. Our goal is to design a measure that increases in regions with dense training-data (due to the usage of CVaR constraint) to aid agents in exploring novel yet safe tuples. To that end, we propose using an expected leave-one-out acquisition function between two Gaussian processes defined, for a one query data point $\tilde{\boldsymbol{x}}_\star$, as:

$$\zeta_{\text{LOO}}(\tilde{\boldsymbol{x}}_\star) = \mathbb{E}_{i \sim \text{Uniform}[1,n_1]} \left[\text{KL} \left(p(\boldsymbol{f}_\star | \mathcal{D}_{\neg i}) || p\left(\boldsymbol{f}_\star | \mathcal{D}\right) \right) \right]$$

[4] As $\mathcal{D}$ changes at every outer iteration, we simply concatenate all data in one larger data set; see Algorithm 2.

[5] For clarity, we describe a one-dimensional scenario.

with $\mathcal{D}_{\neg i}$ being $\mathcal{D}$ with point i left-out. Importantly, such a measure will only grow in regions which are close-enough to sampled training data, as posterior mean and covariance of $p(\boldsymbol{f}_\star|\mathcal{D}_{\neg i})$ shift by a factor that scales linearly and quadratically, respectively, with the total covariance between $\tilde{\boldsymbol{x}}_\star$ and $\boldsymbol{X}_{\neg i}$ where $\boldsymbol{X}_{\neg i}$ denotes a feature matrix with the i^{th} row removed.[6] In other words, such a acquisition function fulfils our requirement in the sense that if a test query is distant (in distribution) from all training input data, it will achieve low ζ_{LOO} score. Though appealing, computing a full-set of ζ_{LOO} can be highly computationally expensive, of the order of $\mathcal{O}(n_1^4)$ – computing $\boldsymbol{A}_{n_{1\neg i},n_{1\neg i}}$ requires $\mathcal{O}(n_1^3)$ and this has to be repeated n_1 times. A major source contributing to this expense, well-known in GP literature, is related to the need to invert covariance matrices. Rather than following variational approximations (which constitute an interesting direction), we prioritise sample-efficiency and focus on exact GPs. To this end, we exploit the already computed $\boldsymbol{A}_{n_1,n_1}$ during the model-learning step and make-use of the matrix inversion lemma [179] to recursively update the mean and covariances of $p(\boldsymbol{f}_\star|\mathcal{D}_{\neg i})$ for all i (see Appendix): $\boldsymbol{\mu}_\star^{(i)} = \boldsymbol{\mu}_\star - \boldsymbol{K}_{n_2,n_1}\frac{\boldsymbol{a}_i^{\mathbf{T}}\boldsymbol{a}_i}{a_{i,i}}\boldsymbol{y}_{n_1}$ and $\boldsymbol{\Sigma}_{n_2,n_2}^{(i)} = \boldsymbol{\Sigma}_{n_2,n_2} + \boldsymbol{K}_{n_2,n_1}\frac{\boldsymbol{a}_i^{\mathbf{T}}\boldsymbol{a}_i}{a_{i,i}}\boldsymbol{K}_{n_1,n_2}$, with $\boldsymbol{a}_i$ being the i^{th} row of $\boldsymbol{A}_{n_1,n_1}$. Hence, updating the inverse covariance matrix only requires computing and adding the outer product of the i^{th} row $\boldsymbol{A}_{n_1,n_1}$, divided by the i^{th} diagonal element. By exploiting the matrix inversion lemma we can reduce our complexity from $\mathcal{O}(n_1^4)$ to $\mathcal{O}(n_1^3)$. Note that now the main contributors to the computational cost of LOO are repeated matrix-vector, vector-vector multiplications, which can be efficiently distributed. Moreover, $\mathcal{O}(n_1^3)$ is also the complexity of the exact GP we rely on for the surrogate model, with the notable difference that the matrix inversions that are required for inference with a GP cannot be distributed as efficiently as the operations detailed above. As a consequence, one would not expect the computational budget to increase dramatically with the use of this acquisition function.

$\zeta_{\text{Bootstrap}}$: Bootstrapped symmetric acquisition We also experimented with another acquisition function that quantifies posterior sensitivity to bi-partitions of the data as measured by symmetric KL-divergence[7] (also called Jensen-Shannon divergence), averaged over possible bi-partitions: $\zeta_{\text{Bootstrap}}(\tilde{\boldsymbol{x}}_\star) = \mathbb{E}_{\langle\mathcal{D}_1,\mathcal{D}_2\rangle}[\text{KL}_{\text{sym}}(p(\boldsymbol{f}_\star|\mathcal{D}_1)||p(\boldsymbol{f}_\star|\mathcal{D}_2))]$, where $\langle\mathcal{D}_1,\mathcal{D}_2\rangle$ is a random bi-partition of the data $\mathcal{D}$. In practice, we randomly split the data in half, and do this K times (where K is a tuneable hyper-parameter) to get a collection of K bi-partitions. We then average over that collection. Similar to ζ_{LOO}, $\zeta_{\text{Bootstrap}}$ also assigns low importance to query points far from the training inputs, and hence, can be useful

[6] Though intuitive, we provide a formal treatment of the reason behind such growth properties in the Appendix.

[7] A symmetric KL-divergence between two distributions p, and q is defined as: $^{(\text{KL}(p\|q)+\text{KL}(q\|p))}/_2$.

for safe-decision making. In our experiments, ζ_{LOO} provided better-behaving exploration strategy, see Section 5.4.

Transforming $\zeta_{\cdot}(\tilde{\boldsymbol{x}}_\star)$ to $\zeta_{\cdot}(\boldsymbol{\tau})$ Both introduced functions are defined in terms of query test points $\tilde{\boldsymbol{x}}_\star$. To incorporate in Equation (5.1), we define trajectory-based expected total information gain as

$$\boldsymbol{\zeta}_{\text{LOO}}(\boldsymbol{\tau}) = \sum_{t=0}^{T} \gamma^t \zeta_{\text{LOO}}(\langle \boldsymbol{x}_t, \boldsymbol{u}_t \rangle),$$

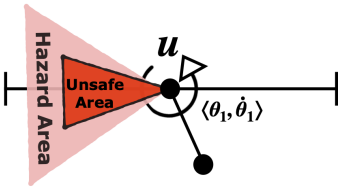
$$\boldsymbol{\zeta}_{\text{Bootstrap}}(\boldsymbol{\tau}) = \sum_{t=0}^{T} \gamma^t \zeta_{\text{Bootstrap}}(\langle \boldsymbol{x}_t, \boldsymbol{u}_t \rangle).$$

Interestingly, this characterisation trades off long-term versus short-term information gain similar to how cost trades-off optimal greedy actions versus long-term decisions. In other words, it is not necessarily optimal to seek an action that maximises immediate information gain since such a transition can ultimately drive the agent to unsafe states (i.e., ones that exhibit low $\zeta_{\cdot}(\tilde{\boldsymbol{x}})$ values). In fact, such horizon-based definitions have also recently been shown to improve modelling of dynamical systems [37, 215]. Of course, our problem is different in the sense that we seek safe policies in a safe decision-making framework, and thus require safely exploring acquisition functions.

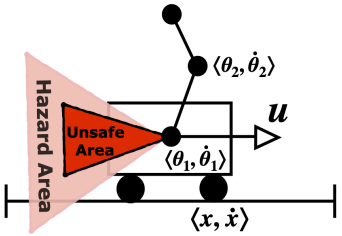
5.4. Experiments

We assess SAMBA in terms of both *safe learning* (train) and *safe final policies* (test) on five dynamical systems, two of which are adaptations of standard dynamical systems for MBRL (Safe Pendulum and Safe Cart-Pole Double Pendulum), while the third (Fetch Robot — optimally control end-effector to reach a 3D goal) we adapt from OpenAI's robotics environments [36]. Lastly, we use the car and point safe robotic task from OpenAI Safety Gym [189]. All environments and respective unsafe (hazard) regions are visualised in Figure 5.1. Total cost (TC) during training is the constraint cost the agent acquires throughout all training interactions with the environment, and similarly total cost during evaluation is the accumulated constraint cost at test time in the environment. Total violation (TV) similarly is the total constraint violations through training/testing. Note, constraints in general are not explicitly designed to minimise TV, and TV can be non-zero even in the optimality.

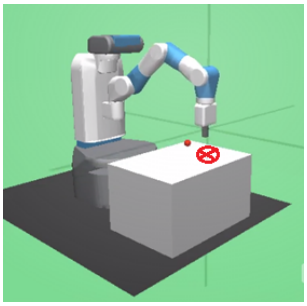
We limit all environments to have a maximum horizon/ trajectory length of 100 steps. We implemented a safety cost function with the following settings; The unsafe region



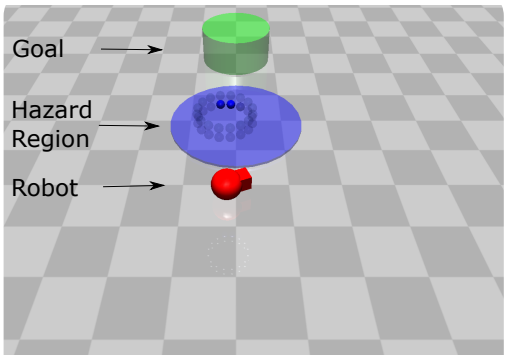
Safe Pendulum



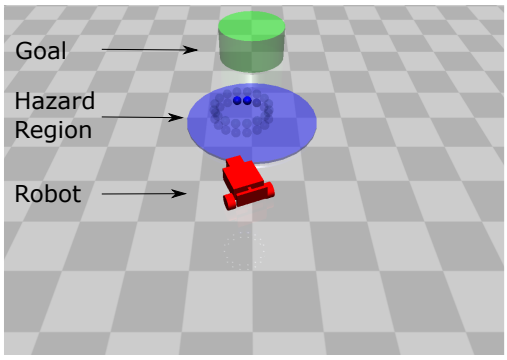
Safe Cartpole Double Pendulum



Safe Fetch Reacher



Safety point robot



Safety Car robot

Figure 5.1.: Environments with their respective unsafe regions visualised for evaluating safe algorithms.

started at $USR_{min} = 20\pi/180$ and ended at $USR_{max} = 30\pi/180$, therefore if θ_1 was in $USR_{min} \leq \theta_1 \leq USR_{max}$ we would refer to this as a violation and safety cost would be incurred. The Hazard Region contained within $HZ_{min} = USR_{min} - \pi/4 \leq HZ \leq USR_{max} + \pi/4 = HZ_{max}$. Where θ_1 is transformed to always remain in the region $\theta_1 \in [-\pi, \pi]$. Therefore, safety cost is linearly proportional to the distance from the edge of the hazard region (HZ_{min} or HZ_{max}) to the centre of the hazard region $(HZ_{max} + HZ_{min})/2$. We also implemented batched versions of the pendulum reward function as well as batched versions of the pendulum safety function using torch. We also implement the unsafe region centre as the middle between the fetchers starting position and its goal (with respect to the x position only), then the hazard region expands around this point with respect to $1/4$ of the total distance between the fetchers start state and goal state. Note, we terminated Safe Pendulum once the last 5 rewards in a trajectory were all ≤ -0.01, as this provides an adequately stabilised pendulum.

For the safety gym robots we take the following state values: velocities, magnetometer measurements, centre of mass position of the robot (all in the x and y axes) and gyroscope measurement (only in z axis). This choice is made to increase the dimension of the observation space, while also making the environment more GP-friendly. For instance, we remove the velocity, magnetometer and position readouts in the z axis as our car moves in a two dimensional plane. Similarly, we include only the measurement of the yaw angle rate, which is provided by the gyroscope measurement in the z axis. The robot's initial position is at $(0, 0)$ coordinates, while the goal is placed at $(3, 3)$ and marked in green in Figure 5.1. The centre of the hazard region of radius 0.5 (marked in blue in Figure 5.1), which the robot should avoid, is placed halfway between the robot's initial position and the goal. Note that we do not place other objects in the safety gym such as pillars, buttons etc.

In each of the benchmark tasks, we define unsafe regions as areas in state space and design the safety loss (i.e., $\mathcal{L}(\boldsymbol{\tau})$) to correspond to the (linearly proportional) distance between the end-effector's position (when in the hazard region) to the centre of the unsafe region. SAMBA implemented a more stable proximal update of Equation (5.4) following a similar method to [203]. We compare against algorithms from both model-free and model-based literature. Model-free comparisons against TRPO [204], PPO [203], CPO [4], STRPO (safety-constrained TRPO) [189] and SPPO (safety-constrained PPO) [189] enable us to determine if SAMBA improves upon the following: sample complexity during training (number of observations during training from the real system); total violations (TV), that is, the total number of timesteps spent inside the unsafe region; total accumulated safety cost.[8] Comparison with model-based solvers (e.g., PlaNet [97],

[8]Note, we report safe learning process TC, which is the total incurred safety cost throughout all training

PILCO [59], MBPO [115]) sheds light on the importance of our acquisition functions. We use PlaNet and MBPO as generic model-based deep reinforcement learning algorithm that use neural network dynamic models, i.e., we do not anticipate a considerably higher sample efficiency from other methods with neural network dynamic models. It is important to note that when implementing PILCO, we preferred a flexible solution that does not assume moment-matching and specific radial-basis function controllers. Hence, we adapted PILCO to support proximal policy updates, referred to as PPILCO in our experiments. In our comparisons, we will also employ a variant of SAMBA, which does not use active learning and call it SAMBA w/o active learning. This algorithm can be seen as a safety-constrained adaptation of PPILCO (i.e., PPILCO with CVaR constraint). The adaptation of PPILCO with active learning (i.e., SAMBA without CVaR constraints) performed very similarly to PPILCO and therefore not presented. Note, we do not show time complexity analysis as we did not observe significant differences between algorithms. As SAMBA introduces exploration components to standard model-based learners, we analysed these independently before combining them and reporting TV and TC (see Table 5.2).[8] All policies are represented by two-hidden-layer (32 units each) neural networks with $tanh$ non-linearities, which is one of the common choices for policy architectures in the RL literature. All hyper-parameters to reproduce our results can be found in the Appendix, we typically fixed all policy and critic parameters. The most crucial hyper-parameter to learning a safe policy was the cost constraint, whereby a simple random search over the range $[0.01, 0.05]$ led to safe policies on all environments. We also invested resources in tuning the gamma used in PPO, searching over the values $\gamma \in [0.99, 0.98, 0.97, 0.96, 0.95]$, although this was not detrimental to algorithms safety performance. We set $\lambda_\pi^\star = 0.9$, which provided reasonable results for all experiments.

Acquisition Function Component Evaluating our (acquisition) functions, we conducted an analysis that reports a 2D projection view of the state space at various intervals in the data-collection process. We compare $\zeta_{\text{LOO}}(\boldsymbol{\tau})$ and $\zeta_{\text{Bootstrap}}(\boldsymbol{\tau})$ against a *Max Entropy* [214] acquisition function and report the results on two systems in Figure 5.2. Our results suggest that LOO is much more conservative and will aim to guide the policy away from the dangerous area. At the same time, entropy encourages exploration into the unsafe area, additionally encouraged with the increase of number of samples - a highly undesirable property when safe active exploration is required. Similar results

environment interactions, and safe evaluation TC and TV, which similarly is the total incurred safety cost during evaluation, and the total violations from the timesteps spent inside the unsafe region during evaluation. Another metric used to compare safety algorithms is samples used for training, this is often referred to purely as samples, and is defined as the numbers of observations of the real system.

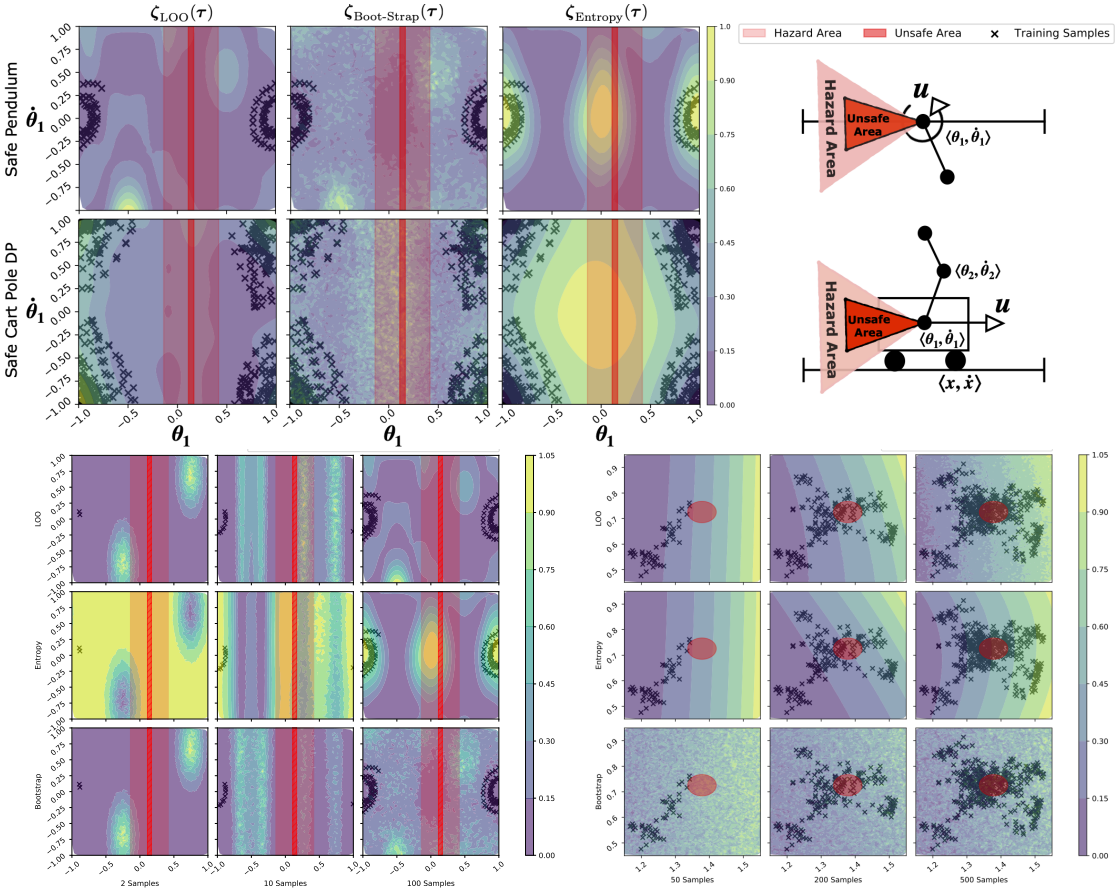


Figure 5.2.: Comparing acquisition function values across the entire state space. Note, policies used within this framework are encouraged to visit regions with higher (yellow) values. We first generate samples by random rollouts, as is typical in initial learning stages for most deep reinforcement learning algorithms. A GP dynamics model for each environment was then trained on 100 samples. The samples used to train the models are marked with black crosses. We then plot the varying acquisition function values using this GP for varying acquisition functions. We show Safe Cartpole Double Pendulum (top), Safe Pendulum (bottom left) and Safe Fetch (bottom).

Table 5.1.: Constraint satisfaction (✓) or constraint violation (×) on Safe Cart Pole Double Pendulum and Safe Fetch of model-based (orange) and model-free (blue) solvers. We evaluated expectation constraints (Exp.) and CVaR. Results show SAMBA satisfied safety constraints, outperforming others in different quartiles. There is also a clear trade-off between returns and safety for all algorithms. Interestingly, SAMBA is safer yet acquiring acceptable expected returns – close to SPPO for instance. Note, all policies shown below solve the given tasks, i.e., reach the target states.

Learner	**Safe Cart Pole Double Pendulum**						**Safe Fetch Reach**					
	Quartile			Constraint			Quartile			Constraint		
	0.25	0.5	0.75	Exp.	CVaR_α	$\mathbb{E}_\tau[\mathcal{C}(\tau)]$	0.25	0.5	0.75	Exp.	CVaR_α	$\mathbb{E}_\tau[\mathcal{C}(\tau)]$
PPO	5.0	5.7	9.4	×	×	**-19**	0.95	1.25	1.88	×	×	**-0.26**
PPILCO	5.0	5.8	8.6	×	×	-21	0.91	1.22	1.45	×	×	-0.36
PlaNet	5.2	6.0	8.8	×	×	-21	0.64	1.09	1.88	×	×	-0.39
MBPO	4.4	5.9	9.6	×	×	-21	0.66	1.56	2.01	×	×	-0.37
TRPO	5.1	5.9	7.0	×	×	-22	0.69	0.79	0.92	×	×	-0.58
CPO	0.41	0.47	0.96	×	×	-23	0.81	0.92	1.02	×	×	-0.43
PlaNet w RS	0.83	0.94	1.11	×	×	-24	-	-	-	-	-	-
SAMBA												
(w/o active learning)	0.21	0.28	0.33	×	×	-25	0.45	0.77	1.13	✓	×	-0.57
STRPO	0.22	0.27	0.32	✓	×	-28	0.14	0.46	0.86	✓	×	-0.98
SPPO	0.19	0.24	0.29	✓	×	-27	0.27	0.44	0.71	✓	✓	-1.51
SAMBA	**0.15**	**0.21**	**0.24**	✓	✓	-27	**0.00**	**0.05**	**0.19**	✓	✓	-2.27

Learner	**Safety Gym Car**						**Safety Gym Point**					
	Quartile			Constraint			Quartile			Constraint		
	0.25	0.5	0.75	Exp.	CVaR_α	$\mathbb{E}_\tau[\mathcal{C}(\tau)]$	0.25	0.5	0.75	Exp.	CVaR_α	$\mathbb{E}_\tau[\mathcal{C}(\tau)]$
PPO	7.1	7.8	8.0	×	×	2.4	0.79	1.15	18.3	×	×	0.24
PPILCO	7.5	7.7	8.1	×	×	2.1	0.83	1.01	14.9	×	×	0.13
PlaNet	2.7	6.0	6.5	×	×	1.03	9.4	11.6	22.9	×	×	0.44
TRPO	3.5	6.7	7.3	×	×	2.14	0.02	12	23.3	×	×	0.43
CPO	2.8	4.3	4.8	✓	×	2.8	0.0	2.3	13.5	✓	×	0.12
PlaNet w RS	1.3	5.2	6.7	×	×	0.9	7.5	13.6	20.1	×	×	0.4
SAMBA												
(w/o active learning)	1.9	3.4	4.0	✓	✓	1.1	1.8	2.2	5.4	✓	×	0.6
STRPO	2.8	3.0	6.3	✓	×	1.9	0.17	1.16	5.20	✓	✓	0.13
SPPO	0.8	7.8	8.1	×	×	1.1	0	1.02	4.49	✓	✓	0.51
MBPO	1.7	7.4	8.2	×	×	1.15	8.5	12.1	15.2	×	×	0.39
SAMBA	0.7	**2.7**	**3.5**	✓	✓	0.92	0.00	**0.62**	**3.87**	✓	✓	0.58

Note: PlaNet w RS on Safe Fetch Reach diverged during training.

are demonstrated with the Fetch Reach robot task (in the Appendix). It is also worth noting that due to the high-dimensional nature of the tasks, visual analysis can only give indications. Still, empirical analysis supports our claim and performance improvements

are clear; see Table 5.2.

Table 5.2.: Safe learning and safe evaluation for Safe Pendulum, Safe Cart Pole Double Pendulum, Safe Fetch Reach, Safety Gym Car and Safety Gym Point. We averaged each policy's safe evaluation over three random seeds, and collected 10000 evaluation samples per seed. [8]

Learner	Safe Pendulum				Safe Cart Pole Double Pendulum				Safe Fetch Reach			
	Learning		Evaluation		Learning		Evaluation		Learning		Evaluation	
	Samples	TC	TV	TC	Samples	TC	TV	TC	Samples	TC	TV	TC
TRPO	1000	110	4.3	4.9	1000	310	10	23	1000	790	1.8	2.7
PPO	1000	81	3.6	5	1000	140	5.5	35	1000	670	1.6	3.1
PlaNet	100	56	4.5	5.2	40	2.6	7.7	29	100	110	2.3	2.9
MBPO	90	49	4.3	5.6	50	3.2	6.9	31	80	101	2.7	3.1
PPILCO	2	0.04	2.8	4.5	6	0.097	7.1	33	5	0.49	1.9	3
PlaNet w RS	100	51	3.5	4.2	40	1.3	1.6	2.5	-	-	-	-
CPO	1000	58	4.7	4.4	1000	95	1.1	5	1000	160	1.9	2.7
STRPO	1000	38	2.0	2.1	1000	83	2.1	2.7	1000	170	1.7	2.1
SPPO	1000	65	1.7	2.4	1000	68	1	1.8	1000	290	1.5	2
SAMBA												
(w/o active learning)	1.8	0.02	1.7	2.1	6	0.062	1.2	1.6	5	0.37	1.4	2.2
SAMBA	**1.6**	**0.01**	**1.5**	**2.0**	**5**	**0.054**	**0.85**	**1.4**	**5**	**0.23**	**1.2**	**1.9**

Learner	Safety Gym Car				Safety Gym Point			
	Learning		Evaluation		Learning		Evaluation	
	Samples	TC	TV	TC	Samples	TC	TV	TC
TRPO	1000	202	3.7	5.6	1000	26.8	8.6	13
PPO	1000	205	7.4	11	1000	9	6.1	9.2
PlaNet	100	3.5	3.2	4.9	100	2.6	7.7	16
MBPO	90	3.4	4.3	5.6	100	3.2	6.9	21
PPILCO	2	0.47	7.4	11	8	0.31	5.4	8.9
PlaNet w RS	100	3.4	3.1	3.9	100	2.5	7.9	13
CPO	1000	49	4.7	4.4	1000	14	1.1	5
STRPO	1000	64	2.0	2.1	1000	17	2.1	2.7
SPPO	1000	66	1.7	2.4	1000	10	1	1.8
SAMBA								
(w/o active learning)	0.8	0.13	1.4	4.8	0.7	0.56	1.8	2.5
SAMBA	0.6	**0.01**	1.3	**2.2**	0.5	**0.04**	**1.3**	**1.5**

The data is scaled by 10^3. Note: PlaNet w RS on Safe Fetch Reach diverged during training. We have stopped training the model-free approaches (PPO, TRPO, CPO, SPPO, STRPO) after collecting $1000 \cdot 10^3$ samples.

Learning and Evaluation Having seen initial results which highlight the benefit of using our acquisition functions, we then conducted learning and evaluation experiments comparing SAMBA against state-of-the-art methods.

Results reported in Table 5.2 demonstrate that SAMBA reduces the amount of training TC, and samples [8] compared to others. This reduction in training TC is expected while comparing SAMBA to model-free baselines (SPPO, CPO, STRPO), which are known to be sample inefficient, and PlaNet w RS, which was designed for high-dimensional environments. Interestingly, during safe evaluation (deploying learnt policy and evaluating performance), we see SAMBA's safety performance competitive with (if not significantly better than) policies trained for safety in terms of TC and TV. [8] Such results are interesting as SAMBA was never explicitly designed to minimise test TV, [8] but it was still able to acquire significant reductions. Of course, one might argue that these results do not convey a safe final policy, as violations are still non-zero. Recall, however, that we define safety in terms of CVaR constraints, which do not totally prohibit violations, rather, limit the average cost of excess violations beyond a (user-defined) risk level α. Indeed, as mentioned above, it is not possible to guarantee total safety without strong assumptions on dynamics and/or an initial safe policy (and of course, none of the algorithms in Table 5.2 achieve zero violations). Finally, note that SAMBA policies consistently deliver similar safety costs (TC and TV) as SPPO and STRPO, which can be explained by similar policy updates for all three methods.

Learning: Reducing Constraint Cost & Sample Complexity. Figure 5.3 shows that SAMBA significantly reduces the total acquired constraint cost (top), and number of samples used to train an agent. Figure 5.3 illustrates that sample efficiency does not come as a trade-off to safety (the values of TC and TV) at the test or the evaluation time. We can attribute this behaviour to use of a Gaussian Process being able to accurately model the transition function in few samples, allowing us to transfer the dangerous learning process from the real environment into the dynamics model.

Evaluation: Reducing Constraint Cost & Violations Figure 5.3 (top) shows large percentage improvements of SAMBA's constraint costs during evaluation in the real environment, compared with baselines. From our ablation studies (Table 5.2), we can clearly see these gains are due to the addition of both the CVaR metric and the active learning component combined. Figure 5.3 (bottom) tells the same story as evaluation constraint cost, although the raw numbers for violations are significantly lower than the constraint costs acquired, to the violation region being much smaller than the constraint cost inducing region.

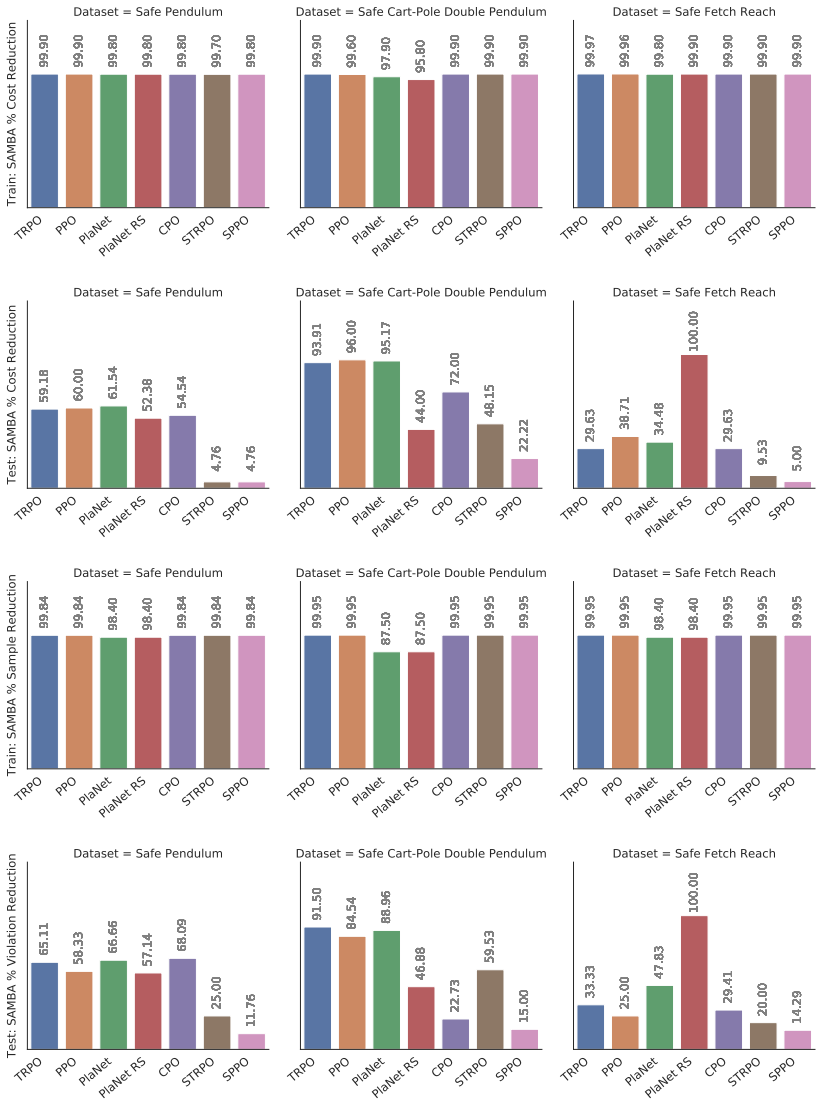


Figure 5.3.: (Top & second row) Percentage reduction of total constraint cost [8] acquired to train SAMBA compared to baselines for train and test. (Third & last row) Percentage reduction of samples [8] used to train SAMBA compared to baselines for train and test.

To evaluate risk performance, we conducted an in-depth study evaluating both expec-

tation and CVaR constraints on the two most challenging environments, Safe Cart Pole Double Pendulum and Safe Fetch Reach. We would expect all constrained methods to satisfy their constraint at test time, which surprisingly is not what we observe for CPO, seen in Table 5.1. Secondly, we observe that CVaR constrained methods are able to meet their constraints, as we as the expectation constraints. Thirdly, it is clear that active learning can play a crucial role in producing a constraint satisfying policy, as is seen in the ablation study in Safe Fetch Reach. Overall, Table 5.1 demonstrates that SAMBA achieves lower safety cost quartiles and lower expected safety cost, using cost limits $\xi = 0.25$ and $\xi = 0.75$. Table 5.1 demonstrates that SAMBA significantly outperforms others with respect to safety during evaluation, where we used a cost limit $\xi = 5$ and $\xi = 10$ for Safety Gym Car and Safety Gym Point environments, respectively. These results suggest that SAMBA produces safe-final policies in terms of its objective in Equation (5.1).

LOO performance analysis

This section empirically compares the typical entropy-based method with LOO. For comparison, we collected N random rollouts from Safe Pendulum to train our GP. Next, we calculated the entropy/LOO acquisition function for unseen random rollouts totaling $20,000$ test samples. Figure 5.2 shows the result of this evaluation, where we compared Entropy vs. LOO for differing sizes of training data N for the GP ($N = 20$, $N = 40$ and $N = 100$). The red box contains the USR (Unsafe Region). The experiment suggests that LOO is much more conservative and will aim to guide the policy away from the dangerous area. At the same time, entropy encourages exploration into the unsafe area, additionally encouraged with the increase of N - a highly undesirable property when safe active exploration is required.

5.5. Future Work

We proposed SAMBA, a safe and active model-based learner that makes use of GP models and solves an objective constraint problem to trade-off cost minimisation, exploration, and safety. We evaluated our method on three benchmarks, including ones from Open AI's safety gym and demonstrated significant reduction in training cost and sample complexity, as well as safe-final policies in terms of CVaR constraints compared to the state-of-the-art.

6. Conclusion

Many real-world datasets are scarce and expensive to collect, thus developing methods that are sample-efficient is crucial to practical application. The general aim of this thesis was to improve the sample efficiency of impact-full algorithms within machine learning. We have contributed to many aspects of sample-efficient optimisation, helping to bridge the gap between high-performing data-hungry machine learning models and practical machine learning models.

6.1. Summary of Contributions

In this thesis, we develop sample-efficient algorithms in two distinct sub-fields within machine learning, bayesian optimisation and safe reinforcement learning. Both bayesian optimisation and safe reinforcement learning require sample efficiency in maximising or minimising an objective function, however, differing in their environments ranging from static (bayesian optimisation) and dynamic (safe reinforcement learning). The contributions of this thesis are organised into three core chapters which can be read independently, however knowledge of the previous chapter will greatly help understand the contributions of the following chapter.

In Chapter 1, we give an introduction to the thesis, which includes an introduction to the core topic of Black-box optimisation and reinforcement learning. We then go on to state the motivating questions we wanted to answer in the thesis, the contributions as well as a general outline of the remainder of the thesis.

In Chapter 2, we give the formal mathematical background to understand all further contributions, specifically, we give background to bayesian optimisation, the most sample-efficient black-box optimisation strategy. We introduce all the components such as acquisition functions, acquisition maximisation etc.

In Chapter 3, we introduced novel formulations of popular acquisition functions in a mathematically equivalent compositional framework, allowing us to bridge the well-studied field of compositional optimisation together and apply the diverse range of optimisers to the acquisition maximisation step. We evaluate these newly create acquisition functions

on a range of datasets, such as synthetic batch bayesian optimisation benchmarks ranging from 16-120 dimensional tasks. We then extend evaluation to a suite of 180 unique, and challenging, hyperparameter tuning tasks. Throughout the evaluation, we separate optimisers by compositional/ non-compositional, as well as by their order, considering 0th, 1st and 2nd order optimizers. Overall, we show the benefits of utilising compositionality during acquisition maximisation across all orders of optimizers, as well as all datasets studies.

In Chapter 4, we first study popular hyperparameter optimisation tasks and highlight certain assumptions which are violated in common optimisers applied to even the simplest of tuning tasks, such as homoscedasticity and stationarity. We then develop a new bayesian optimisation algorithm HEBO, that tries to eliminate the negative consequence of these challenging properties of datasets/ tasks. We utilise a novel combination of some well-known, and less well-known components from machine learning literature, such as joint input and output warping, as well as multi-objective acquisition maximisation. We then show HEBO to be state-of-the-art during intrinsic evaluation against prior bayesian optimisation algorithms, as well as extrinsic evaluation where it won 1st place at the NeurIPS 2020 Black-box Optimisation competition [237] (BBO). Lastly, the effectiveness and superiority of HEBO led to it being used to tune a carefully selected subset of hyperparameters which results won the dual-task in the NeurIPS 2021 Machine Learning for Combinatorial Optimisation competition [77] (ML4CO). It was shown in [237] that HEBO was the only submission to be two orders of magnitude (128 times) more efficient for black-box optimisation than random search. The fact that it consecutively won international fame in two extremely challenging optimisation challenges, proves the significance of HEBO as a contribution to the machine learning community.

In Chapter 5 we introduce the new problem of reinforcement learning and safe reinforcement learning. We then utilise knowledge of probabilistic modelling and acquisition functions, to construct a novel model-based safe reinforcement learning algorithm named SAMBA. Specifically, we introduce novel acquisition functions for safe reinforcement learning e.g. LOO and Bootstrap KL. SAMBA can maximise reward, whilst minimising unsafe interactions through a constrained objective as well as attempting to simultaneously maximise an acquisition function which promotes exploration in safe areas of the state space that will improve the generalisation of the underlying model. We show through extensive experimentation that when visualising the acquisition function over the entire state space, for certain safety tasks it does indeed promote safer exploration. We study the effect of the probabilistic sample-efficient model, the safety constrained objective and the acquisition function on the overall sample efficiency and performance (in terms of reward and safety) of the resulting algorithm. We study SAMBA across a range of 5 challenging safety tasks and show that all components have a significant impact on

the overall safety of the algorithm during training and at deployment (test time). We also highlight how SAMBA is the only algorithm able to consistently satisfy expectation and CVaR safety constraints across the evaluated tasks, where most may only meet the expectation constraint occasionally.

The overall contributions of this thesis are as follows;

- We have introduced a new family of acquisition functions, that will further allow the development of novel acquisition maximisation methods (Chapter 3).
- An in-depth quantitative analysis of the dataset and task properties of common machine learning hyperparameter tuning tasks, leading to new insights and assumptions required to successfully tackle tuning tasks (Chapter 4).
- We have introduced a new sample-efficient Black-box optimisation algorithm HEBO (Chapter 4).
- Formed a unique connection between the fields of Bayesian optimisation and safe reinforcement learning through utilising acquisition functions in the agent objective function as an auxiliary loss (Chapter 5).
- Introduced a novel sample-efficient and safe reinforcement learning algorithm SAMBA that reduces sample complexity by several orders of magnitude versus existing safe methods (Chapter 5).

6.2. Future Work

While this thesis has made contributions towards sample-efficient optimisation ranging from static to dynamic environments, there is much left to improve upon and explore.

HEBO. For future work on improving HEBO, we wish to merge it with a method that trades sample-efficiency with speed, such as successive halving, and compare it to other SOTA multi-fidelity methods such as BOHB for asynchronous batch bayesian optimisation. Secondly, we wish to explore the use of the trust-region further in TuRBO and utilise a trust region in HEBO. Initial experimentation of trust regions in HEBO was un-fruitful, however, was under-explored due to time constraints. As we defined a hierarchical Bayesian model for this work, we would like to give a fully Bayesian treatment to it by integrating out the length-scales, noise and input and output warping parameters using Markov Chain Monte Carlo for further sample-efficiency improvements. Lastly, we would like to further improve the discrete kernels used in HEBO, such as utilising the transformed overlap kernel found in many SOTA combinatorial bayesian optimisation algorithms.

Compositional Bayesian Optimisation. In the future, we would like to see extensions of compositional acquisition functions to cover non-myopic acquisition functions, constrained and safe BO, as well as to investigate compositional structures of causal BO. It would prove useful to extend the evaluation of this work to a wider range of tasks, such as deep network hyperparameter tuning (e.g. on Transformers), tuning of controllers as well as various bayesian optimisation tasks found in life sciences such as protein design and material design. Lastly, it would be interesting to uncover the relationship between properties of black-box optimisation algorithms that might hint at whether a compositional or non-compositional acquisition function will perform best.

Safe Model-Based Active Learner. Lastly, for sequential decision-making under uncertainty with reinforcement learning, we wish to see other, more scalable, probabilistic surrogate models used, such as variational GPs to scale to larger dimensional control systems, and to apply our method in real-world robotics. Additionally, we would like to further study the formal construction of the constrained multi-objective Markov decision process, and the theoretical guarantees of `SAMBA` to support the empirical performance of well-behaving, exploratory, and safe policies. One of the strengths of our approach is the active learning component, which explores by sampling around the safe areas. The pitfalls of this exploration technique are that it can lead to overly cautious sampling impeding the task learning process. An overly cautious exploration can potentially be very restrictive in complex environments, such as a humanoid robot learning to stand, walk and eventually run. In this case, a bold exploration outside the seen data may be needed. While overcoming this limitation is outside the scope of this work, we wish to see this addressed in future work. Lastly, we believe whilst the acquisition functions promote safety, they have no inherent knowledge of safety, thus we would like to see further development of acquisition functions that provably promote safety in all environments. This could be done by constructing a probabilistic model of the safety function and then utilising an additional auxiliary loss based on an acquisition function of this model, however, there may be a more effective method for incorporating this safe knowledge jointly into the acquisition function constructed around the reward model.

Appendices

List of Algorithms

List of Figures

List of Tables

Bibliography

[1] Majid Abdolshah et al. “Multi-objective Bayesian optimisation with preferences over objectives”. In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019, pp. 12235–12245.

[2] Mohammed Amin Abdullah et al. “Wasserstein robust reinforcement learning”. In: *arXiv preprint arXiv:1907.13196* (2019).

[3] Mohammed Amin Abdullah et al. “Wasserstein robust reinforcement learning”. In: *arXiv preprint arXiv:1907.13196* (2019).

[4] Joshua Achiam et al. “Constrained policy optimization”. In: *International Conference on Machine Learning*. 2017, pp. 22–31.

[5] Anayo K Akametalu et al. “Reachability-based safe learning with Gaussian processes”. In: *IEEE Conference on Decision and Control*. 2014, pp. 1424–1431.

[6] Eitan Altman. *Constrained markov decision processes*. Vol. 7. CRC Press, 1999.

[7] Shun-ichi Amari. “Natural gradient works efficiently in learning”. In: *Neural Computation* 10.2 (1998), pp. 251–276.

[8] Joost van Amersfoort et al. “Simple and Scalable Epistemic Uncertainty Estimation Using a Single Deep Deterministic Neural Network”. In: *arXiv preprint arXiv:2003.02037* (2020).

[9] Haitham Bou Ammar et al. “Online multi-task learning for policy gradient methods”. In: *International Conference on Machine Learning*. 2014, pp. 1206–1214.

[10] Jason Ansel et al. “OpenTuner: An extensible framework for program autotuning”. In: *Proceedings of the 23rd International Conference on Parallel Architectures and Compilation*. 2014, pp. 303–316.

[11] Anup Aprem and Stephen Roberts. “A Bayesian Optimization Approach to Compute Nash Equilibrium of Potential Games Using Bandit Feedback”. In: *The Computer Journal* (2018).

[12] Kaito Ariu et al. “Regret in Online Recommendation Systems”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[13] Raul Astudillo and Peter Frazier. “Bayesian Optimization of Composite Functions”. In: *International Conference on Machine Learning*. 2019, pp. 354–363.

[14] Anil Aswani et al. “Provably safe and robust learning-based model predictive control”. In: *Automatica* 49.5 (2013), pp. 1216–1226.

[15] The GPyOpt authors. *GPyOpt: A Bayesian Optimization framework in Python*.

[16] Marco Baioletti et al. “Differential Evolution for Neural Networks Optimization”. In: *Mathematics* 8.1 (2020), p. 69.

[17] Maximilian Balandat et al. “BoTorch: A framework for efficient Monte-Carlo Bayesian optimization”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[18] Maximilian Balandat et al. “BoTorch: A framework for efficient Monte-Carlo Bayesian optimization”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[19] Philip Ball et al. “Ready Policy One: World Building Through Active Learning”. In: *arXiv preprint arXiv:2002.02693* (2020).

[20] Peter J Bentley. *Evolutionary design by computers*. Morgan Kaufmann, 1999.

[21] James Bergstra and Yoshua Bengio. “Random search for hyper-parameter optimization”. In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 281–305.

[22] James Bergstra et al. “Algorithms for hyper-parameter optimization”. In: *Advances in Neural Information Processing Systems* 24 (2011), pp. 2546–2554.

[23] James Bergstra et al. “Hyperopt: a Python library for model selection and hyperparameter optimization”. In: *Computational Science & Discovery* 8.1 (2015), p. 014008.

[24] Felix Berkenkamp et al. “Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 4661–4666.

[25] Felix Berkenkamp et al. “Safe model-based reinforcement learning with stability guarantees”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 908–918.

[26] Dimitri P Bertsekas. “Nonlinear programming”. In: *Journal of the Operational Research Society* 48.3 (1997), pp. 334–334.

[27] Dimitris Bertsimas, Omid Nohadani, and Kwong Meng Teo. “Nonconvex robust optimization for problems with constraints”. In: *INFORMS Journal on Computing* 22.1 (2010), pp. 44–58.

[28] Julian. Blank and Kalyanmoy. Deb. “Pymoo: Multi-Objective Optimization in Python”. In: *IEEE Access* 8 (2020), pp. 89497–89509.

[29] Jesús Bobadilla et al. “Recommender systems survey”. In: *Knowledge-Based Systems* 46 (2013), pp. 109–132.

[30] Ilija Bogunovic et al. “Adversarially robust optimization with Gaussian processes”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 5765–5775.

[31] Léon Bottou and Olivier Bousquet. “The tradeoffs of large scale learning”. In: *Advances in Neural Information Processing Systems* 20 (2007), pp. 161–168.

[32] Léon Bottou, Frank E Curtis, and Jorge Nocedal. “Optimization methods for large-scale machine learning”. In: *Siam Review* 60.2 (2018), pp. 223–311.

[33] George EP Box and David R Cox. “An analysis of transformations”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 26.2 (1964), pp. 211–243.

[34] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[35] Guy Bresler, Devavrat Shah, and Luis Filipe Voloch. “Collaborative filtering with low regret”. In: *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*. 2016, pp. 207–220.

[36] Greg Brockman et al. “Openai gym”. In: *arXiv preprint arXiv:1606.01540* (2016).

[37] Mona Buisson-Fenet, Friedrich Solowjow, and Sebastian Trimpe. “Actively Learning Gaussian Process Dynamics”. In: *arXiv preprint arXiv:1911.09946* (2019).

[38] Richard H Byrd et al. “A limited memory algorithm for bound constrained optimization”. In: *SIAM Journal on scientific computing* 16.5 (1995), pp. 1190–1208.

[39] Roberto Calandra. “Bayesian modeling for optimization and control in robotics”. PhD thesis. Darmstadt, Technische Universität, 2017.

[40] Roberto Calandra et al. “Manifold Gaussian processes for regression”. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 3338–3345.

[41] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[42] Xi-Ren Cao. “Convergence of parameter sensitivity estimates in a stochastic experiment”. In: *IEEE Transactions on Automatic Control* 30.9 (1985), pp. 845–853.

[43] Wei Chen, Yajun Wang, and Yang Yuan. “Combinatorial multi-armed bandit: General framework and applications”. In: *International Conference on Machine Learning*. 2013, pp. 151–159.

[44] Clément Chevalier and David Ginsbourger. “Fast computation of the multi-points expected improvement with applications in batch selection”. In: *International Conference on Learning and Intelligent Optimization*. Springer. 2013, pp. 59–69.

[45] Yinlam Chow and Mohammad Ghavamzadeh. “Algorithms for CVaR optimization in MDPs”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 3509–3517.

[46] Yinlam Chow et al. “A lyapunov-based approach to safe reinforcement learning”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 8092–8101.

[47] Yinlam Chow et al. “Lyapunov-based Safe Policy Optimization for Continuous Control”. In: *arXiv preprint arXiv:1901.10031* (2019).

[48] Yinlam Chow et al. “Risk-constrained reinforcement learning with percentile risk criteria”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6070–6120.

[49] Yinlam Chow et al. “Risk-sensitive and robust decision-making: a cvar optimization approach”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 1522–1530.

[50] Emile Contal et al. “Parallel Gaussian process optimization with upper confidence bound and pure exploration”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 225–240.

[51] Alberto Costa et al. “Black-box optimization of lighting simulation in architectural design”. In: *Complex Systems Design & Management Asia*. Springer, 2015, pp. 27–39.

[52] Alexander I Cowen-Rivers et al. “An empirical study of assumptions in bayesian optimisation”. In: *arXiv preprint arXiv:2012.03826* (2020).

[53] Alexander I Cowen-Rivers et al. “Samba: Safe model-based & active reinforcement learning”. In: *arXiv preprint arXiv:2006.09436* (2020).

[54] John P Cunningham, Philipp Hennig, and Simon Lacoste-Julien. “Gaussian probabilities and expectation propagation”. In: *arXiv preprint arXiv:1111.6832* (2011).

[55] Gal Dalal et al. “Safe exploration in continuous action spaces”. In: *arXiv preprint arXiv:1801.08757* (2018).

[56] Erik Daxberger et al. “Mixed-Variable Bayesian Optimization”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. July 2020, pp. 2633–2639.

[57] Alexander G De G. Matthews et al. “GPflow: A Gaussian process library using TensorFlow”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 1299–1304.

[58] Kalyanmoy Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197.

[59] Marc Deisenroth and Carl E Rasmussen. “PILCO: A model-based and data-efficient approach to policy search”. In: *International Conference on Machine Learning*. 2011, pp. 465–472.

[60] Luc Devroye. “The compound random search”. In: *International Symposium on Systems Engineering and Analysis*. 1972, pp. 195–110.

[61] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml`.

[62] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159.

[63] David Eriksson, David Bindel, and Christine A Shoemaker. “pySOT and POAP: An event-driven asynchronous framework for surrogate optimization”. In: *arXiv preprint arXiv:1908.00420* (2019).

[64] David Eriksson and Matthias Poloczek. “Scalable Constrained Bayesian Optimization”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 730–738.

[65] David Eriksson et al. “Scalable global optimization via local Bayesian optimization”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 5496–5507.

[66] Stefan Falkner, Aaron Klein, and Frank Hutter. “BOHB: Robust and Efficient Hyperparameter Optimization at Scale”. In: *International Conference on Machine Learning*. 2018, pp. 1437–1446.

[67] Stefan Falkner, Aaron Klein, and Frank Hutter. “BOHB: Robust and efficient hyperparameter optimization at scale”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1437–1446.

[68] Meherwar Fatima and M. Pasha. “Survey of Machine Learning Algorithms for Disease Diagnostic”. In: *Journal of Intelligent Learning Systems and Applications* 09 (2017), pp. 1–16.

[69] Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 2013.

[70] Michael A Fligner and Timothy J Killeen. “Distribution-free two-sample tests for scale”. In: *Journal of the American Statistical Association* 71.353 (1976), pp. 210–213.

[71] Peter I Frazier. “A tutorial on Bayesian optimization”. In: *arXiv preprint arXiv:1807.02811* (2018).

[72] Victor Gabillon et al. “Derivative-Free & Order-Robust Optimisation”. In: *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. Vol. 108. PMLR, 2020, pp. 2293–2303.

[73] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. “Deep bayesian active learning with image data”. In: *arXiv preprint arXiv:1703.02910* (2017).

[74] Jacob R Gardner et al. “GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration”. In: (2018).

[75] Jacob R Gardner et al. “GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration”. In: *Advances in Neural Information Processing Systems*. 2018.

[76] Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. “Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes”. In: *Neurocomputing* 380 (2020), pp. 20–35.

[77] Maxime Gasse et al. “The Machine Learning for Combinatorial Optimization Competition (ML4CO): Results and Insights”. In: *arXiv preprint arXiv:2203.02433* (2022).

[78] Alan Genz. “Numerical computation of multivariate normal probabilities”. In: *Journal of computational and graphical statistics* 1.2 (1992), pp. 141–149.

[79] Alan Genz. “Numerical computation of rectangular bivariate and trivariate normal and t probabilities”. In: *Statistics and Computing* 14.3 (2004), pp. 251–260.

[80] Saeed Ghadimi, Andrzej Ruszczynski, and Mengdi Wang. “A Single Timescale Stochastic Approximation Method for Nested Stochastic Optimization”. In: *SIAM Journal on Optimization* 30.1 (2020), pp. 960–979.

[81] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. *A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes*. Tech. rep. Mar. 2008. URL: https://hal.archives-ouvertes.fr/hal-00260579.

[82] Paul Glasserman. "Performance continuity and differentiability in Monte Carlo optimization". In: *1988 Winter Simulation Conference Proceedings*. IEEE. 1988, pp. 518–524.

[83] CJ Goh and XQ Yang. "Nonlinear Lagrangian theory for nonconvex optimization". In: *Journal of Optimization Theory and Applications* 109.1 (2001), pp. 99–121.

[84] Rafael Gómez-Bombarelli et al. "Automatic chemical design using a data-driven continuous representation of molecules". In: *ACS central science* 4.2 (2018), pp. 268–276.

[85] Alon Gonen and Shai Shalev-Shwartz. "Fast Rates for Empirical Risk Minimization of Strict Saddle Problems". In: *Conference on Learning Theory*. 2017, pp. 1043–1063.

[86] Javier González, Michael Osborne, and Neil Lawrence. "GLASSES: Relieving the myopia of Bayesian optimisation". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 790–799.

[87] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. "Constrained Bayesian optimization for automatic chemical design using variational autoencoders". In: *Chemical Science* 11.2 (2020), pp. 577–586.

[88] Ryan-Rhys Griffiths et al. "Achieving Robustness to Aleatoric Uncertainty with Heteroscedastic Bayesian Optimisation". In: *arXiv preprint arXiv:1910.07779* (2019).

[89] Ryan-Rhys Griffiths et al. "Achieving Robustness to Aleatoric Uncertainty with Heteroscedastic Bayesian Optimisation". In: *arXiv preprint arXiv:1910.07779* (2021). arXiv: 1910.07779 [stat.ML].

[90] Jean-Bastien Grill, Michal Valko, and Rémi Munos. "Black-box optimization of noisy functions with unknown smoothness". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*. 2015, pp. 667–675.

[91] Antoine Grosnit et al. "Are we forgetting about compositional optimisers in bayesian optimisation?" In: *Journal of Machine Learning Research* 22.160 (2021), pp. 1–78.

[92] Antoine Grosnit et al. “High-Dimensional Bayesian Optimisation with Variational Autoencoders and Deep Metric Learning”. In: *arXiv preprint arXiv:2106.03609* (2021).

[93] Antoine Grosnit et al. “High-Dimensional Bayesian Optimisation with Variational Autoencoders and Deep Metric Learning”. In: *arXiv preprint arXiv:2106.03609* (2021).

[94] Isabelle Guyon et al. “Analysis of the AutoML Challenge series 2015-2018”. In: *AutoML*. Springer series on Challenges in Machine Learning. 2019. URL: https://www.automl.org/wp-content/uploads/2018/09/chapter10-challenge.pdf.

[95] David Ha and Jürgen Schmidhuber. “World models”. In: *arXiv preprint arXiv:1803.10122* (2018).

[96] Danijar Hafner et al. “Dream to control: Learning behaviors by latent imagination”. In: *International Conference on Learning Representations*. 2020.

[97] Danijar Hafner et al. “Learning latent dynamics for planning from pixels”. In: *International Conference on Machine Learning*. 2019.

[98] Nadav Hallak, Panayotis Mertikopoulos, and Volkan Cevher. “Regret minimization in stochastic non-convex learning via a proximal-gradient approach”. In: *arXiv preprint arXiv:2010.06250* (2020).

[99] Nikolaus Hansen. “The CMA evolution strategy: A tutorial”. In: *arXiv preprint arXiv:1604.00772* (2016).

[100] Nikolaus Hansen and Andreas Ostermeier. “Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation”. In: *Proceedings of IEEE international conference on evolutionary computation*. IEEE. 1996, pp. 312–317.

[101] Elad Hazan. “Introduction to Online Convex Optimization.” In: *Found. Trends Optim.* 2.3-4 (2016), pp. 157–325.

[102] Philipp Hennig and Christian J Schuler. “Entropy search for information-efficient global optimization”. In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 1809–1837.

[103] James Hensman, Nicolò Fusi, and Neil D Lawrence. “Gaussian processes for Big data”. In: *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*. 2013, pp. 282–290.

[104] José Miguel Hernández-Lobato et al. “Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space”. In: *International Conference on Machine Learning*. 2017, pp. 1470–1479.

[105] Matteo Hessel et al. “Rainbow: Combining improvements in deep reinforcement learning”. In: *AAAI Conference on Artificial Intelligence*. 2018.

[106] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent*. Coursera: Neural Networks for Machine Learning. 2012.

[107] Matthew Hoffman, Eric Brochu, and Nando de Freitas. “Portfolio allocation for Bayesian optimization”. In: *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. 2011, pp. 327–336.

[108] Matthew Hoffman, Eric Brochu, Nando de Freitas, et al. “Portfolio Allocation for Bayesian Optimization.” In: *UAI*. Citeseer. 2011, pp. 327–336.

[109] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. “Sequential model-based optimization for general algorithm configuration”. In: *International Conference on Learning and Intelligent Optimization*. Springer. 2011, pp. 507–523.

[110] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. “Sequential Model-Based Optimization for General Algorithm Configuration”. In: *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*. LION’05. Rome, Italy: Springer-Verlag, 2011, pp. 507–523. ISBN: 9783642255656. DOI: 10.1007/978-3-642-25566-3_40. URL: https://doi.org/10.1007/978-3-642-25566-3_40.

[111] Christian Igel, Thorsten Suttorp, and Nikolaus Hansen. “A computational efficient covariance matrix update and a (1+ 1)-CMA for evolution strategies”. In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. 2006, pp. 453–460.

[112] Achin Jain et al. “Learning and control using Gaussian processes”. In: *ACM/IEEE International Conference on Cyber-Physical Systems*. 2018, pp. 140–149.

[113] Momin Jamil and Xin-She Yang. “A literature survey of benchmark functions for global optimisation problems”. In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013), pp. 150–194.

[114] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical reparameterization with Gumbel-softmax”. In: *International Conference on Learning Representations*. 2017.

[115] Michael Janner et al. “When to trust your model: Model-based policy optimization”. In: *arXiv preprint arXiv:1906.08253* (2019).

[116] Grahame A Jastrebski and Dirk V Arnold. “Improving evolution strategies through active covariance matrix adaptation”. In: *2006 IEEE International Conference on Evolutionary Computation*. IEEE. 2006, pp. 2814–2821.

[117] Donald R Jones, Matthias Schonlau, and William J Welch. “Efficient global optimization of expensive black-box functions”. In: *Journal of Global Optimization* 13.4 (1998), pp. 455–492.

[118] Sanket Kamthe and Marc Peter Deisenroth. “Data-efficient reinforcement learning with probabilistic model predictive control”. In: *International Conference on Artificial Intelligence and Statistics*. 2018.

[119] Kirthevasan Kandasamy et al. “Multi-fidelity Bayesian optimisation with continuous approximations”. In: *International Conference on Machine Learning*. 2017, pp. 1799–1808.

[120] Kirthevasan Kandasamy et al. “Neural architecture search with Bayesian optimisation and optimal transport”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 2018, pp. 2020–2029.

[121] Kirthevasan Kandasamy et al. “Parallelised Bayesian optimisation via Thompson sampling”. In: *International Conference on Artificial Intelligence and Statistics*. 2018, pp. 133–142.

[122] Kirthevasan Kandasamy et al. “Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly”. In: *Journal of Machine Learning Research* 21.81 (2020), pp. 1–27. URL: http://jmlr.org/papers/v21/18-223.html.

[123] Kristian Kersting et al. “Most likely heteroscedastic Gaussian process regression”. In: *Proceedings of the 24th International Conference on Machine Learning*. 2007, pp. 393–400.

[124] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*. Vol. 3. Prentice hall Upper Saddle River, NJ, 2002.

[125] Hyunjik Kim et al. “Attentive Neural Processes”. In: *International Conference on Learning Representations*. 2018.

[126] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[127] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014.

[128] Johannes Kirschner et al. “Distributionally robust Bayesian optimization”. In: *International Conference on Artificial Intelligence and Statistics*. 2020, pp. 2174–2184.

[129] Aaron Klein et al. “Robo: A flexible and robust bayesian optimization framework in python”. In: *NIPS 2017 Bayesian optimization workshop*. 2017.

[130] Aaron Klein et al. “Robo: A flexible and robust bayesian optimization framework in python”. In: *NIPS 2017 Bayesian optimization workshop*. 2017.

[131] Nicolas Knudde et al. *GPflowOpt: A Bayesian Optimization Library using TensorFlow*. 2017. arXiv: 1711.03845 [stat.ML].

[132] Jens Kober, J Andrew Bagnell, and Jan Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[133] Torsten Koller et al. “Learning-based Model Predictive Control for Safe Exploration”. In: *IEEE Conference on Decision and Control*. 2018, pp. 6059–6066.

[134] Ksenia Korovina et al. “Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 3393–3403.

[135] Andreas Krause and Carlos Guestrin. “Nonmyopic active learning of gaussian processes: an exploration-exploitation approach”. In: *International Conference on Machine Learning*. 2007, pp. 449–456.

[136] Andreas Krause, Ajit Singh, and Carlos Guestrin. “Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies”. In: *Journal of Machine Learning Research* 9.Feb (2008), pp. 235–284.

[137] Tipaluck Krityakierne, Taimoor Akhtar, and Christine A Shoemaker. “SOP: parallel surrogate global optimization with Pareto center selection for computationally expensive single objective problems”. In: *Journal of Global Optimization* 66.3 (2016), pp. 417–437.

[138] Scott R Kuindersma, Roderic A Grupen, and Andrew G Barto. “Variable risk control via stochastic optimization”. In: *The International Journal of Robotics Research* 32.7 (2013), pp. 806–825.

[139] Ponnambalam Kumaraswamy. “A generalized probability density function for double-bounded random processes”. In: *Journal of Hydrology* 46.1-2 (1980), pp. 79–88.

[140] Harold 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 (Mar. 1964), pp. 97–106.

[141] Manuel Laguna and Rafael Marti. “Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions”. In: *Journal of Global Optimization* 33 (Oct. 2005), pp. 235–255. DOI: 10.1007/s10898-004-1936-z.

[142] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[143] Miguel Lázaro-Gredilla and Michalis K Titsias. “Variational heteroscedastic Gaussian process regression”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 2011, pp. 841–848.

[144] Howard Levene. “Contributions to probability and statistics”. In: *Essays in Honor of Harold Hotelling* (1960), pp. 278–292.

[145] Lisha Li et al. “Hyperband: A novel bandit-based approach to hyperparameter optimization”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816.

[146] Lisha Li et al. “Hyperband: A novel bandit-based approach to hyperparameter optimization”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6765–6816.

[147] Tianyi Lin, Chi Jin, and Michael Jordan. “On gradient descent ascent for nonconvex-concave minimax problems”. In: *International Conference on Machine Learning*. 2020, pp. 6083–6093.

[148] Geert Litjens et al. “A survey on deep learning in medical image analysis”. In: *Medical Image Analysis* 42 (2017), pp. 60–88.

[149] D. C. Liu and J. Nocedal. “On the limited memory BFGS method for large scale optimization”. In: *Math. Programming* 45.3 (1989), pp. 503–528.

[150] Ilya Loshchilov and Frank Hutter. “CMA-ES for hyperparameter optimization of deep neural networks”. In: *arXiv preprint arXiv:1604.07269* (2016).

[151] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations*. 2019.

[152] Wenlong Lyu et al. “Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design”. In: *International Conference on Machine Learning*. 2018, pp. 3306–3314.

[153] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. “The concrete distribution: A continuous relaxation of discrete random variables”. In: *International Conference on Learning Representations*. 2017.

[154] Prasant Kumar Mahapatra, Susmita Ganguli, and Amod Kumar. “A hybrid particle swarm optimization and artificial immune system algorithm for image enhancement”. In: *Soft Computing* 19.8 (2015), pp. 2101–2109.

[155] Stanislav Markov. “SKOPT Documentation”. In: (2017).

[156] Juan Maronas et al. “Transforming Gaussian Processes With Normalizing Flows”. In: *arXiv preprint arXiv:2011.01596* (2020). arXiv: 2011.01596 [cs.LG].

[157] David Q Mayne, Marıa M Seron, and SV Raković. “Robust model predictive control of constrained linear systems with bounded disturbances”. In: *Automatica* 41.2 (2005), pp. 219–224.

[158] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. “Sparse Gaussian processes for Bayesian optimization”. In: *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*. 2016, pp. 517–526.

[159] Thomas P Minka. “A family of algorithms for approximate Bayesian inference”. PhD thesis. Massachusetts Institute of Technology, 2001.

[160] Thomas P Minka. “Expectation propagation for approximate Bayesian inference”. In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. 2001, pp. 362–369.

[161] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[162] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[163] Jonas Močkus. “On Bayesian methods for seeking the extremum”. In: *Optimization Techniques IFIP Technical Conference*. Springer. 1975, pp. 400–404.

[164] Henry Moss et al. “BOSS: Bayesian Optimization over String Spaces”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[165] Henry B Moss and Ryan-Rhys Griffiths. “Gaussian Process Molecule Property Prediction with FlowMO”. In: *arXiv preprint arXiv:2010.01118* (2020).

[166] Henry B Moss et al. “BOFFIN TTS: Few-Shot Speaker Adaptation by Bayesian Optimization”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7639–7643.

[167] Vu Nguyen and Michael A Osborne. “Knowing the what but not the where in Bayesian optimization”. In: *International Conference on Machine Learning*. 2020, pp. 7317–7326.

[168] ChangYong Oh, Efstratios Gavves, and Max Welling. “Bock: Bayesian optimization with cylindrical kernels”. In: *International Conference on Machine Learning*. 2018, pp. 3868–3877.

[169] Manfred Opper, Ole Winther, et al. “From naive mean field theory to the TAP equations”. In: *Advanced mean field methods: theory and practice* (2001), pp. 7–20.

[170] Michael A Osborne, Roman Garnett, and Stephen J Roberts. “Gaussian processes for global optimization”. In: *3rd International Conference on Learning and Intelligent Optimization (LION3)*. 2009, pp. 1–15.

[171] Art B Owen. “Quasi-Monte Carlo sampling”. In: *Monte Carlo Ray Tracing: Siggraph 2003 Course 44* (2003), pp. 69–88.

[172] Andrei Paleyes et al. “Emulation of physical processes with emukit”. In: *Second workshop on machine learning and the physical sciences, NeurIPS*. 2019.

[173] Chiwoo Park et al. “Robust Gaussian process regression with a bias model”. In: *arXiv preprint arXiv:2001.04639* (2020).

[174] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[175] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[176] Bo Peng and Lei Li. “An improved localization algorithm based on genetic algorithm in wireless sensor networks”. In: *Cognitive Neurodynamics* 9.2 (2015), pp. 249–256.

[177] Jan Peters, Katharina Mulling, and Yasemin Altun. “Relative entropy policy search”. In: *Twenty-Fourth AAAI Conference on Artificial Intelligence*. 2010.

[178] Jan Peters and Stefan Schaal. “Natural actor-critic”. In: *Neurocomputing* 71.7-9 (2008), pp. 1180–1190.

[179] KB Petersen, MS Pedersen, et al. “The Matrix Cookbook, vol. 7”. In: *Technical University of Denmark* 15 (2008).

[180] Nikolaos Ploskas et al. “Optimization of circuitry arrangements for heat exchangers using derivative-free optimization”. In: *Chemical Engineering Research and Design* 131 (2018), pp. 16–28.

[181] Kyriakos Polymenakos, Alessandro Abate, and Stephen Roberts. “Safe Policy Search Using Gaussian Process Models”. In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2019, pp. 1565–1573.

[182] Kyriakos Polymenakos et al. “SafePILCO: A Software Tool for Safe and Data-Efficient Policy Synthesis”. In: *Quantitative Evaluation of Systems*. Ed. by Marco Gribaudo, David N. Jansen, and Anne Remke. Springer International Publishing, 2020, pp. 18–26.

[183] LA Prashanth. “Policy gradients for CVaR-constrained MDPs”. In: *International Conference on Algorithmic Learning Theory*. Springer. 2014, pp. 155–169.

[184] LA Prashanth and Mohammad Ghavamzadeh. “Actor-critic algorithms for risk-sensitive MDPs”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 252–260.

[185] Kenneth V Price. “Differential evolution: a fast and simple numerical optimizer”. In: *Proceedings of North American Fuzzy Information Processing*. IEEE. 1996, pp. 524–527.

[186] J. Rapin and O. Teytaud. *Nevergrad - A gradient-free optimization platform*. `https://GitHub.com/FacebookResearch/Nevergrad`. 2018.

[187] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN: 026218253X.

[188] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.

[189] Alex Ray, Joshua Achiam, and Dario Amodei. *Benchmarking Safe Exploration in Deep Reinforcement Learning*. 2019. URL: `https://cdn.openai.com/safexp-short.pdf`.

[190] H Rechenberg et al. “Champs hyperfins et modele semi-microscopique non-local de l’invar”. In: *Journal of Physics and Chemistry of Solids* 34.7 (1973), pp. 1251–1265.

[191] Danilo Rezende and Shakir Mohamed. “Variational Inference with Normalizing Flows”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. 2015, pp. 1530–1538.

[192] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *Proceedings of the 31st International Conference on Machine Learning-Volume 32*. 2014, pp. II–1278.

[193] Martin Riedmiller and Heinrich Braun. “A direct adaptive method for faster backpropagation learning: The Rprop algorithm”. In: *IEEE International Conference on Neural Networks*. 1993.

[194] Sander van Rijn et al. “Evolving the structure of evolution strategies”. In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2016, pp. 1–8.

[195] Gonzalo Rios and Felipe Tobar. “Compositionally-warped Gaussian processes”. In: *Neural Networks* 118 (2019), pp. 235–246.

[196] Herbert Robbins and Sutton Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407.

[197] R Tyrrell Rockafellar, Stanislav Uryasev, et al. “Optimization of conditional value-at-risk”. In: *Journal of Risk* 2 (2000), pp. 21–42.

[198] Binxin Ru et al. “Bayesian optimisation over multiple continuous and categorical inputs”. In: *arXiv preprint arXiv:1906.08878* (2019).

[199] Rohan Saphal et al. “SEERL: Sample Efficient Ensemble Reinforcement Learning”. In: *arXiv preprint arXiv:2001.05209* (2020).

[200] Mark Schmidt, Nicolas Le Roux, and Francis Bach. “Minimizing finite sums with the stochastic average gradient”. In: *Mathematical Programming* 162.1-2 (2017), pp. 83–112.

[201] Robin M Schmidt, Frank Schneider, and Philipp Hennig. “Descending through a Crowded Valley–Benchmarking Deep Learning Optimizers”. In: *arXiv preprint arXiv:2007.01547* (2020).

[202] Günther Schrack and Mark Choit. “Optimized relative step size random searches”. In: *Mathematical Programming* 10.1 (1976), pp. 230–244.

[203] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[204] John Schulman et al. “Trust region policy optimization”. In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[205] Matthias Schultheis et al. “Receding Horizon Curiosity”. In: *Conference on Robot Learning*. 2019.

[206] MA Schumer and Kenneth Steiglitz. “Adaptive step size random search”. In: *IEEE Transactions on Automatic Control* 13.3 (1968), pp. 270–276.

[207] MA Schumer and Kenneth Steiglitz. “Adaptive step size random search”. In: *IEEE Transactions on Automatic Control* 13.3 (1968), pp. 270–276.

[208] Rajat Sen, Kirthevasan Kandasamy, and Sanjay Shakkottai. “Multi-fidelity black-box optimization with hierarchical partitions”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4538–4547.

[209] Burr Settles. *Active learning literature survey*. Tech. rep. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[210] Shweta B Shah and Nikolaos V Sahinidis. “SAS-Pro: Simultaneous residue assignment and structure superposition for protein structure alignment”. In: *PloS one* 7.5 (2012), e37493.

[211] Bobak Shahriari et al. “An entropy search portfolio for Bayesian optimization”. In: *arXiv preprint arXiv:1406.4625* (2014).

[212] Bobak Shahriari et al. “Taking the Human Out of the Loop: A Review of Bayesian Optimization”. In: *Proceedings of the IEEE* 1.104 (2016), pp. 148–175.

[213] Shai Shalev-Shwartz and Yoram Singer. “Online learning: Theory, algorithms, and applications”. In: (2007).

[214] Claude Elwood Shannon. “A mathematical theory of communication”. In: *ACM SIGMOBILE mobile computing and communications review* 5.1 (2001), pp. 3–55.

[215] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. “Model-based active exploration”. In: *International Conference on Machine Learning*. 2019.

[216] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489.

[217] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. “Warped Gaussian processes”. In: *Advances in Neural Information Processing Systems* 16 (2004), pp. 337–344.

[218] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian optimization of machine learning algorithms”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 2951–2959.

[219] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. “Practical Bayesian optimization of machine learning algorithms”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 2951–2959.

[220] Jasper Snoek et al. “Input warping for Bayesian optimization of non-stationary functions”. In: *International Conference on Machine Learning*. 2014, pp. 1674–1682.

[221] Jasper Snoek et al. “Scalable Bayesian Optimization Using Deep Neural Networks”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2171–2180. URL: http://proceedings.mlr.press/v37/snoek15.html.

[222] Jasper Snoek et al. “Scalable Bayesian optimization using deep neural networks”. In: *International conference on machine learning*. PMLR. 2015, pp. 2171–2180.

[223] Jialin Song, Yuxin Chen, and Yisong Yue. “A general framework for multi-fidelity Bayesian optimization with Gaussian processes”. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 3158–3167.

[224] Dmitrii V Speranskii. “Ant colony optimization algorithms for digital device diagnostics”. In: *Automatic Control and Computer Sciences* 49.2 (2015), pp. 82–87.

[225] Jost Tobias Springenberg et al. “Bayesian optimization with robust Bayesian neural networks”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 4134–4142.

[226] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. “Curl: Contrastive unsupervised representations for reinforcement learning”. In: *arXiv preprint arXiv:2004.04136* (2020).

[227] Niranjan Srinivas et al. “Gaussian process optimization in the bandit setting: no regret and experimental design”. In: *Proceedings of the 27th International Conference on Machine Learning*. 2010, pp. 1015–1022.

[228] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[229] Shiliang Sun et al. “A survey of optimization methods from a machine learning perspective”. In: *IEEE Transactions on Cybernetics* 50.8 (2019), pp. 3668–3681.

[230] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[231] Aditya R Thawani et al. “The photoswitch dataset: A molecular machine learning benchmark for the advancement of synthetic chemistry”. In: *arXiv preprint arXiv:2008.03226* (2020).

[232] William R Thompson. “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”. In: *Biometrika* 25.3/4 (1933), pp. 285–294.

[233] Michalis Titsias. “Variational learning of inducing variables in sparse Gaussian processes”. In: *Artificial Intelligence and Statistics*. 2009, pp. 567–574.

[234] Josh Tobin et al. “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[235] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. “Sample-efficient optimization in the latent space of deep generative models via weighted retraining”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[236] Ryan Turner et al. “Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020”. In: *arXiv preprint arXiv:2104.10201* (2021).

[237] Ryan Turner et al. “Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020”. In: *arXiv preprint arXiv:2104.10201* 1 (2021), p. 1.

[238] Rasul Tutunov, Haitham Bou-Ammar, and Ali Jadbabaie. “Distributed Newton method for large-scale consensus optimization”. In: *IEEE Transactions on Automatic Control* 64.10 (2019), pp. 3983–3994.

[239] Rasul Tutunov, Haitham Bou-Ammar, and Ali Jadbabaie. “Distributed SDDM solvers: Theory & applications”. In: *arXiv preprint arXiv:1508.04096* (2015).

[240] Rasul Tutunov et al. “Compositional ADAM: An Adaptive Compositional Solver”. In: *arXiv preprint arXiv:2002.03755* (2020).

[241] Michal Valko, Alexandra Carpentier, and Rémi Munos. “Stochastic simultaneous optimistic optimization”. In: *International Conference on Machine Learning*. 2013, pp. 19–27.

[242] Paolo Viappiani and Craig Boutilier. “Regret-based optimal recommendation sets in conversational recommender systems”. In: *Proceedings of the third ACM conference on Recommender systems*. 2009, pp. 101–108.

[243] Pauli Virtanen et al. “SciPy 1.0: fundamental algorithms for scientific computing in Python”. In: *Nature methods* 17.3 (2020), pp. 261–272.

[244] Jialei Wang et al. “Parallel Bayesian global optimization of expensive functions”. In: *Operations Research* (2020).

[245] Mengdi Wang, Ethan X Fang, and Han Liu. “Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions”. In: *Mathematical Programming* 161.1-2 (2017), pp. 419–449.

[246] Mengdi Wang and Ji Liu. “A stochastic compositional gradient method using Markov samples”. In: *2016 Winter Simulation Conference (WSC)*. IEEE. 2016, pp. 702–713.

[247] Mengdi Wang, Ji Liu, and Ethan X Fang. “Accelerating stochastic composition optimization”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 3721–3743.

[248] Zi Wang and Stefanie Jegelka. “Max-value Entropy Search for Efficient Bayesian Optimization”. In: *International Conference on Machine Learning*. 2017, pp. 3627–3635.

[249] Colin White, Willie Neiswanger, and Yash Savani. “BANANAS: Bayesian optimization with neural architectures for neural architecture search”. In: *arXiv preprint arXiv:1910.11858* (2019).

[250] Christopher KI Williams and Carl Edward Rasmussen. “Gaussian processes for regression”. In: (1996).

[251] Ashia C. Wilson et al. *The Marginal Value of Adaptive Gradient Methods in Machine Learning*. 2018. arXiv: 1705.08292 [stat.ML].

[252] James Wilson, Frank Hutter, and Marc Deisenroth. “Maximizing acquisition functions for Bayesian optimization”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 9884–9895.

[253] James Wilson, Frank Hutter, and Marc Deisenroth. “Maximizing acquisition functions for Bayesian optimization”. In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 9884–9895.

[254] James T Wilson et al. “Efficiently sampling functions from Gaussian process posteriors”. In: *International Conference on Machine Learning*. 2020.

[255] Taco de Wolff, Alejandro Cuevas, and Felipe Tobar. “MOGPTK: The Multi-Output Gaussian Process Toolkit”. In: *arXiv preprint arXiv:2002.03471* (2020).

[256] Jian Wu and Peter Frazier. “The parallel knowledge gradient method for batch Bayesian optimization”. In: *Advances in Neural Information Processing Systems* 29 (2016), pp. 3126–3134.

[257] Yaodong Yang et al. "α^α-Rank: Practically Scaling α-Rank through Stochastic Optimisation". In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 2020, pp. 1575–1583.

[258] In-Kwon Yeo and Richard A Johnson. "A new family of power transformations to improve normality or symmetry". In: *Biometrika* 87.4 (2000), pp. 954–959.

[259] Mingzhang Yin and Mingyuan Zhou. "Semi-Implicit Variational Inference". In: *International Conference on Machine Learning*. 2018, pp. 5660–5669.

[260] Kwang-Seon Yoo and Seog-Young Han. "Modified ant colony optimization for topology optimization of geometrically nonlinear structures". In: *International Journal of Precision Engineering and Manufacturing* 15.4 (2014), pp. 679–687.

[261] Matthew D Zeiler. "Adadelta: An adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[262] Zhuhong Zhang, Lei Wang, and Fei Long. "Immune optimization approach solving multi-objective chance-constrained programming". In: *Evolving Systems* 6.1 (2015), pp. 41–53.

[263] Ciyou Zhu et al. "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". In: *ACM Transactions on Mathematical Software (TOMS)* 23.4 (1997), pp. 550–560.

[264] Christoph Zimmer, Mona Meister, and Duy Nguyen-Tuong. "Safe Active Learning for Time-Series Modeling with Gaussian Processes". In: *Advances in Neural Information Processing Systems*. 2018, pp. 2730–2739.

A. Curriculum Vitae

A.1. Work experience

Sept 2022 – Present	**Research Engineer DeepMind** *Working in AlphaFold team.*
Sept 2019 – Aug 2022	**Research Scientist Huawei R&D London** *Papers accepted at ICML, CoRL, JMLR, JAIR, MLJ, AutoML.*
June 2019 – Sept 2019	**Research Internship Preferred Networks, Inc** *Exploring topics at the intersection of **natural language**, **robotics** and **reinforcement learning**. NeurIPS 2019 workshop paper (spotlight) on Emergent Communication.*
Dec 2018 – Dec 2019	**Machine Learning Researcher For.ai** *Working on researching **model based reinforcement learning**. Core contributor of a modularised reinforcement learning codebase written in Tensorflow (65 stars on Github). Submitted an open source RL library in Tensorflow to JOSS 2019.*
Oct 2018 - June 2019	**Research Engineer Mediagamma** *Research, writing papers and presenting results. Developed novel algorithms to tackle **natural language** mapping tasks using **word embeddings** from universal sentence encoder, by Cer, Daniel, et al. (2018).*
July 2014 - July 2015	**Data Analyst Burberry Group Plc** *ML projects using Sklearn, presenting results to stakeholders.*

A.2. Education

2019 – 2022 **Machine Learning PhD**
Technische Universität Darmstadt
Supervised by Prof Jan Peters (TUD) and Dr Haitham Ammar (UCL). Focusing on Black-box optimization, applications of black-box optimization such as drug design and reinforcement learning.

2017 – 2018 **Machine Learning & Data Science (MSc)**
University College London
Supervised by Prof Riedel. Completing research internship for dissertation at UCL NLP.

2013 – 2017 **Mathematical Sciences (BSc)**
University of Bath
Supervised by Prof Mike Tipping. General report of Neural networks applied to EEG data.

A.3. Reviewer

ICML 2022 – Present

TMLR 2022 – Present

A.4. Awards

2021 **Huawei**
London Research Center Timely Award

2021 **Huawei**
London Research Center Best Team

2021 **Huawei**
Innovation Pioneer Award (5 researchers receive this annually out of all R&D employees (23,000)

2021 **G-Research**
G-Research PhD Grant

2020 **CoRL**
Best Systems Paper

2020 **NeurIPS**
1st Place Black-box Optimisation Challenge (BBO) Award.

2019 **IJCAI**
Student Travel Award

2018 **Federated Logic Conference 2018**
Student Travel Award

A.5. Open Source Code

High-Dimensional Bayesian Optimisation with Variational Auto-encoders and Deep Metric Learning

Developed a SOTA High Dimensional Bayesian Optimisation algorithm for tackling combinatorial problems such as **molecule design**. Graph Neural Networks & Gaussian Process's. Code available with 261 ☆.

SMARTS: An Open-Source Scalable Multi-Agent RL Training School for Autonomous Driving, Conference on Robot Learning

Worked on developing a scalable multi-agent autonomous driving simulator. Focused on implementing constrained deep reinforcement learning agents. Code available with 592 ☆.

An Empirical Study of Assumptions in Bayesian Optimisation

Developed a SOTA [1] Bayesian Optimisation algorithm. Gaussian Process's. Code available with 261 ☆.

Reinforcement Learning Codebase

Core developer for an open source modular codebase for reinforcement learning models training, testing and visualisation. Code available with 87 ☆.

Are we Forgetting about Compositional Optimisers in Bayesian Optimisation?

Developed a new family of acquisition functions. Implemented compositional optimisation methods and compositional acquisition functions. Bayesian Optimisation method. Code available with 261 ☆.

Almost Surely Safe RL Using State Augmentation - Saute RL

Developed a new family of acquisition functions. Implemented compositional optimisation methods and compositional acquisition functions. Bayesian Optimisation method. Code available with 261 ☆.

A.6. Software skills

Tensorflow, Python, PyTorch, Bash, PyMOL

[1] https://bbochallenge.com/leaderboard/

B. Research Articles

B.1. Accepted Journal Articles

- Cowen-Rivers AI, Grosnit A, Tutunov R, Griffiths RR, Wang J, Bou-Ammar H. Are we forgetting about compositional optimisers in bayesian optimisation?. Journal Machine Learning Research. 2021 Jan 1;22:160-.

- Cowen-Rivers AI, Lyu W, Tutunov R, Wang Z, Grosnit A, Griffiths RR, Maraval AM, Jianye H, Wang J, Peters J, Bou-Ammar H. HEBO: Pushing The Limits of Sample-Efficient Hyper-parameter Optimisation. Journal of Artificial Intelligence Research. 2022 Jul 11;74:1269-349.

- Cowen-Rivers AI, Palenicek D, Moens V, Abdullah MA, Sootla A, Wang J, Bou-Ammar H. Samba: Safe model-based & active reinforcement learning. Machine Learning. 2022 Jan;111(1):173-203.

B.2. Accepted Conference Articles

- Sootla A, Cowen-Rivers AI, Wang J, Bou-Ammar H. Effects of Safety State Augmentation on Safe Exploration. Advances in neural information processing systems. 2022.

- Khan A, Cowen-Rivers AI, Deik DG, Grosnit A, Dreczkowski K, Robert PA, Greiff V, Tutunov R, Bou-Ammar D, Wang J, Bou-Ammar H. AntBO: Trust-region Bayesian Optimisation with developabilty constraints enables sample efficient, high-affinity antibody design. International Conference Machine Learning, Computational Biology Workshop. 2022.

- Sootla A, Cowen-Rivers AI, Jafferjee T, Wang Z, Mguni D, Wang J, Bou-Ammar H. SAUTE RL: Almost Surely Safe Reinforcement Learning Using State Augmentation. International Conference Machine Learning. 2022.

- Zhou M, Luo J, Villella J, Yang Y, Rusu D, Miao J, Zhang W, Alban M, FADAKAR I, Chen Z, Huang C. SMARTS: An Open-Source Scalable Multi-Agent RL Training School for Autonomous Driving. Conference on Robot Learning 2021 4: 264-285.

B.3. Under review

- Alexander I. Cowen-Rivers et al. Toward Antibody Design with Combinatorial Optimisation, Cell Reports
- Alexander I. Cowen-Rivers et al. Structured Q-Learning For Antibody Design, Machine Learning Journal
- Antoine Grosnit, Rasul Tutunov, Alexandre Max Maraval, Ryan-Rhys Griffiths, Alexander I. Cowen-Rivers et al. High-Dimensional Bayesian Optimisation, Journal of Artificial Intelligence Research