Practical and Theoretical Advances in Bayesian Optimization



Ziyu Wang

Wolfson College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas Term, 2016

This thesis is dedicated to
my wife Lin
for her unfailing love and support

Abstract

Bayesian optimization forms a set of powerful tools that allows efficient black-box optimization and has general applications in a large variety of fields. In this work we seek to advance Bayesian optimization both in the theoretical and the practical fronts as well as apply Bayesian optimization to novel and difficult problems in order to advance the state of the art.

Chapter 1 gives a broad overview of Bayesian optimization. We start by covering the published applications of Bayesian optimization. The chapter then proceeds to introduce the essential ingredients of Bayesian optimization in depth. After going through some practical considerations, the theory and history of Bayesian optimization, we end the chapter with a discussion on the latest extensions and open problems.

In Chapters 2-4, we solve three outstanding problems in the Bayesian optimization literature. Traditional Bayesian optimization approaches need to solve an auxiliary non-convex global optimization problem in the inner loop. The difficulties in solving this auxiliary optimization problem not only break the assumptions of most theoretical works in this area but also lead to computationally inefficient solutions. In Chapter 2, we propose the first algorithm in Bayesian optimization that does not need to solve auxiliary optimization problems and prove its convergence.

In Bayesian optimization, it is often important to tune the hyper-parameters of the underlying Gaussian processes. There did not exist theoretical results that allow noisy observations and at the same time varying hyper-parameters. Chapter 3, proves the first such result.

Bayesian optimization is very effective when the dimensionality of the problem is low. Scaling Bayesian optimization to high dimensionality, however, has been a long standing open problem of the field. In Chapter 4, we develop an algorithm that extends Bayesian optimization to very high dimensionalities where the underlying problems have low intrinsic dimensionality. We also prove theoretical guarantees of the proposed algorithm.

In Chapter 5, we turn our attention to improving an essential component of Bayesian optimization: acquisition functions. Acquisition functions form a critical component of Bayesian optimization and yet there does not exist an optimal acquisition function that is easily computable. Instead of relying on one acquisition function, we develop a new information-theoretic portfolio of acquisition functions. We show empirically that our approach is more effective than any single acquisition function in the portfolio.

Last but not least, in Chapter 6 we adapt Bayesian optimization to derive an adaptive Hamiltonian Monte Carlo sampler. Hamiltonian Monte Carlo is one of the most effective MCMC algorithms. It is, however, notoriously difficult to tune. In this chapter, we follow the approach of adapting Markov chains in order to improve their convergence where our adaptive strategy is based on Bayesian optimization. We provide theoretical analysis as well as a comprehensive set of experiments demonstrating the effectiveness of our proposed algorithm.

Preface

This work was entirely done under the supervision of Professor Nando de Freitas without whose help and guidance this work would not have been possible.

The work originated with the application of Bayesian optimization to adapt HMC samplers. This became the foundation of Chapter 6. Shakir and Nando helped with running the experiments and improving the writeup of the paper. Much later in my studies, we applied this idea of adaptive HMC samplers to Bayesian analysis of continuous time Markov chains and succeeded in boosting dramatically improving sample efficiency.

After discovering Bayesian optimization, I was fascinated by its data-efficiency and maybe more so by its long standing open problems. The next few projects I worked on were aimed at solving these open problems.

Chapter 2 was the result of our attempt in avoiding the acquisition function optimization which presented both theoretical and practical challenges. This is joint work with Babak Shakibi, and Lin Jin. Babak helped setting up the term extraction example and Lin helped in running some of the earlier experiments.

In Chapter 3, we aimed at improving the theoretical understanding of Bayesian optimization in its most common setting as such theoretical work did not exist prior to our work.

In Chapter 4, we attacked probably the most difficult open question in Bayesian optimization of all: high-dimensionality. Through fruitful discussions with Frank Hutter and Masrour Zoghi, we managed to scale Bayesian optimization to very high dimensions by exploiting an abundant class of functions that are common in hyper-parameter optimization problems. David Matheson provided the code for the random forest example and Masrour as well as Frank helped with the regret analysis. The paper that is based off this chapter won the IJCAI Distinguished Paper Award in 2013. A follow up of this work is published at the Journal of Artificial Intelligence Research.

Instead of attacking an open problem of Bayesian optimization, Chapter 5 turned to improving its practical performance. Together with Bobak Shahriari, we conceived the idea of and wrote the code for an information-theoretic portfolio of acquisition functions

which not only matched the highest achieving acquisition function on any of the problems tested but also often surpassed all in performance.

Later in my studies, I turned my attention to sequential decision making problems and specifically reinforcement learning. Bayesian optimization and reinforcement learning, although different in appearance, are both sequential decision problems and therefore share many similarities. Together with my co-authors, we dramatically improved the performance of Deep Q-networks and achieved state of the art results on 57 Atari games. The resulting paper won the Best Paper Award in ICML in 2016. More recently, we introduced the first stable off-policy actor critic algorithm and matched the state of the art results on Atari as well as control environments while improving data efficiency drastically. Neither work in the field of reinforcement learning, however, are presented as part of this thesis to maintain a consistent theme.

Please see the following selected list of publications submitted in the duration of my studies.

Selected Publications:

- Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *ICLR*, 2017
- T. Zhao, Z. Wang, A. Cumberworth, J. Gsponer, N. de Freitas, and A. Bouchard-Côté. Bayesian analysis of continuous time Markov chains with application to phylogenetic modelling. *Bayesian Analysis*, 2016
- Z. Wang, S. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016
- B. Shahriari, K. Swersky, Z. Wang, R. P Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 2016
- Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 2016
- Z. Wang and N. de Freitas. Bayesian multiscale optimistic optimization. In *AIStats*, 2014
- Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. de Freitas. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, pages 1778–1784, 2013

- Z. Wang, S. Mohamed, and N. de Freitas. Adaptive Hamiltonian and Riemann manifold Monte Carlo samplers. In *ICML*, 2013

Contents

Abstract ii

Preface iv

Table of Contents vii

List of Figures xi

Acknowlegements xiii

1 Introduction 1

1.1 Applications of Bayesian optimization . . . 4

1.1.1 A/B testing . . . 4

1.1.2 Recommender systems . . . 6

1.1.3 Robotics and Reinforcement learning . . . 6

1.1.4 Environmental monitoring and sensor networks . . . 6

1.1.5 Preference learning and interactive interfaces . . . 7

1.1.6 Automatic machine learning and hyper-parameter tuning . . . 7

1.1.7 Combinatorial optimization . . . 7

1.1.8 Natural language processing and text . . . 8

1.2 Chapter Overview . . . 8

1.3 Bayesian optimization with parametric models . . . 8

1.3.1 Thompson sampling in the Beta-Bernoulli bandit model . . . 9

1.3.2 Linear models . . . 11

1.3.3 Generalized linear models . . . 14

1.3.4 Related literature . . . 15

1.3.4.1 Active learning and experimental design . . . 15

1.3.4.2 Multi-armed bandit . . . 16

1.4 Non-parametric models . . . 17

1.4.1 The Gaussian process . . . 18
1.4.2 Common kernels . . . 19
1.4.3 Prior mean functions . . . 20
1.4.4 Marginal likelihood . . . 20
1.4.5 Computational costs and other regression models . . . 21
1.4.5.1 Sparse pseudo-input Gaussian processes (SPGP) . . . 21
1.4.5.2 Sparse spectrum Gaussian processes (SSGP) . . . 23
1.4.5.3 Random forests . . . 23
1.5 Acquisition functions . . . 25
1.5.1 Improvement-based policies . . . 25
1.5.2 Optimistic policies . . . 26
1.5.3 Information-based policies . . . 27
1.5.4 Portfolios of acquisition functions . . . 30
1.6 Practical considerations . . . 30
1.6.1 Handling hyper-parameters . . . 31
1.6.2 Optimizing acquisition functions . . . 32
1.6.3 Conditional Spaces . . . 33
1.6.4 Non-stationarity . . . 34
1.6.5 Parallelization . . . 36
1.7 Theory of Bayesian optimization . . . 37
1.8 History of Bayesian optimization and related approaches . . . 38
1.9 Extensions and Open Questions . . . 40
1.9.1 Constrained Bayesian optimization . . . 40
1.9.2 Cost-sensitivity . . . 41
1.9.3 High-dimensional problems . . . 41
1.9.4 Multi-Task . . . 42
1.9.5 Freeze-Thaw . . . 44
1.10 Outline of this work and contributions . . . 45

2 Bayesian Multi-scale Optimistic optimization 47
2.1 Bayesian optimization with GP confidence bounds . . . 49
2.1.1 Shrinking feasible regions . . . 49
2.2 Simultaneous optimistic optimization . . . 50
2.3 BaMSOO . . . 52
2.4 Analysis . . . 53
2.5 Experiments with global optimization benchmarks . . . 56

2.6 Application to term extraction . . . 59
2.7 Discussion . . . 59

3 Theoretical Analysis of Bayesian Optimization with Unknown Gaussian Process Hyper-Parameters 62
3.1 Preliminaries . . . 63
3.2 Theoretical analysis . . . 64
3.2.1 Background: Regret . . . 64
3.2.2 Background: Sub-Gaussian noise . . . 65
3.2.3 Background: Information gain . . . 65
3.2.4 Background: Reproducing kernel Hilbert spaces . . . 65
3.2.5 Main result . . . 67
3.3 An algorithm inspired by the theory . . . 69
3.4 Conclusion . . . 72

4 Bayesian Optimization in a Billion Dimensions via Random Embeddings 73
4.1 Random Embedding for Bayesian Optimization . . . 75
4.1.1 Increasing the Success Rate of REMBO . . . 80
4.1.2 Choice of Kernel . . . 81
4.1.3 Regret Bounds . . . 82
4.1.4 Hyper-parameter Optimization . . . 85
4.2 Experiments . . . 86
4.2.1 Experimental Setup . . . 87
4.2.2 Bayesian Optimization in a Billion Dimensions . . . 87
4.2.3 Synthetic Discrete Experiment . . . 89
4.2.4 Automatic Configuration of a Mixed Integer Linear Programming Solver . . . 90
4.2.5 Automatic Configuration of Random Forest Kinect Body Part Classifier . . . 92
4.3 Conclusion . . . 95

5 An Entropy Search Portfolio for Bayesian Optimization 96
5.1 Entropy search over portfolios . . . 97
5.1.1 Entropy as a meta-criterion . . . 100
5.1.2 Sampling posterior global minimizers . . . 102
5.1.3 Algorithm summary . . . 103
5.2 Experiments . . . 104

5.2.1 Model details and hyper-parameter marginalization . . . 104
5.2.2 Global optimization test functions . . . 105
5.2.3 Geostatistics datasets . . . 107
5.2.4 Control tasks . . . 108
5.3 Conclusion . . . 108

6 Adaptive Hamiltonian Monte Carlo 110
6.1 Hamiltonian-based Monte Carlo Sampling . . . 111
6.2 Adaptive Hamiltonian Monte Carlo . . . 113
6.3 Analysis of Convergence . . . 117
6.4 Results . . . 118
6.4.1 Bayesian Logistic Regression . . . 119
6.4.2 Log-Gaussian Cox Point Process . . . 120
6.4.3 Stochastic Volatility . . . 122
6.5 Results on Bayesian Analysis of Continuous Time Markov Chains . . . 123
6.6 Conclusion . . . 126

7 Conclusion 127
7.1 Contributions and Results . . . 127
7.2 Future Work . . . 129
7.2.1 Improvement to Bayesian Optimization and its Applications . . . 129
7.2.2 Statistical Modeling in Reinforcement Learning . . . 130

Bibliography 132

Appendix A Proofs for BaMSOO 152
A.1 Technical Lemmas . . . 152
A.2 Main Results . . . 157
A.2.1 Simple Regret . . . 157

Appendix B Proofs for Hyper Parameter Optimization 159
B.1 Concentration . . . 159
B.2 Supporting lemmas . . . 162
B.3 Properties of the expected improvement acquisition function . . . 163
B.4 Proof of main result . . . 165

Appendix C Regret Bound for REMBO 168

List of Figures

1.1 Illustration of the Bayesian optimization procedure. 5
1.2 Example of the Beta-Bernoulli model for A/B testing. 11
1.3 Comparison between different kernels. 20
1.4 Comparison of surrogate regression models. 22
1.5 Visualization of the surrogate regression model and various acquisition functions. 28
1.6 Illustrations of input warping. 35
1.7 Visualization of multi-output GPs. 43

2.1 The global shrinking method of [55]. 50
2.2 Comparing SOO and BaMSOO. 52
2.3 Comparison of GP-UCB, SOO, and BaMSOO on multi-modal test functions of low dimensionality. 57
2.4 Comparison of GP-UCB, SOO, and BaMSOO on multi-modal test functions of moderate dimensionality. 58
2.5 Comparison of GP-UCB, SOO, and BaMSOO on optimizing an NLP problem. 60

3.1 Comparing slice sampling for estimating hyper-parameters with our proposed algorithm. 71

4.1 Illustration of effective dimensionality. 74
4.2 Core idea of REMBO. 77
4.3 Comparing Random search with HD BO and REMBO. 88
4.4 Comparing random search, BO and REMBO on 25-dimensional problem. . 90
4.5 Performance of various methods for configuration of `lpsolve`. 91
4.6 Illustration of the random forest prediction problem. 93
4.7 Performance of various methods for optimizing RF parameters for body part classification. 94

5.1 Visualization of Bayesian optimization with acquisition function portfolios. 98
5.2 Histogram of 1000 approximate samples from pmin. 99
5.3 Visualization of the Entropy Search Portfolio. 101
5.4 Absolute error of the best observation for the Branin and Hartmann 3 synthetic functions. 105
5.5 Best observed evaluation on mining datasets Brenda and Agromet. 107
5.6 Best observed evaluation on the Walker and Repeller control tasks. 108

6.1 1000 samples from a bivariate Gaussian distribution generated using HMC. 113
6.2 Box plots comparing ESS/L for Bayesian logistic regression. 120
6.3 Comparing minimum (red), median (blue) and maximum (black) ESS/L for the Log-Gaussian Cox model. 121
6.4 Comparing quality of posterior distributions from samples obtained using AHMC and NUTS for the log-Gaussian Cox model. 122
6.5 Effective sample size per 10^4 seconds in the log scale using AHMC, an adaptive NMH with a joint proposal guided by an estimated covariance matrix and NMH with bandwidths of 0.01, 0.02, 0.05, 0.10, where NMH x stands for a Normal proposal Metropolis Hastings algorithms with proposal bandwidth x. 124
6.6 ESS per second (displayed in log scale) across all features obtained by running HMC with tuning parameters coming from a grid centered at the values ϵ and L coming from AHMC. 126

C.1 A illustration of the different kernels defined in the regret proof for REMBO. 170

Acknowledgment

I would first and foremost like to thank my supervisor, Professor Nando de Freitas, for his knowledge, guidance and support. Nando's enthusiasm and his extremely broad interests opened my eyes to the field of machine learning.

I would also like to thank all my collaborators without whom this work would not have be possible. The same goes to my lab mates who made this journey much more enjoyable.

I would like to thank my parents for their love and support while I have been work towards this. Last but not least, I would like to thank my wife Lin as well as my sons Ben and Yi for their unfailing love and understanding.

Chapter 1

Introduction

Design problems are pervasive in scientific and industrial endeavours: scientists design experiments to gain insights into physical and social phenomena, engineers design machines to execute tasks more efficiently, pharmaceutical researchers design new drugs to fight disease, companies design websites to enhance user experience and increase advertising revenue, geologists design exploration strategies to harness natural resources, environmentalists design sensor networks to monitor ecological systems, and developers design software to drive computers and electronic devices. All these design problems are fraught with choices, choices that are often complex and high-dimensional, with interactions that make them difficult for individuals to reason about.

For example, many organizations routinely use the popular mixed integer programming solver IBM ILOG CPLEX[1] for scheduling and planning. This solver has 76 free parameters, which the designers must tune manually – an overwhelming number to deal with by hand. This search space is too vast for anyone to effectively navigate.

More generally, consider teams in large companies that develop software libraries for other teams to use. These libraries have hundreds or thousands of free choices and parameters that interact in complex ways. In fact, the level of complexity is often so high that it becomes impossible to find domain experts capable of tuning these libraries to generate a new product.

As a second example, consider massive online games involving the following three parties: content providers, users, and the analytics company that sits between them. The analytics company must develop procedures to automatically design game variants across millions of users; the objective is to enhance user experience and maximize the content provider's revenue.

[1] http://www.ibm.com/software/commerce/optimization/cplex-optimizer/

The preceding examples highlight the importance of automating design choices. For a nurse scheduling application, we would like to have a tool that automatically chooses the 76 CPLEX parameters so as to improve healthcare delivery. When launching a mobile game, we would like to use the data gathered from millions of users in real-time to automatically adjust and improve the game. When a data scientist uses a machine learning library to forecast energy demand, we would like to automate the process of choosing the best forecasting technique and its associated parameters.

Any significant advances in automated design can result in immediate product improvements and innovation in a wide area of domains, including advertising, health-care informatics, banking, information mining, life sciences, control engineering, computing systems, manufacturing, e-commerce, and entertainment.

Bayesian optimization has emerged as a powerful solution for these varied design problems. In academia, it is impacting a wide range of areas, including interactive user-interfaces [33], robotics [139, 148], environmental monitoring [144], information extraction [222], combinatorial optimization [106, 226], automatic machine learning [20, 202, 208, 211, 98], sensor networks [72, 205], adaptive Monte Carlo [143], experimental design [14] and reinforcement learning [34].

When software engineers develop programs, they are often faced with myriad choices. By making these choices explicit, Bayesian optimization can be used to construct optimal programs [99]: that is to say, programs that run faster or compute better solutions. Furthermore, since different components of software are typically integrated to build larger systems, this framework offers the opportunity to automate integrated products consisting of many parametrized software modules.

Mathematically, we are considering the problem of finding a global maximizer (or minimizer) of an unknown objective function f:

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{1.1}$$

where $\mathcal{X}$ is some design space of interest; in global optimization, $\mathcal{X}$ is often a compact subset of $\mathbb{R}^d$ but the Bayesian optimization framework can be applied to more unusual search spaces that involve categorical or conditional inputs, or even combinatorial search spaces with multiple categorical inputs. Furthermore, we will assume the *black-box* function f has no simple closed form, but can be evaluated at any arbitrary query point $\mathbf{x}$ in the domain. This evaluation produces noise-corrupted (stochastic) outputs $y \in \mathbb{R}$ such that $\mathbb{E}[y|f(\mathbf{x})] = f(\mathbf{x})$. In other words, we can only observe the function f through unbiased noisy point-wise observations y. Although this is the minimum requirement for

Bayesian optimization, when gradients are available, they can be incorporated in the algorithm as well; see for example Sections 4.2.1 and 5.2.4 of [137]. In this setting, we consider a sequential search algorithm which, at iteration t, selects a location $\mathbf{x}_{t+1}$ at which to query f and observe y_{t+1}. After T queries, the algorithm (optionally) makes a final recommendation $\bar{\mathbf{x}}_T$, which represents the algorithm's best estimate of the optimizer.

In the context of big data applications for instance, the function f can be an object recognition system (e.g., deep neural network) with tunable parameters $\mathbf{x}$ (e.g., architectural choices, learning rates, etc) with a stochastic observable classification accuracy $y = f(\mathbf{x})$ on a particular dataset such as ImageNet. Because the Bayesian optimization framework is very data efficient, it is particularly useful in situations like these where evaluations of f are costly, where one does not have access to derivatives with respect to $\mathbf{x}$, and where f is non-convex and multimodal. In these situations, Bayesian optimization is able to take advantage of the full information provided by the history of the optimization to make this search efficient.

Fundamentally, Bayesian optimization is a sequential model-based approach to solving problem (1.1). In particular, we prescribe a prior belief over the possible objective functions and then sequentially refine this model as data are observed via Bayesian posterior updating. The Bayesian posterior represents our updated beliefs – given data – on the likely objective function we are optimizing. Equipped with this probabilistic model, we can sequentially induce acquisition functions $\alpha_t : \mathcal{X} \mapsto \mathbb{R}$ that leverage the uncertainty in the posterior to guide exploration. Intuitively, the acquisition function evaluates the utility of candidate points for the next evaluation of f; therefore $\mathbf{x}_{t+1}$ is selected by maximizing α_t, where the index t indicates the implicit dependence on the currently available data. Here the "data" refers to previous locations where f has been evaluated, and the corresponding noisy outputs.

In summary, the Bayesian optimization framework has two key ingredients. The first ingredient is a (probabilistic) surrogate model, which constitutes a prior that captures our beliefs about the behavior of the unknown objective function and an observation model that describes the data generation mechanism. The second ingredient is a loss function that describes how optimal a sequence of queries are; in practice, these loss functions often take the form of regret, either simple or cumulative. Ideally, the expected loss is then minimized to select an optimal sequence of queries. After observing the output of each query of the objective, the prior is updated to produce a more informative posterior distribution over the space of objective functions; see Figure 1.1 and Algorithm 1 for an illustration and pseudo-code of this framework. See Section 4 of [84] for another introduction.

Algorithm 1 Bayesian optimization

1: **for** $n = 1, 2, \ldots$ **do**
2: select new $\mathbf{x}_{t+1}$ by optimizing acquisition function α

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x}; \mathcal{D}_t)$$

3: query objective function to obtain y_{t+1}
4: augment data $\mathcal{D}_{t+1} = \{\mathcal{D}_t, (\mathbf{x}_{t+1}, y_{t+1})\}$
5: update statistical model
6: **end for**

One problem with this minimum expected risk framework is that the true sequential risk, up to the full evaluation budget, is typically computationally intractable. This has led to the introduction of many myopic heuristics known as acquisition functions, e.g., Thompson sampling, probability of improvement, expected improvement, upper-confidence-bounds, and entropy search. These acquisition functions trade off exploration and exploitation; their optima are located where the uncertainty in the surrogate model is large (exploration) and/or where the model prediction is high (exploitation). Bayesian optimization algorithms then select the next query point by maximizing such acquisition functions. Naturally, these acquisition functions are often even more multi-modal and difficult to optimize, in terms of querying efficiency, than the original black-box function f. Therefore it is critical that the acquisition functions be cheap to evaluate or approximate: cheap in relation to the expense of evaluating the black-box f. Since acquisition functions have analytical forms that are easy to evaluate or at least approximate, it is usually much easier to optimize them than the original objective function.

1.1 Applications of Bayesian optimization

Before embarking on a detailed introduction to Bayesian optimization, the following sections provide an overview of the many and varied successful applications of Bayesian optimization that should be of interest to data scientists.

1.1.1 A/B testing

Though the idea of A/B testing dates back to the early days of advertising in the form of so-called focus groups, the advent of the internet and smart phones has given web and app developers a new forum for implementing these tests at unprecedented scales. By redirecting small fractions of user traffic to experimental designs of an ad, app, game,

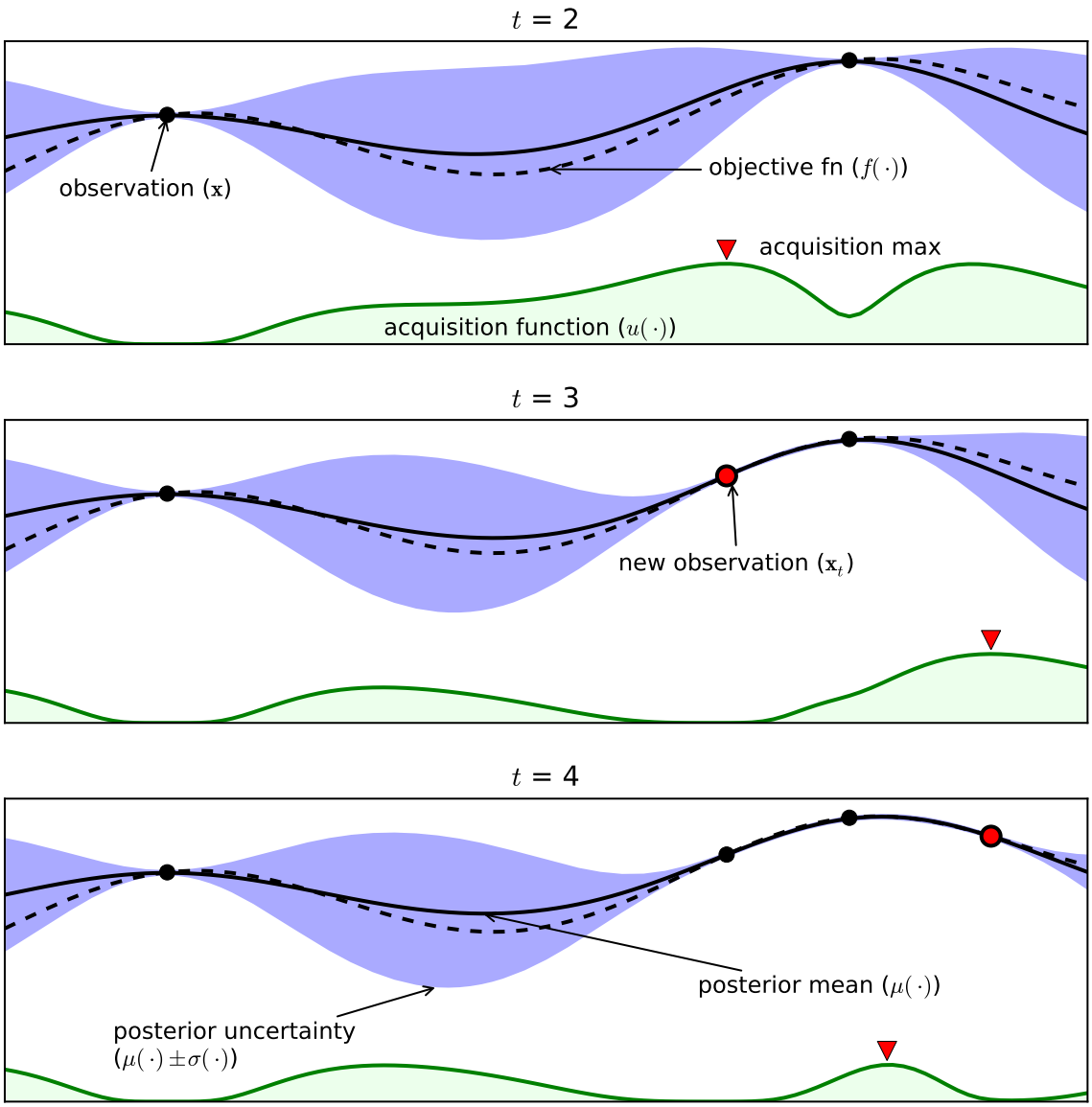


Figure 1.1: Illustration of the Bayesian optimization procedure over three iterations. The plots show the mean and confidence intervals estimated with a probabilistic model of the objective function. Although the objective function is shown, in practice it is unknown. The plots also show the acquisition functions in the lower shaded plots. The acquisition is high where the model predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration). Note that the area on the far left remains unsampled, as while it has high uncertainty, it is correctly predicted to offer little improvement over the highest observation [34].

or website, the developers can utilize noisy feedback to optimize any observable metric with respect to the product's configuration. In fact, depending on the particular phase of a product's life, new subscriptions may be more valuable than revenue or user retention,

or vice versa; the click-through rate might be the relevant objective to optimize for an ad, whereas for a game it may be some measure of user engagement.

The crucial problem is how to optimally query these subsets of users in order to find the best product with high probability within a predetermined query budget, or how to redirect traffic sequentially in order to optimize a cumulative metric while incurring the least opportunity cost [122, 193, 46].

1.1.2 Recommender systems

In a similar setting, on-line content providers make product recommendations to their subscribers in order to optimize either revenue in the case of e-commerce sites, readership for news sites, or consumption for video and music streaming websites. In contrast to A/B testing, the content provider can make multiple suggestions to any given subscriber. The techniques reviewed in this work have been successfully used for the recommendation of news articles [133, 46, 215].

1.1.3 Robotics and Reinforcement learning

Bayesian optimization has also been successfully applied to policy search. For example, by parameterizing a robot's gait it is possible to optimize it for velocity or smoothness as was done on the Sony AIBO ERS-7 in [139]. Similar policy parameterization and search techniques have been used to navigate a robot through landmarks, minimizing uncertainty about its own location and map estimate [148, 146]. See [34] for an example of applying Bayesian optimization to hierarchical reinforcement learning, where the technique is used to automatically tune the parameters of a neural network policy and to learn value functions at higher levels of the hierarchy. Bayesian optimization has also been applied to learn attention policies in image tracking with deep networks [57]

1.1.4 Environmental monitoring and sensor networks

Sensor networks are used to monitor environmentally relevant quantities: temperature, concentration of pollutants in the atmosphere, soil, oceans, etc. Whether inside a building or at a planetary scale, these networks make noisy local measurements that are interpolated to produce a global model of the quantity of interest. In some cases, these sensors are expensive to activate but one can answer important questions like what is the hottest or coldest spot in a building by activating a relatively small number of sensors. Bayesian optimization was used for this task and the similar one of finding the location of greatest highway traffic congestion [205]. Also, see [72] for a meteorological application.

When the sensor is mobile, there is a cost associated with making a measurement which relates to the distance traveled by a vehicle on which the sensor is mounted (e.g., a drone). This cost can be incorporated in the decision making process as in [144].

1.1.5 Preference learning and interactive interfaces

The computer graphics and animation fields are filled with applications that require the setting of tricky parameters. In many cases, the models are complex and the parameters unintuitive for non-experts. In [35, 33], the authors use Bayesian optimization to set the parameters of several animation systems by showing the user examples of different parametrized animations and asking for feedback. This *interactive Bayesian optimization* strategy is particularly effective as humans can be very good at comparing examples, but unable to produce an objective function whose optimum is the example of interest.

1.1.6 Automatic machine learning and hyper-parameter tuning

In this application, the goal is to automatically select the best model (e.g., random forests, support vector machines, neural networks, etc.) and its associated hyper-parameters for solving a task on a given dataset. For big datasets or when considering many alternatives, cross-validation is very expensive and hence it is important to find the best technique within a fixed budget of cross-validation tests. The objective function here is the generalization performance of the models and hyper-parameter settings; a noisy evaluation of the objective corresponds to training a single model on all but one cross-validation folds and returning, e.g., the empirical error on the held out fold.

The traditional alternatives to cross-validation include racing algorithms that use conservative concentration bounds to rule out under-performing models [145, 153]. Recently, the Bayesian optimization approach for the model selection and tuning task has received much attention in tuning deep belief networks [20], Markov chain Monte Carlo methods [143, 85], convolutional neural networks [202, 208], and automatically selecting among WEKA and `scikit-learn` offerings [211, 98].

1.1.7 Combinatorial optimization

Bayesian optimization has been used to solve difficult combinatorial optimization problems in several applications. One notable approach is called empirical hardness models (EHMs) that use a set of problem features to predict the performance of an algorithm on a specific problem instance [132]. Bayesian optimization with an EHM amounts to finding the best algorithm and configuration for a given problem. This concept has been applied

to e.g., tuning mixed integer solvers [105, 226], and tuning approximate nearest neighbor algorithms [147]. Bayesian optimization has also been applied to fast object localization in images [231].

1.1.8 Natural language processing and text

Bayesian optimization has been applied to machine translation in [152] and to tune text representations for more general text and language tasks in [230].

1.2 Chapter Overview

In the rest of the chapter, we introduce the ingredients of Bayesian optimization in depth. We aim to provide a broad introduction of Bayesian optimization that disentangles the multiple components that determine the success of Bayesian optimization implementations. Specifically, we would discuss but not limit ourselves to the Gaussian process based Bayesian optimization which is the focus of this work.

We begin in Sections 1.3 and 1.4, with an introduction to parametric and non-parametric models, respectively, for binary- and real-valued objective functions. In Section 1.5, we will introduce many acquisition functions, compare them, and even combine them into portfolios. Several practical and implementation details, including available software packages, are discussed in Section 1.6. A survey of theoretical results and a brief history of model-based optimization are provided in Sections 1.7 and 1.8, respectively. Finally, we introduce more recent developments in Section 1.9.

In the last section of this chapter, we outline the remainder chapters of our work and contributions.

1.3 Bayesian optimization with parametric models

The central idea of Bayesian optimization is to build a model that can be updated and queried to drive optimization decisions. In this section, we cover several such models, but for the sake of clarity, we first consider a generic family of models parameterized by $\mathbf{w}$. Let $\mathcal{D}$ denote the available data. We will generalize to the non-parametric situation in the proceeding section.

Since $\mathbf{w}$ is an unobserved quantity, we treat it as a latent random variable with a *prior* distribution $p(\mathbf{w})$, which captures our *a priori* beliefs about probable values for $\mathbf{w}$ before

any data is observed. Given data $\mathcal{D}$ and a *likelihood* model $p(\mathcal{D}|\mathbf{w})$, we can then infer a *posterior* distribution $p(\mathbf{w}|\mathcal{D})$ using Bayes' rule:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})\, p(\mathbf{w})}{p(\mathcal{D})}\,. \tag{1.2}$$

This posterior represents our updated beliefs about $\mathbf{w}$ after observing data $\mathcal{D}$. The denominator $p(\mathcal{D})$ is the marginal likelihood, or evidence, and is usually computationally intractable. Fortunately, it does not depend on $\mathbf{w}$ and is therefore simply a normalizing constant. A typical modeling choice is to use conjugacy to match the prior and likelihood so that the posterior (and often the normalizing constant) can be computed analytically.

1.3.1 Thompson sampling in the Beta-Bernoulli bandit model

We begin our discussion with a treatment of perhaps the simplest statistical model, the Beta-Bernoulli. Imagine that there are K drugs that have unknown effectiveness, where we define "effectiveness" as the probability of a successful cure. We wish to cure patients, but we must also identify which drugs are effective. Such a problem is often called a Bernoulli (or binomial) *bandit* problem by analogy to a group of slot machines, which each yield a prize with some unknown probability. In addition to clinical drug settings, this formalism is useful for A/B testing [193], advertising, and recommender systems [133, 46], among a wide variety of applications. The objective is to identify which *arm* of the bandit to pull, e.g., which drug to administer, which movie to recommend, or which advertisement to display. Initially, we consider the simple case where the arms are independent insofar as observing the success or failure of one provides no information about another.

Returning to the drug application, we can imagine the effectiveness of different drugs (arms on the bandit) as being determined by a function f that takes an index $a \in 1, \ldots, K$ and returns a Bernoulli parameter in the interval $(0, 1)$. With $y_i \in \{0, 1\}$, we denote the Bernoulli outcome of the treatment of patient i, and this has mean parameter $f(a_i)$ if the drug administered was a_i. Note that we are assuming stochastic feedback, in contrast to deterministic or adversarial feedback [12, 13]. With only K arms, we can fully describe the function f with a parameter $\mathbf{w} \in (0, 1)^K$ so that $f_{\mathbf{w}}(a) := w_a$.

Over time, we will see outcomes from different patients and different drugs. We can denote these data as a set of tuples $\mathcal{D}_t = \{(a_i, y_i)\}_{i=1}^t$, where a_i indicates which of the K drugs was administered and y_i is 1 if the patient was cured and 0 otherwise. In a Bayesian setting, we will use these data to compute a posterior distribution over $\mathbf{w}$. A natural choice

for the prior distribution is a product of K beta distributions:

$$p(\mathbf{w}|\alpha, \beta) = \prod_{a=1}^{K} \text{Beta}(w_a|\alpha, \beta)\,, \tag{1.3}$$

as this is the conjugate prior to the Bernoulli likelihood, and it leads to efficient posterior updating. We denote by $t_{a,1}$ the number of patients cured by drug a and by $t_{a,0}$ the number of patients who received a but were unfortunately not cured; that is

$$t_{a,0} = \sum_{i=1}^{t} \mathbb{I}(y_i = 0, a_i = a) \tag{1.4}$$

$$t_{a,1} = \sum_{i=1}^{t} \mathbb{I}(y_i = 1, a_i = a)\,. \tag{1.5}$$

The convenient conjugate prior then leads to a posterior distribution which is also a product of betas:

$$p(\mathbf{w}|\mathcal{D}) = \prod_{a=1}^{K} \text{Beta}(w_a|\alpha + t_{a,1}, \beta + t_{a,0})\,. \tag{1.6}$$

Note that this makes it clear how the hyper-parameters $\alpha, \beta > 0$ in the prior can be interpreted as *pseudo-counts*. Figure 1.2 provides a visualization of the posterior of a three-armed Beta-Bernoulli bandit model with a Beta(2, 2) prior.

In Section 1.5, we will introduce various strategies for selecting the next arm to pull within models like the Beta-Bernoulli, but for the sake of illustration, we introduce Thompson sampling [210], the earliest and perhaps the simplest non-trivial bandit strategy. This strategy is also commonly known as randomized probability matching [193] because it selects the arm based on the posterior probability of optimality, here given by a beta distribution. In simple models like the Beta-Bernoulli, it is possible to compute this distribution in closed form, but more often it must be estimated via, e.g., Monte Carlo.

After observing t patients in our drug example, we can think of a bandit strategy as being a rule for choosing which drug to administer to patient $t + 1$, i.e., choosing a_{t+1} among the K options. In the case of Thompson sampling, this can be done by drawing a single sample $\tilde{\mathbf{w}}$ from the posterior and then maximizing the resulting surrogate $f_{\tilde{\mathbf{w}}}$, i.e.,

$$a_{n+1} = \arg\max_a f_{\tilde{\mathbf{w}}}(a) \;\text{ where } \tilde{\mathbf{w}} \sim p(\mathbf{w}|\mathcal{D}_t). \tag{1.7}$$

For the Beta-Bernoulli, this corresponds to simply drawing $\tilde{\mathbf{w}}$ from (1.6) and then choosing the action with the largest $\tilde{w}_a$. This procedure, shown in pseudo-code in Algorithm 2, is also commonly called *posterior sampling* [183]. It is popular for several reasons: 1) there are no

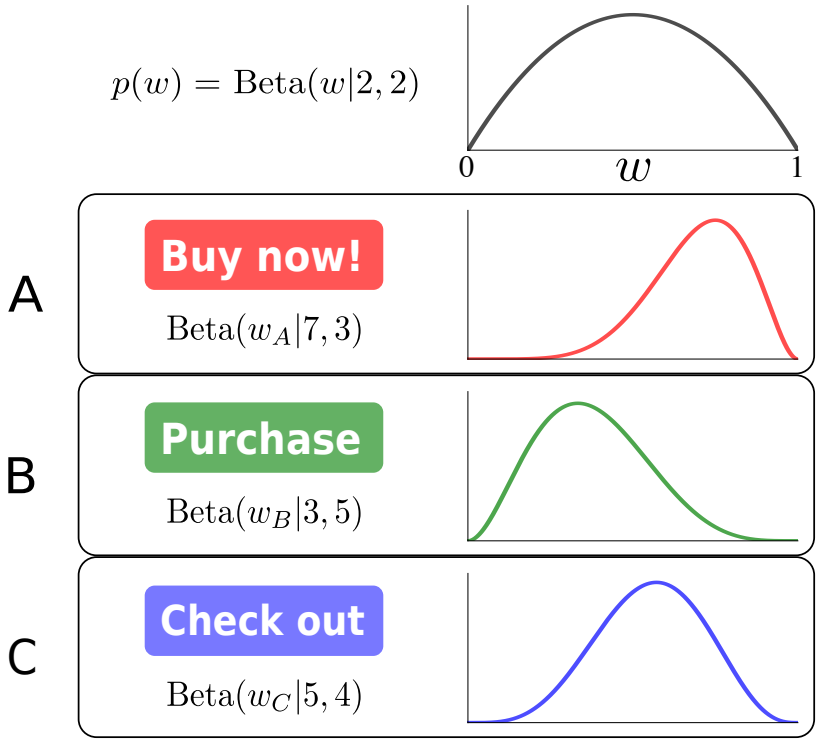


Figure 1.2: Example of the Beta-Bernoulli model for A/B testing. Three different buttons are being tested with various colours and text. Each option is given 2 successes (click-throughs) and 2 failures as a prior (top). As data are observed, each option updates its posterior over w. Option A is the current best with 5 successes and only 1 observed failure.

free parameters other than the prior hyper-parameters of the Bayesian model, 2) the strategy naturally trades off between exploration and exploitation based on its posterior beliefs on $\mathbf{w}$; arms are explored only if they are likely (under the posterior) to be optimal, 3) the strategy is relatively easy to implement as long as Monte Carlo sampling mechanisms are available for the posterior model, and 4) the randomization in Thompson sampling makes it particularly appropriate for batch or delayed feedback settings where many selections a_{n+1} are based on the identical posterior [193, 46].

1.3.2 Linear models

In many applications, the designs available to the experimenter have components that can be varied independently. For example, in designing an advertisement, one has choices such as artwork, font style, and size; if there are five choices for each, the total number of possible configurations is 125. In general, this number grows combinatorially in the number of components. This presents challenges for approaches such as the independent Beta-Bernoulli model discussed in the previous section: modelling the arms as independent will

Algorithm 2 Thompson sampling for Beta-Bernoulli bandit

Require: α, β: hyper-parameters of the beta prior

Initialize $t_{a,0} = t_{a,1} = i = 0$ for all a
repeat
  for $a = 1, \ldots, K$ **do**
    $\tilde{w}_a \sim \text{Beta}(\alpha + t_{a,1}, \beta + t_{a,0})$
  end for
  $a_i = \arg\max_a \tilde{w}_a$
  Observe y_i by pulling arm a_i
  if $y_i = 0$ **then**
    $t_{a_i,0} = t_{a_i,0} + 1$
  else
    $t_{a_i,1} = t_{a_i,1} + 1$
  end if
  $i = i + 1$
until stopping criterion reached

lead to strategies that must try every option at least once. This rapidly becomes infeasible in the large spaces of real-world problems. In this section, we discuss a parametric approach that captures dependence between the arms via a linear model. For simplicity, we first consider the case of real-valued outputs y and generalize this model to binary outputs in the succeeding section.

As before, we begin by specifying a likelihood and a prior. In the linear model, it is natural to assume that each possible arm a has an associated feature vector $\mathbf{x}_a \in \mathbb{R}^d$. We can then express the expected payout (reward) of each arm as a function of this vector, i.e., $f(a) = f(\mathbf{x}_a)$. Our objective is to learn this function $f : \mathbb{R}^d \mapsto \mathbb{R}$ for the purpose of choosing the best arm, and in the linear model we require f to be of the form $f_{\mathbf{w}}(a) = \mathbf{x}_a^T \mathbf{w}$, where the parameters $\mathbf{w}$ are now feature weights. This forms the basis of our likelihood model, in which the observations for arm a are drawn from a Gaussian distribution with mean $\mathbf{x}_a^T \mathbf{w}$ and variance σ^2.

We use $\mathbf{X}$ to denote the $n \times d$ design matrix in which row i is the feature vector associated with the arm pulled in the ith iteration, $\mathbf{x}_{a_i}$. We denote by $\mathbf{y}$ the n-vector of observations. In this case, there is also a natural conjugate prior for $\mathbf{w}$ and σ^2: the normal-

inverse-gamma, with density given by

$$\begin{aligned}\mathrm{NIG}(\mathbf{w}, \sigma^2 \mid \mathbf{w}_0, \mathbf{V}_0, \alpha_0, \beta_0) = \\ |2\pi\sigma^2\mathbf{V}_0|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{w}-\mathbf{w}_0)^T\mathbf{V}_0^{-1}(\mathbf{w}-\mathbf{w}_0)\right\} \\ \times \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)(\sigma^2)^{\alpha_0+1}} \exp\left\{-\frac{\beta_0}{\sigma^2}\right\}. \quad (1.8)\end{aligned}$$

There are four prior hyper-parameters in this case, $\mathbf{w}_0$, $\mathbf{V}_0$, α_0, and β_0. As in the Beta-Bernoulli case, this conjugate prior enables the posterior distribution to be computed easily, leading to another normal-inverse-gamma distribution, now with parameters

$$\mathbf{w}_t = \mathbf{V}_t(\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{X}^T\mathbf{y}) \quad (1.9)$$

$$\mathbf{V}_t = (\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1} \quad (1.10)$$

$$\alpha_t = \alpha_0 + t/2 \quad (1.11)$$

$$\beta_t = \beta_0 + \frac{1}{2}\left(\mathbf{w}_0^T\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{y}^T\mathbf{y} - \mathbf{w}_t^T\mathbf{V}_t^{-1}\mathbf{w}_t\right). \quad (1.12)$$

Integrating out the weight parameter $\mathbf{w}$ leads to coupling between the arms and makes it possible for the model to generalize observations of reward from one arm to another.

In this linear model, Thompson sampling draws a $\tilde{\mathbf{w}}$ from the posterior $p(\mathbf{w}|\mathcal{D}_t)$ and selects the arm with the highest expected reward under that parameter, i.e.,

$$a_{t+1} = \arg\max_a \mathbf{x}_a^T\tilde{\mathbf{w}} \text{ where } \tilde{\mathbf{w}} \sim p(\mathbf{w}|\mathcal{D}_t). \quad (1.13)$$

After arm a_{t+1} is pulled and y_{t+1} is observed, the posterior model can be readily updated using equations (1.9–1.12).

Various generalizations can be immediately seen. For example, by embedding the arms of a multi-armed bandit into a feature space denoted $\mathcal{X}$, we can generalize to objective functions f defined on the entire domain $\mathcal{X}$, thus unifying the multi-armed bandit problem with that of general global optimization:

$$\text{maximize} \quad f(\mathbf{x}) \qquad \text{s.t.} \quad \mathbf{x} \in \mathcal{X}. \quad (1.14)$$

In the multi-armed bandit, the optimization is over a discrete and finite set $\{\mathbf{x}_a\}_{a=1}^K \subset \mathcal{X}$, while global optimization seeks to solve the problem on, e.g., a compact set $\mathcal{X} \subset \mathbb{R}^d$.

As in other forms of regression, it is natural in increase the expressiveness of the model with non-linear basis functions. In particular, we can use J basis functions $\phi_j : \mathcal{X} \mapsto \mathbb{R}$, for $j = 1, \ldots, J$, and model the function f with a linear combination

$$f(\mathbf{x}) = \Phi(\mathbf{x})^T\mathbf{w}, \quad (1.15)$$

where $\Phi(\mathbf{x})$ is the column vector of concatenated features $\{\phi_j(\mathbf{x})\}_{j=1}^J$. Common classical examples of such ϕ_j include radial basis functions such as

$$\phi_j(\mathbf{x}) = \exp\left\{-\tfrac{1}{2}(\mathbf{x}-\mathbf{z}_j)^T\mathbf{\Lambda}(\mathbf{x}-\mathbf{z}_j)\right\}, \tag{1.16}$$

where $\mathbf{\Lambda}$ and $\{\mathbf{z}_j\}_{j=1}^J$ are model hyper-parameters, and Fourier bases

$$\phi_j(\mathbf{x}) = \exp\left\{-i\mathbf{x}^T\boldsymbol{\omega}_j\right\}, \tag{1.17}$$

with hyper-parameters $\{\boldsymbol{\omega}_j\}_{j=1}^J$.

Recently, such basis functions have also been learned from data by training deep belief networks [94], deep neural networks [129, 203], or by factoring the empirical covariance matrix of historical data [205, 98]. For example, in [42] each sigmoidal layer of an L layer neural network is defined as $\mathcal{L}_\ell(\mathbf{x}) := \sigma(\mathbf{W}_\ell\mathbf{x} + \mathbf{B}_\ell)$ where σ is some sigmoidal non-linearity, and $\mathbf{W}_\ell$ and $\mathbf{B}_\ell$ are the layer parameters. Then the feature map $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^J$ can be expressed as $\Phi(\mathbf{x}) = \mathcal{L}_L \circ \cdots \circ \mathcal{L}_1(\mathbf{x})$, where the final layer $\mathcal{L}_L$ has J output units. In [203], the weights of the last layer of a deep neural network are integrated out to result in a tractable Bayesian model with flexible learned basis functions.

Regardless of the feature map Φ, when conditioned on these basis functions, the posterior over the weights $\mathbf{w}$ can be computed analytically using (1.9-1.12). Let $\Phi(\mathbf{X})$ denote the $t \times J$ matrix where $[\Phi(\mathbf{X})]_{i,j} = \phi_j(\mathbf{x}_i)$; then the posterior is as in Bayesian linear regression, substituting $\Phi(\mathbf{X})$ for the design matrix $\mathbf{X}$.

1.3.3 Generalized linear models

While simple linear models capture the dependence between bandit arms in a straightforward and expressive way, the model as described does not immediately apply to other types of observations, such as binary or count data. Generalized linear models (GLMs) [163] allow more flexibility in the response variable through the introduction of a link function. Here we examine the GLM for binary data such as might arise from drug trials or AB testing.

The generalized linear model introduces a *link function* g that maps from the observation space into the reals. Most often, we consider the *mean function* g^{-1}, which defines the expected value of the response as a function of the underlying linear model: $\mathbb{E}[y|\mathbf{x}] = g^{-1}(\mathbf{x}^T\mathbf{w}) = f(\mathbf{x})$. In the case of binary data, a common choice is the logit link function, which leads to the familiar logistic regression model in which $g^{-1}(z) = 1/(1+\exp\{z\})$. In probit regression, the logistic mean function is replaced with the CDF of a standard normal. In either case, the observations y_i are taken to be Bernoulli random variables with parameter $g^{-1}(\mathbf{x}_i^T\mathbf{w})$.

Unfortunately, there is no conjugate prior for the parameters $\mathbf{w}$ when such a likelihood is used and so we must resort to approximate inference. Markov chain Monte Carlo (MCMC) methods [6] approximate the posterior with a sequence of samples that converge to the posterior; this is the approach taken in [193] on the probit model. In contrast, the Laplace approximation fits a Gaussian distribution to the posterior by matching the curvature of the posterior distribution at the mode. For example in [46], Bayesian logistic regression with a Laplace approximation was used to model click-throughs for the recommendation of news articles in a live experiment. In the generalized linear model, Thompson sampling draws a $\tilde{\mathbf{w}}$ from the posterior $p(\mathbf{w}|\mathcal{D}_t)$ using MCMC or a Laplace approximation, and then selects the arm with the highest expected reward given the sampled parameter $\tilde{\mathbf{w}}$, i.e., $a_{t+1} = \arg\max_a g^{-1}(\mathbf{x}_a^T \tilde{\mathbf{w}})$.

1.3.4 Related literature

There are various strategies beyond Thompson sampling for Bayesian optimization that will be discussed in succeeding sections of the paper. However, before we can reason about which selection strategy is optimal, we need to establish what the goal of the series of sequential experiments will be. Historically, these goals have been quantified using the principle of maximum expected utility. In this framework, a utility function U is prescribed over a set of experiments $\mathbf{X} := \{\mathbf{x}_i\}_{i=1}^t$, their outcomes $\mathbf{y} := \{y_i\}_{i=1}^t$, and the model parameter $\mathbf{w}$. The unknown model parameter and outcomes are marginalized out to produce the expected utility

$$\alpha(\mathbf{X}) := \mathbb{E}_{\mathbf{w}} \mathbb{E}_{\mathbf{y}|\mathbf{X},\mathbf{w}} \left[U(\mathbf{X}, \mathbf{y}, \mathbf{w}) \right] , \tag{1.18}$$

which is then maximized to obtain the best set of experiments with respect to the given utility U and the current posterior. The expected utility α is related to acquisition functions in Bayesian optimization, reviewed in Section 1.5. Depending on the literature, researchers have focussed on different goals which we briefly discuss here.

1.3.4.1 Active learning and experimental design

In this setting, we are usually concerned with learning about $\mathbf{w}$, which can be framed in terms of improving an estimator of $\mathbf{w}$ given the data. One popular approach is to select points that are expected to minimize the differential entropy of the posterior distribution $p(\mathbf{w}|\mathbf{X}, \mathbf{y})$, i.e., maximize:

$$\alpha(\mathbf{X}) = \mathbb{E}_{\mathbf{w}} \mathbb{E}_{\mathbf{y}|\mathbf{X},\mathbf{w}} \left[\int p(\mathbf{w}'|\mathbf{X}, \mathbf{y}) \log p(\mathbf{w}'|\mathbf{X}, \mathbf{y}) \mathrm{d}\mathbf{w}' \right].$$

In the Bayesian experimental design literature, this criterion is known as the D-optimality utility and was first introduced by Lindley [135]. Since this seminal work, many alternative utilities have been proposed in the experimental design literature. See [45] for a detailed survey.

In the context of A/B testing, following this strategy would result in exploring all possible combinations of artwork, font, and sizes, no matter how bad initial outcomes were. This is due to the fact that the D-optimality utility assigns equal value to any information provided about any advertisement configuration, no matter how effective.

In contrast to optimal experimental design, Bayesian optimization explores uncertain arms $a \in \{1, \ldots, K\}$, or areas of the search space $\mathcal{X}$, only until they can confidently be ruled out as being suboptimal. Additional impressions of suboptimal ads would be a waste of our evaluation budget. In Section 1.5, we will introduce another differential entropy based utility that is better suited for the task of optimization and that partially bridges the gap between optimization and improvement of estimator quality.

1.3.4.2 Multi-armed bandit

Until recently, the multi-armed bandit literature has focussed on maximizing the sum of rewards y_i, possibly discounted by a *discount factor* $\gamma \in (0, 1]$:

$$\alpha(\mathbf{X}) = \mathbb{E}_{\mathbf{w}} \mathbb{E}_{\mathbf{y}|\mathbf{X},\mathbf{w}} \left[\sum_{i=1}^{t} \gamma^{i-1} y_i \right]. \tag{1.19}$$

When $\gamma < 1$, a Bayes-optimal sequence $\mathbf{X}$ can be computed for the Bernoulli bandit via dynamic programming, due to Gittins [78]. However, this solution is intractable for general reward distributions, and so in practice sequential heuristics are used and analyzed in terms of a frequentist measure, namely *cumulative regret* [128, 193, 205, 46, 183].

Cumulative regret is a frequentist measure defined as

$$R_t(\mathbf{w}) = \sum_{i=1}^{t} f^{\star}_{\mathbf{w}} - f_{\mathbf{w}}(\mathbf{x}_{a_i}), \tag{1.20}$$

where $f^{\star}_{\mathbf{w}} := \max_a f_{\mathbf{w}}(\mathbf{x}_a)$ denotes the best possible expected reward. Whereas the D-optimality utility leads to too much exploration, the cumulative regret encourages exploitation by including intermediate selections a_i in the final loss function R_t. For certain tasks, this is an appropriate loss function: for example, when sequentially selecting ads, each impression incurs an opportunity cost. Meanwhile, for other tasks such as model selection, we typically have a predetermined evaluation budget for optimization and only the performance of the final recommended model should be assessed by the loss function.

Recently, there has been growing interest in the *best arm identification* problem, which is more suitable for the model selection task [142, 37, 10, 68, 67, 98]. When using Bayesian surrogate models, this is equivalent to performing Bayesian optimization on a finite, discrete domain. In this so-called pure exploration settings, in addition to a selection strategy, a *recommendation strategy* ρ is specified to recommend an arm (or ad or drug) at the end of the experimentation based on observed data. The experiment is then judged via the *simple regret*, which depends on the recommendation $\bar{a} = \rho(\mathcal{D})$:

$$r_t(\mathbf{w}) = f^{\star}_{\mathbf{w}} - f_{\mathbf{w}}(\mathbf{x}_{\bar{a}})\,. \tag{1.21}$$

1.4 Non-parametric models

In this section, we show how it is possible to marginalize away the weights in Bayesian linear regression and apply the kernel trick to construct a Bayesian non-parametric regression model. As our starting point, we assume the observation variance σ^2 is fixed and place a zero-mean Gaussian prior on the regression coefficients $p(\mathbf{w}|\mathbf{V}_0) = \mathcal{N}(0, \mathbf{V}_0)$. In this case, we notice that it possible to analytically integrate out the weights, and in doing so we preserve Gaussianity:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \sigma^2) &= \int p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2)\, p(\mathbf{w}|0, \mathbf{V}_0)\, \mathrm{d}\mathbf{w} \\ &= \int \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I})\, \mathcal{N}(\mathbf{w}|0, \mathbf{V}_0)\, \mathrm{d}\mathbf{w} \\ &= \mathcal{N}(\mathbf{y}|0, \mathbf{X}\mathbf{V}_0\mathbf{X}^T + \sigma^2\mathbf{I})\,. \end{aligned} \tag{1.22}$$

As noted earlier, it can be useful to introduce basis functions ϕ and in the context of Bayesian linear regression we in effect replace the design matrix $\mathbf{X}$ with a feature mapping matrix $\mathbf{\Phi} = \Phi(\mathbf{X})$. In Equation (1.22), this results in a slightly different Gaussian for weights in feature space:

$$p(\mathbf{y}|\mathbf{X}, \sigma^2) = \mathcal{N}(\mathbf{y}|0, \mathbf{\Phi}\mathbf{V}_0\mathbf{\Phi}^T + \sigma^2\mathbf{I}) \tag{1.23}$$

Note that $\mathbf{\Phi}\mathbf{V}_0\mathbf{\Phi}^T \in \mathbb{R}^{n\times n}$ is a symmetric positive semi-definite matrix made up of pairwise inner products between each of the data in their basis function representations. The celebrated *kernel trick* emerges from the observation that these inner products can be equivalently computed by evaluating the corresponding kernel function k for all pairs to form the matrix $\mathbf{K}$

$$\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i)\mathbf{V}_0\mathbf{\Phi}(\mathbf{x}_j)^T \tag{1.24}$$

$$= \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle_{\mathbf{V}_0}\,. \tag{1.25}$$

The kernel trick allows us to specify an intuitive similarity between pairs of points, rather than a feature map $\mathbf{\Phi}$, which in practice can be hard to define. In other words, we can either think of predictions as depending directly on features $\mathbf{\Phi}$, as in the linear regression problem, or on kernels k, as in the lifted variant, depending on which paradigm is more interpretable or computationally tractable. Indeed, the former requires a $J \times J$ matrix inversion compared to the latter's $t \times t$.

Note also that this approach not only allows us to compute the marginal likelihood of data that have already been seen, but it enables us to make predictions of outputs $\mathbf{y}_\star$ at new locations $\mathbf{X}_\star$. This can be done by observing that

$$p(\mathbf{y}_\star|\mathbf{X}_\star, \mathbf{X}, \mathbf{y}, \sigma^2) = \frac{p(\mathbf{y}_\star, \mathbf{y}|\mathbf{X}_\star, \mathbf{X}, \sigma^2)}{p(\mathbf{y}|\mathbf{X}, \sigma^2)}\,. \tag{1.26}$$

Both the numerator and the denominator are Gaussian with the form appearing in Equation (1.23), and so the predictions are jointly Gaussian and can be computed via some simple linear algebra. Critically, given a kernel k, it becomes unnecessary to explicitly define or compute the features $\mathbf{\Phi}$ because both the predictions and the marginal likelihood only depend on $\mathbf{K}$.

1.4.1 The Gaussian process

By kernelizing a marginalized version of Bayesian linear regression, what we have really done is construct an object called a Gaussian process. The Gaussian process $\mathrm{GP}(\mu_0, k)$ is a non-parametric model that is fully characterized by its prior mean function $\mu_0 : \mathcal{X} \mapsto \mathbb{R}$ and its positive-definite kernel, or covariance function, $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ [176]. Consider any finite collection[2] of n points $\mathbf{x}_{1:t}$, and define variables $f_i := f(\mathbf{x}_i)$ and $y_{1:t}$ to represent the unknown function values and noisy observations, respectively. In Gaussian process regression, we assume that $\mathbf{f} := f_{1:t}$ are jointly Gaussian resulting in the following model:

$$\mathbf{f}|\mathbf{X} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}) \tag{1.27}$$

where the elements of the mean vector and covariance matrix are defined as $m_i := \mu_0(\mathbf{x}_i)$ and $K_{i,j} := k(\mathbf{x}_i, \mathbf{x}_j)$, respectively. Equation (1.27) represents the prior distribution $p(\mathbf{f})$ induced by the GP.

Let $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^t$ denote the set of observations and $\mathbf{x}$ denote an arbitrary test point. As mentioned when kernelizing linear regression, the random variable $f(\mathbf{x})$ conditioned on observations $\mathcal{D}_t$ is also normally distributed with the following posterior mean

[2] We use the notation $z_{i:j} = \{z_i, \ldots, z_j\}$.

and variance functions

$$\mu_t(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\mathsf{T}\mathbf{K}^{-1}(\mathbf{f} - \mathbf{m}) \tag{1.28}$$

$$\sigma_t^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\mathsf{T}\mathbf{K}^{-1}\mathbf{k}(\mathbf{x})\,, \tag{1.29}$$

where $\mathbf{k}(\mathbf{x})$ is a vector of covariance terms between $\mathbf{x}$ and $\mathbf{x}_{1:t}$.

Often we do not have access to the function values themselves, but only noisy versions thereof. Assuming that the observations are normally distributed given $\mathbf{f}$:

$$\mathbf{y}|\mathbf{f}, \sigma^2 \sim \mathcal{N}(\mathbf{f}, \sigma^2\mathbf{I}), \tag{1.30}$$

the posterior predictive of $f(\mathbf{x})$ is still analytically tractable where

$$\mu_t(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^\mathsf{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}) \tag{1.31}$$

$$\sigma_t^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\mathsf{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})\,. \tag{1.32}$$

The posterior mean and variance evaluated at any point $\mathbf{x}$ represent the model's prediction and uncertainty, respectively, in the objective function at the point $\mathbf{x}$. These posterior functions are used to select the next query point $\mathbf{x}_{t+1}$ as detailed in Section 1.5.

1.4.2 Common kernels

In Gaussian process regression, the covariance function k dictates the structure of the response functions we can fit. For instance, if we expect our response function to be periodic, we can prescribe a periodic kernel. In this review, we focus on stationary kernels, which are shift invariant.

Matérn kernels are a very flexible class of stationary kernels. These kernels are parameterized by a smoothness parameter $\nu > 0$, so called because samples from a GP with such a kernel are differentiable $\lfloor \nu - 1 \rfloor$ times [176]. The exponential kernel is a special case of the Matérn kernel with $\nu = \frac{1}{2}$, and the squared exponential kernel is the limiting kernel when $\nu \to \infty$. The following are the most commonly used kernels, labelled by the smoothness parameter, omitting the factor of $\frac{1}{2}$.

$$k_{\text{MATÉRN1}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-r) \tag{1.33}$$

$$k_{\text{MATÉRN3}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{3}r)(1 + \sqrt{3}r) \tag{1.34}$$

$$k_{\text{MATÉRN5}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{5}r)(1 + \sqrt{5}r + \tfrac{5}{3}r^2) \tag{1.35}$$

$$k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\tfrac{1}{2}r^2), \tag{1.36}$$

where $r^2 = (\mathbf{x} - \mathbf{x}')^\mathsf{T}\mathbf{\Lambda}(\mathbf{x} - \mathbf{x}')$ and $\mathbf{\Lambda}$ is a diagonal matrix of d squared length scales θ_i^2. This family of covariance functions are therefore parameterized by an amplitude and

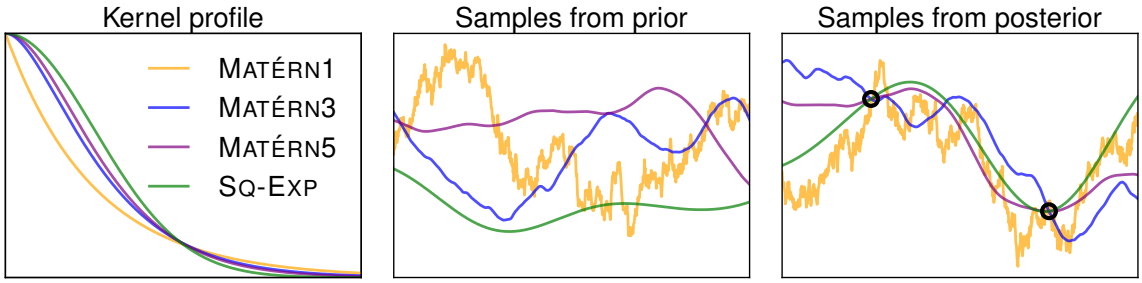


Figure 1.3: **Left:** Visualization of various kernel profiles. The horizontal axis represents the distance $r > 0$. **Middle:** Samples from GP priors with the corresponding kernels. **Right:** Samples from GP posteriors given two data points (black circles). Note the sharper drop in the Matérn1 kernel leads to rough features in the associated samples, while samples from a GP with the Matérn3 and Matérn5 kernels are increasingly smooth.

d length scale hyper parameters, jointly denoted θ. Covariance functions with learnable length scale parameters are also known as automatic relevance determination (ARD) kernels. Figure 1.3 provides a visualization of the kernel profiles and samples from the corresponding priors and posteriors.

1.4.3 Prior mean functions

It is common, in the Bayesian optimization literature, to set the prior mean to $\mu_0(\mathbf{x}) \equiv 0$ [205, 34]. The prior mean, however, could provide a possible offset. In practice, this function is sometimes set to a constant $\mu_0(\mathbf{x}) \equiv \mu_0$ and inferred from data using techniques covered in Section 1.6.1. However, the prior mean function is a principled way of incorporating expert knowledge of the objective function, if it is available, and the following analysis can be readily applied to non-constant functions μ_0.

1.4.4 Marginal likelihood

Another attractive property of the Gaussian process model is that it provides an analytical expression for the marginal likelihood of the data, where marginal refers to the fact that the unknown latent function f is marginalized out. The expression for the log marginal likelihood is simply given by[3]:

$$\log p(\mathbf{y}|\mathbf{x}_{1:t}, \theta) = -\frac{1}{2}(\mathbf{y} - \mathbf{m}_\theta)^\intercal (\mathbf{K}^\theta + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}_\theta) - \frac{1}{2}\log|\mathbf{K}^\theta + \sigma^2\mathbf{I}| - \frac{t}{2}\log(2\pi), \quad (1.37)$$

[3]For the purpose of brevity, we omit the marginal likelihood formula for the noiseless case. The interested reader could find the formula in Chapter 2 of [176].

where in a slight abuse of notation we augment the vector $\theta := (\theta_{0:d}, \mu_0, \sigma^2)$; and the dependence on θ is made explicit by adding a superscript to the covariance matrix $\mathbf{K}^\theta$. The marginal likelihood is very useful in learning the hyper-parameters, as we will see in Section 1.6.1. The right hand side of (1.37) can be broken into three terms: the first term quantifies how well the model fits the data, which is simply a Mahalanobis distance between the model predictions and the data; the second term quantifies the model complexity – smoother covariance matrices will have smaller determinants and therefore lower complexity penalties; finally, the last term is simply a linear function of the number of data points n, indicating that the likelihood of data tends to decrease with larger datasets.

Conveniently, as long as the kernel is differentiable with respect to its hyper-parameters θ, the marginal likelihood can be differentiated and can therefore be optimized in an off-the-shelf way to obtain a type II maximum likelihood (MLII) or empirical Bayes estimate of the kernel parameters. When data is scarce this can overfit the available data. In Section 1.6.1 we will review various practical strategies for learning hyper-parameters which all use the marginal likelihood.

1.4.5 Computational costs and other regression models

Although we have analytic expressions, exact inference in Gaussian process regression is $O(t^3)$ where t is the number of observations. This cost is due to the inversion of the covariance matrix. In practice, the Cholesky decomposition can be computed once and saved so that subsequent predictions are $O(t^2)$. However, this Cholesky decomposition must be recomputed every time the kernel hyper-parameters are changed, which usually happens at every iteration (see Section 1.6.1). For large datasets, or large function evaluation budgets in the Bayesian optimization setting, the cubic cost of exact inference is prohibitive and there have been many attempts at reducing this computational burden via approximation techniques. In this section we review two sparsification techniques for Gaussian processes and the alternative random forest regression.

1.4.5.1 Sparse pseudo-input Gaussian processes (SPGP)

One early approach to modelling large t with Gaussian processes considered using $m < t$ *inducing pseudo-inputs* to reduce the rank of the covariance matrix to m, resulting in a significant reduction in computational cost [195, 200]. By forcing the interaction between the t data points $\mathbf{x}_{1:t}$ and any test point $\mathbf{x}$ to go through this set of m inducing pseudo-inputs, these methods can compute an approximate posterior in $O(tm^2 + m^3)$ time. Pseudo-input

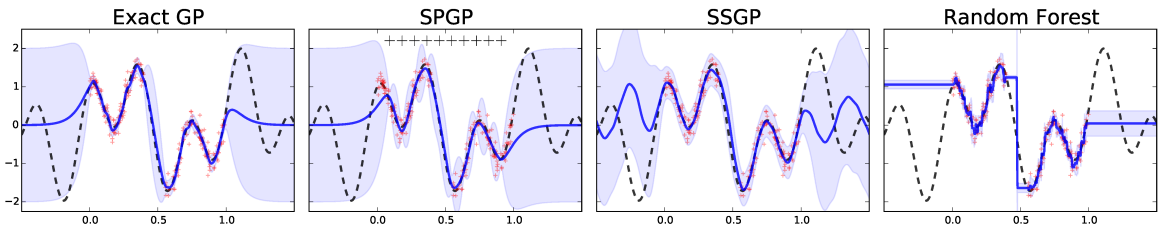


Figure 1.4: Comparison of surrogate regression models. Four different surrogate model posteriors are shown in blue (shaded area delimits 95% credible intervals), given noisy evaluations (red crosses) of a synthetic function (dashed line). The 10 pseudo-inputs for the SPGP method are shown as black crosses. The SSGP model used a basis of 80 Fourier features.

methods have since been unified in a single theory based on the following overarching approximation.

Let $\mathbf{f}$ and $\mathbf{f}_\star$ denote two sets of latent function values, commonly representing the function at training and test locations, respectively. The simplifying assumption is that $\mathbf{f}$ and $\mathbf{f}_\star$ are independent given a third set of variables $\mathbf{u}$, such that

$$p(\mathbf{f}_\star, \mathbf{f}) = \int p(\mathbf{f}_\star, \mathbf{f}, \mathbf{u}) \, \mathrm{d}\mathbf{u} \tag{1.38}$$

$$\approx \int q(\mathbf{f}_\star|\mathbf{u}) \, q(\mathbf{f}|\mathbf{u}) \, p(\mathbf{u}) \, \mathrm{d}\mathbf{u} = q(\mathbf{f}, \mathbf{f}_\star) \tag{1.39}$$

where $\mathbf{u}$ is the vector of function values at the pseudo-inputs. All sparse pseudo-input GP approximations can be specified in terms of the form used for the training and test conditionals, $q(\mathbf{f}|\mathbf{u})$ and $q(\mathbf{f}_\star|\mathbf{u})$, respectively [173].

In the seminal works on pseudo-input methods, the locations of the pseudo-inputs were selected to optimize the marginal likelihood of the SPGP [195, 200]. In contrast, a variational approach has since been proposed to marginalize the pseudo-inputs to maximize fidelity to the original exact GP [212] rather than the likelihood of the approximate GP.

The computational savings in the pseudo-input approach to approximating the GP comes at the cost of poor variance estimates. As can be observed in Figure 1.4, the uncertainty (blue shaded area) exhibits unwanted pinching at pseudo-inputs, while it is overly conservative in between and away from pseudo-inputs. In this instance, the 10 inducing points, indicated with black crosses, were not optimized to emphasize the potential pathologies of the method. Since in Bayesian optimization we use the credible intervals to guide exploration, these artefacts can mislead our search.

1.4.5.2 Sparse spectrum Gaussian processes (SSGP)

While inducing pseudo-inputs reduce computational complexity by using a fixed number of points in the search space, sparse spectrum Gaussian processes (SSGP) take a similar approach to the kernel's spectral space [130]. Bochner's theorem states that any stationary kernel $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$ has a positive and finite Fourier spectrum $s(\boldsymbol{\omega})$, i.e.,

$$k(\mathbf{x}) = \frac{1}{(2\pi)^d} \int e^{-i\boldsymbol{\omega}^\mathsf{T}\mathbf{x}} s(\boldsymbol{\omega}) \, \mathrm{d}\boldsymbol{\omega} \, . \tag{1.40}$$

Since the spectrum is positive and bounded, it can be normalized such that $p(\boldsymbol{\omega}) := s(\boldsymbol{\omega})/\nu$ is a valid probability density function. In this formulation, evaluating the stationary kernel is equivalent to computing the expectation of the Fourier basis with respect to its specific spectral density $p(\boldsymbol{\omega})$ as in the following,

$$k(\mathbf{x}, \mathbf{x}') = \nu \, \mathbb{E}_{\boldsymbol{\omega}}[e^{-i\boldsymbol{\omega}^\mathsf{T}(\mathbf{x}-\mathbf{x}')}]. \tag{1.41}$$

As the name suggests, SSGP approximates this expectation via Monte Carlo estimation using m samples drawn from the spectral density so that

$$k(\mathbf{x}, \mathbf{x}') \approx \frac{\nu}{m} \sum_{i=1}^{m} e^{-i\boldsymbol{\omega}^{(i)\mathsf{T}}\mathbf{x}} e^{i\boldsymbol{\omega}^{(i)\mathsf{T}}\mathbf{x}'} \tag{1.42}$$

where $\boldsymbol{\omega}^{(i)} \sim s(\boldsymbol{\omega})/\nu$. The resulting finite dimensional problem is equivalent to Bayesian linear regression with m basis functions and the computational cost is once again reduced to $O(tm^2 + m^3)$. Please see Chapter 5 for a more detailed discussion as well as an application of this technique to the Matérn kernels.

As with the pseudo-inputs, the spectral points can also be tuned via marginal likelihood optimization. Although this violates the Monte Carlo assumption and introduces a risk of overfitting, it allows for a smaller number of basis functions with good predictive power [130]. Once again, in Figure 1.4 we have not tuned the 80 spectral points in this way. Whereas around observed data (red crosses) the uncertainty estimates are smoother than the pseudo-inputs method, away from observations both the prediction and uncertainty regions exhibit spurious oscillations. This is highly undesirable for Bayesian optimization where we expect our surrogate model to fall back on the prior away from observed data.

1.4.5.3 Random forests

Finally, as an alternative to Gaussian processes, random forest regression has been proposed as an expressive and flexible surrogate model in the context of sequential model-based algorithm configuration (SMAC) [106]. Introduced in 2001 [31], random forests are a

class of scalable and highly parallelizable regression models that have been very successful in practice [54]. More precisely, the random forest is an ensemble method where the weak learners are decision trees trained on random subsamples of the data [31]. Averaging the predictions of the individual trees produces an accurate response surface.

Subsampling the data, and the inherent parallelism of the random forest regression model give SMAC the ability to readily scale to large evaluation budgets, beyond where the cubic cost of an exact GP would be infeasible. Similarly, at every decision node of every tree, a fixed-sized subset of the available dimensions is sampled to fit a decision rule; this subsampling also helps the random forest scale to high-dimensional search spaces. Perhaps most importantly, random forests inherit the flexibility of decision trees when dealing with various data types; they can easily handle categorical and conditional variables. For example, when considering a decision node, the algorithm can exclude certain search dimensions from consideration when the path leading up to said node includes a particular boolean feature that is turned off.

The exploration strategy in SMAC still requires an uncertainty estimate for predictions at test points. While the random forest does not provide an estimate of the variance of its predictions, Hutter et al. proposed using the empirical variance in the predictions across trees in the ensemble [106]. Though these are not principled uncertainty estimates, this heuristic has been shown to work well in practice for the SMAC algorithm.

Although random forests are good interpolators in the sense that they output good predictions in the neighbourhood of training data, they are very poor extrapolators. Indeed, far from the data, the predictions of all trees could be identical, resulting in a poor prediction; more importantly, using the variance estimate of SMAC results in extremely confident intervals. In Figure 1.4 for example, away from data the shaded area is very narrow around a very poor constant prediction. Even more troubling is the fact that in areas of missing data multiple conflicting predictions can cause the empirical variance to blow up sharply, as can be seen in Figure 1.4. While Gaussian processes are also poor extrapolators (when used with local kernels), they produce relatively uncertain predictions away from the data by reverting to the prior – a more desirable behavior when trading off exploration and exploitation.

Finally, another drawback of random forests for Bayesian optimization is that the response surface is discontinuous and non-differentiable so gradient based optimization methods are not applicable. SMAC relies on a combination of local and random search when maximizing the acquisition function.

1.5 Acquisition functions

Thus far, we have described the statistical model used to represent our belief about the unknown function f at iteration t. We have not described the exact mechanism or policy for selecting the sequence of query points $\mathbf{x}_{1:t}$. One could select these arbitrarily but this would be wasteful. Instead, there is a rich literature on selection strategies that utilize the posterior model to guide the sequential search, i.e., the selection of the next query point $\mathbf{x}_{t+1}$ given $\mathcal{D}_t$.

Consider the utility function $U : \mathbb{R}^d \times \mathbb{R} \times \Theta \mapsto \mathbb{R}$ which maps an arbitrary query point $\mathbf{x}$, its corresponding function value $v = f(\mathbf{x})$, and a setting of the model hyper-parameters θ to a measure of quality of the experiment, e.g., how much information this query will provide as in [135]. Given some data accumulated thus far, we can marginalize the unseen outcome y and the unknown model hyper-parameters θ to obtain the expected utility of a query point $\mathbf{x}$:

$$\alpha(\mathbf{x}; \mathcal{D}_t) = \mathbb{E}_\theta \mathbb{E}_{v|\mathbf{x},\theta}[U(\mathbf{x}, v, \theta)] \tag{1.43}$$

For simplicity, in this section we will mostly ignore the θ dependence and we will discuss its marginalization in Section 1.6.1.

Whereas in experimental design and decision theory, the function α is called the expected utility, in Bayesian optimization it is often called the acquisition or infill function. These acquisition functions are carefully designed to trade off exploration of the search space and exploitation of current promising areas. We first present traditional improvement-based and optimistic acquisition functions, followed by more recent information-based approaches.

1.5.1 Improvement-based policies

Improvement-based acquisition functions favor points that are likely to improve upon an incumbent target τ. An early strategy in the literature, probability of improvement (PI) [127], measures the probability that a point $\mathbf{x}$ leads to an improvement upon τ. Since the posterior distribution of $v = f(\mathbf{x})$ is Gaussian, we can analytically compute this probability as follows:

$$\alpha_{\text{PI}}(\mathbf{x}; \mathcal{D}_t) := \mathbb{P}[v > \tau] = \Phi\left(\frac{\mu_t(\mathbf{x}) - \tau}{\sigma_t(\mathbf{x})}\right), \tag{1.44}$$

where Φ is the standard normal cumulative distribution function. Recall that α_{PI} is then maximized to select the next query point. For this criterion, the utility function is simply an indicator of improvement $U(\mathbf{x}, v, \theta) = \mathbb{I}[v > \tau]$, where the utility function is expressed

(and marginalized) with respect to the latent variable v. Therefore, all improvements are treated equal and PI simply accumulates the posterior probability mass above τ at $\mathbf{x}$.

Although probability of improvement can perform very well when the target is known, in general the heuristic used for an unknown target τ causes PI to exploit quite aggressively [112].

One could instead measure the expected improvement (EI) [156] which incorporates the *amount* of improvement. This new criterion corresponds to a different utility that is called the improvement function, denoted by $I(\mathbf{x})$. Formally, the improvement function I is defined as follows

$$I(\mathbf{x}, v, \theta) := (v - \tau)\,\mathbb{I}(v > \tau). \tag{1.45}$$

Note that $I > 0$ only if there is an improvement. Once again, because the random variable v is normally distributed, the expectation can be computed analytically as follows

$$\begin{aligned}\alpha_{\mathrm{EI}}(\mathbf{x}; \mathcal{D}_t) &:= \mathbb{E}\left[I(\mathbf{x}, v, \theta)\right] \\ &= (\mu_t(\mathbf{x}) - \tau)\Phi\left(\frac{\mu_t(\mathbf{x}) - \tau}{\sigma_t(\mathbf{x})}\right) \\ &\quad + \sigma_t(\mathbf{x})\phi\left(\frac{\mu_t(\mathbf{x}) - \tau}{\sigma_t(\mathbf{x})}\right),\end{aligned} \tag{1.46}$$

when $\sigma_t > 0$ and vanishes otherwise. Here, not to be confused with the previous section, ϕ is the standard normal probability density function. These improvement strategies have been empirically studied in the literature [113, 112, 34] and recently convergence rates have been proven for EI [39].

Finally, although the target objective value (i.e., the best reachable objective value) is often unknown, in practice τ is adaptively set to the best observed value $y^+ = \max_{i=1:t} y_i$. Whereas for PI this heuristic can lead to an overly greedy optimization [112], it works reasonably with EI in practice [202]. When the objective function being maximized is very noisy, using the highest mean value as the target is reasonable. See Chapter 3 for a more detailed discussion.

1.5.2 Optimistic policies

Dating back to the seminal work of Lai & Robbins [128] on the multi-armed bandit problem, the upper confidence bound criterion has been a popular way of negotiating exploration and exploitation, often with provable cumulative regret bounds. The guiding principle behind this class of strategies is to be optimistic in the face of uncertainty. Indeed, using the upper confidence for every query point $\mathbf{x}$ corresponds to effectively using a fixed

probability best case scenario according to the model. Originally, the upper confidence was given by frequentist Chernoff–Hoeffding bounds [11].

More recently, the Gaussian process upper confidence bound (GP-UCB [205]) algorithm was proposed as a Bayesian optimistic algorithm with provable cumulative regret bounds. In the deterministic case, a branch-and-bound extension to GP-UCB was proven to have exponentially vanishing instantaneous regret [55]. The GP-UCB algorithm has since been generalized to other Bayesian models by considering upper quantiles [114] instead of Equation (1.47) defined below, which is more reminiscent of frequentist concentration bounds. In the GP case, since the posterior at any arbitrary point $\mathbf{x}$ is a Gaussian, any quantile of the distribution of $f(\mathbf{x})$ is computed with its corresponding value of β_n as follows:

$$\alpha_{\text{UCB}}(\mathbf{x}; \mathcal{D}_t) := \mu_t(\mathbf{x}) + \beta_t \sigma_t(\mathbf{x}). \tag{1.47}$$

There are theoretically motivated guidelines for setting and scheduling the hyper-parameter β_t to achieve optimal regret [205] and, as with τ in the improvement policies, tuning this parameter within these guidelines can offer a performance boost.

Finally, there also exist variants of these algorithms for the contextual bandits [215] (see Section 1.9.4) and parallel querying [58] (see Section 1.6.5).

1.5.3 Information-based policies

In contrast to the acquisition functions introduced so far, information-based policies consider the posterior distribution over the unknown minimizer $\mathbf{x}^\star$, denoted $p_\star(\mathbf{x}|\mathcal{D}_t)$. This distribution is implicitly induced by the posterior over objective functions f. There are two policies in this class, namely Thompson sampling and entropy search.

Though it was introduced in 1933 [210], Thompson sampling has attracted renewed interest in the multi-armed bandit community, producing empirical evaluations [193, 46] as well as theoretical results [115, 2, 183]. Thompson sampling (TS) is a randomized strategy which samples a reward function from the posterior and selects the arm with the highest simulated reward. Therefore the selection made by TS can be expressed as the randomized acquisition function $\mathbf{x}_{t+1} \sim p_\star(\mathbf{x}|\mathcal{D}_t)$.

However, in continuous search spaces, the analog of Thompson sampling is to draw a continuous function $f^{(t)}$ from the posterior GP and optimize it to obtain $\mathbf{x}_{t+1}$. In order to be optimized, the sample $f^{(t)}$ needs to be fixed so it can be queried at arbitrary points; unfortunately, it is not clear how to fix an exact sample from the GP. However, using recent spectral sampling techniques [27, 174, 130], we can draw an approximate sample from the

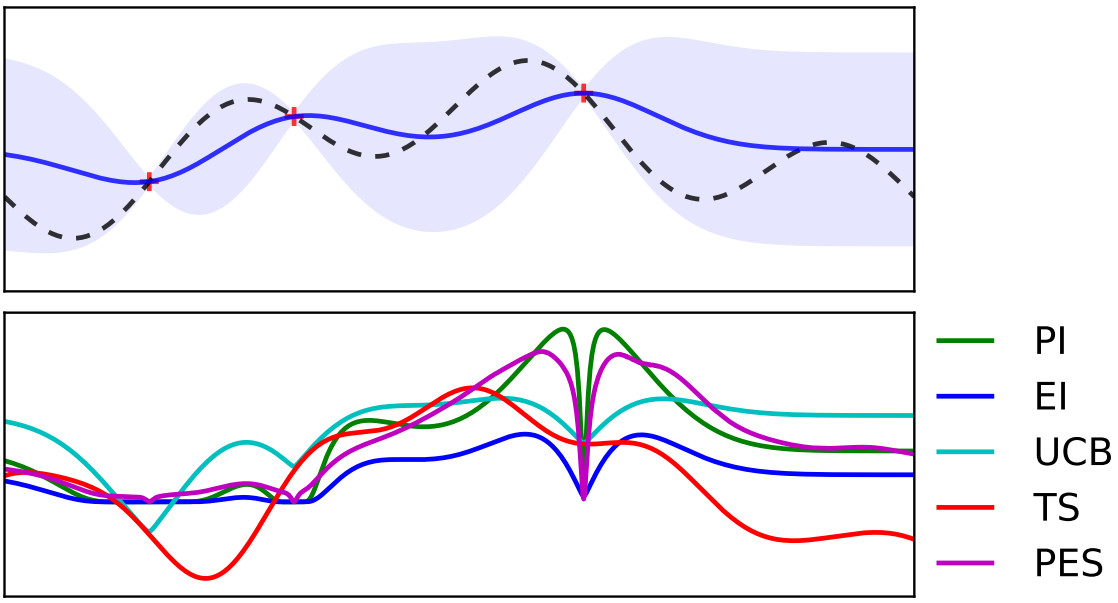


Figure 1.5: Visualization of the surrogate regression model and various acquisition functions. **(Top)** The true objective function is shown as a dashed line and the probabilistic regression model is shown as a blue line with a shaded region delimiting the $2\sigma_t$ credible intervals. Finally, the observations are shown as red crosses. **(Bottom)** Four acquisition functions are shown. In the case of PI, the optimal mode is much closer to the best observation as in the alternative methods, which explains its greedy behavior. In contrast, the randomization in TS allows it to explore more aggressively.

posterior that can be evaluated at any arbitrary point $\mathbf{x}$ [92], which extends TS to continuous search spaces. As an acquisition function, TS can be formulated as

$$\alpha_{\text{TS}}(\mathbf{x}; \mathcal{D}_n) := f^{(t)}(\mathbf{x}) \\ \text{where } f^{(t)} \overset{s.s.}{\sim} \text{GP}(\mu_0, k|\mathcal{D}_t) \tag{1.48}$$

where $\overset{s.s.}{\sim}$ indicates approximate simulation via spectral sampling. Empirical evaluations show good performance which, however, seems to deteriorate in high dimensional problems, likely due to aggressive exploration [198].

Instead of sampling the distribution $p_\star(\mathbf{x}|\mathcal{D}_t)$, entropy search (ES) techniques aim to reduce the uncertainty in the location $\mathbf{x}^\star$ by selecting the point that is expected to cause the largest reduction in entropy of the distribution $p_\star(\mathbf{x}|\mathcal{D}_t)$ [219, 90, 92]. In terms of utility, entropy search methods use the information gain defined as follows

$$U(\mathbf{x}, y, \theta) = H(\mathbf{x}^\star|\mathcal{D}_t) - H(\mathbf{x}^\star|\mathcal{D}_t \cup \{(\mathbf{x}, y)\}), \tag{1.49}$$

where the θ implicitly parameterizes the distribution of y.

In other words, ES measures the expected information gain from querying an arbitrary point $\mathbf{x}$ and selects the point that offers the most information about the unknown $\mathbf{x}_\star$. The acquisition function for ES can be expressed formally as

$$\alpha_{\text{ES}}(\mathbf{x}; \mathcal{D}_t) := H(\mathbf{x}^\star|\mathcal{D}_t) - \mathbb{E}_{y|\mathcal{D}_t,\mathbf{x}} H(\mathbf{x}^\star|\mathcal{D}_t \cup \{(\mathbf{x}, y)\})$$

where $H(\mathbf{x}^\star|\mathcal{D}_t)$ denotes the differential entropy of the posterior distribution $p_\star(\mathbf{x}|\mathcal{D}_t)$, and the expectation is over the distribution of the random variable $y \sim \mathcal{N}(\mu_t(\mathbf{x}), \sigma_t^2(\mathbf{x}) + \sigma^2)$.

Once again, this function is not tractable for continuous search spaces $\mathcal{X}$ so approximations must be made. Early work discretized the space $\mathcal{X}$ and computed the conditional entropy via Monte Carlo sampling [219]. More recent work uses a discretization of the $\mathcal{X}$ to obtain a smooth approximation to $p_\star$ and its expected information gain [90]. This method is unfortunately $O(M^4)$ where M is the number of discrete so-called representer points.

Finally, predictive entropy search (PES) removes the need for a discretization and approximates the acquisition function in $O((t+d)^3)$ time, which, for $d < t$ is of the same order as EI [92]. This is achieved by using the symmetric property of mutual information to rewrite $\alpha_{\text{ES}}(\mathbf{x})$ as

$$\alpha_{\text{PES}}(\mathbf{x}; \mathcal{D}_t) := H(y|\mathcal{D}_t, \mathbf{x}) - \mathbb{E}_{\mathbf{x}^\star|\mathcal{D}_t}\big[H(y|\mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star)\big]$$

The expectation can be approximated via Monte Carlo with Thompson samples; and three simplifying assumptions are made to compute $H(y|\mathcal{D}_t, \mathbf{x}, \mathbf{x}^\star)$. Empirically, this algorithm

has been shown to perform as well or better than the discretized version without the unappealing quartic term [92], making it arguably the state of the art in entropy search approximation.

1.5.4 Portfolios of acquisition functions

No single acquisition strategy provides better performance over all problem instances. In fact, it has been empirically observed that the preferred strategy can change at various stages of the sequential optimization process. To address this issue, [97] proposed the use of a portfolio containing multiple acquisition strategies. At each iteration, each strategy in the portfolio provides a candidate query point and meta-criterion is used to select the next query point among these candidates. The meta-criterion is analogous to an acquisition function at a higher level; whereas acquisition functions are optimized in the entire input space, a meta-criterion is only optimized within the set of candidates suggested by its base strategies.

The earlier approach of Hoffman et al. is based on a modification of the well-known Hedge algorithm [12], designed for the full-information adversarial multi-armed bandit. This particular portfolio algorithm relies on using the past performance of each acquisition function to predict future performance, where performance is measured by the objective function. However, this performance metric does not account for valuable information that is gained through exploration.

In Chapter 5, we introduce a new approached which we name Entropy Search Portfolio (ESP) that considers the use of an information-based metric instead [198]. In contrast to the GP-Hedge portfolio, ESP selects among different candidates by considering the gain of information towards the optimum.

We provide empirical evidence that our approach results in gains not only over previous portfolio strategies but also over the fundamental strategies that make up the portfolio.

1.6 Practical considerations

In this section, we discuss some implementation details and more advanced topics. In particular, we first describe how the unknown hyper-parameters θ are dealt with, we then provide a survey of techniques used to optimize the acquisition functions, followed by a discussion of non-stationarity and Bayesian optimization with parallelizable queries.

1.6.1 Handling hyper-parameters

Thus far in the discussion we have mostly ignored the kernel hyper-parameters and assumed they were given. In this section we describe two data-driven ways of handling hyper-parameters, namely point estimation and approximate marginalization. Consider a generic function $\alpha : \mathcal{X} \times \Theta \mapsto \mathbb{R}$, where $\theta \in \Theta$ represents the hyper-parameters of our GP. In the context of Bayesian optimization, this function could be our objective function or any function derived from the Gaussian process, but for concreteness, it may help to think of it specifically as the acquisition function, hence the symbol α. We wish to marginalize out our uncertainty about θ with the following expression

$$\alpha_t(x) := \mathbb{E}_{\theta|\mathcal{D}_t}\left[\alpha(x;\theta)\right] = \int \alpha(x;\theta)p(\theta|\mathcal{D}_t)\mathrm{d}\theta. \tag{1.50}$$

This integral is over our posterior belief over θ given observations $\mathcal{D}_t$, which can be decomposed via Bayes' rule as

$$p(\theta|\mathcal{D}_t) = \frac{p(\mathbf{y}|\mathbf{X},\theta)p(\theta)}{p(\mathcal{D}_t)}. \tag{1.51}$$

The simplest approach to tackling (1.50) is to fit the hyper-parameter to observed data using a point estimate $\hat{\theta}_t^{\mathrm{ML}}$ or $\hat{\theta}_t^{\mathrm{MAP}}$, corresponding to type II maximum likelihood or maximum *a posteriori* estimates, respectively. The posterior is then replaced by a delta measure at the corresponding $\hat{\theta}_t$ which yields

$$\hat{\alpha}_t(x) = \alpha(x;\hat{\theta}_t). \tag{1.52}$$

The estimators $\hat{\theta}_t^{\mathrm{ML}}$ and $\hat{\theta}_t^{\mathrm{MAP}}$ can be obtained by optimizing the marginal likelihood or the unnormalized posterior, respectively. For certain priors and likelihoods, these quantities as well as their gradients can be computed analytically. For example, the GP regression model yields the following marginal likelihood defined in (1.37), which we denote here by $\mathcal{L}_t$.

Therefore it is common to use multi-started quasi-Newton hill-climbers (e.g., the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method) on objectives such as the likelihood $\mathcal{L}_t$ or the unnormalized posterior.

In Bayesian optimization, our uncertainty about the response surface plays a key role in guiding exploration and therefore it is important to incorporate our uncertainty about θ in the regression model. Naturally, these point estimates cannot capture this uncertainty. For this reason we consider marginalizing out the hyper-parameters using either quadrature or Monte Carlo [166, 33, 202].

The common component in Monte Carlo (MC) methods is that they approximate the integral in (1.50) using M samples $\left\{\theta_t^{(i)}\right\}_{i=1}^{M}$ from the posterior distribution $p(\theta|\mathcal{D}_t)$:

$$\mathbb{E}_{\theta|\mathcal{D}_t}\left[\alpha(x;\theta)\right] \approx \frac{1}{M}\sum_{i=1}^{M}\alpha(x;\theta_t^{(i)}). \tag{1.53}$$

However, in practice it is impossible to sample directly from the posterior so Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) techniques are used to produce a sequence of samples that are marginally distributed according to $p(\theta|\mathcal{D}_t)$ in the limit of infinitely long chains. Once the M hyper-parameter samples are obtained, the acquisition function is evaluated and averaged over all samples; this marginal acquisition function incorporates the uncertainty in θ. In addition to MC methods, one could also use quadrature as shown in [166]. Here, samples (not necessarily drawn from the posterior) are combined using a weighted mixture:

$$\mathbb{E}_{\theta|\mathcal{D}_t}\left[\alpha(x;\theta)\right] \approx \sum_{i=1}^{M}\omega_i\alpha(x;\theta_t^{(i)}). \tag{1.54}$$

We could do away with samples entirely and approximately integrate out the hyper-parameters as shown in [70]. To make the integral tractable, the authors adopted a linear approximation to the likelihood which enables them to derive an approximate posterior. This method, however, has not been demonstrated in the setting of Bayesian optimization.

Estimating the hyper-parameters of GP kernels with very few function evaluations is a challenging task, often with disastrous consequences as illustrated by a simple example in [19]. The typical estimation of the hyper-parameters by maximizing the marginal likelihood [176, 113] can easily fall into traps, as shown in [39]. In Chapter 3, we discuss this topic further and illustrate the potential pitfalls of popular hyper-parameter estimation methods.

1.6.2 Optimizing acquisition functions

A central step of the Bayesian optimization framework is the maximization of the acquisition function. Naturally, an acquisition function is only useful if it is cheap to evaluate relative to the objective function f. Nevertheless, the acquisition function is often multimodal and maximizing it is not a trivial task. In practice, the community has resorted to using various techniques such as discretization [202] and adaptive grids [16], or similarly, the divided rectangles approach of [111], which was used in [35, 148, 143]. When gradients are available, or can be cheaply approximated, one can use a multi-started quasi-Newton

hill-climbing approach [138, 202]. Alternatively, [20] and [226] use the CMA-ES method of [87], and [106] apply multi-start local search.

Unfortunately, these auxiliary optimization techniques can be problematic for several reasons. First, in practice it is difficult to assess whether the auxiliary optimizer has found the global maximizer of the acquisition function. This raises important concerns about the convergence of Bayesian optimization algorithms because theoretical guarantees are only valid with the assumption that the exact optimizer is found and selected; see for example [205, 216] and [39]. Second, between any two consecutive iterations of the Bayesian optimization algorithm, the acquisition function may not change dramatically. Therefore, rerunning the auxiliary optimizer can be unnecessarily wasteful.

Recent proposed optimistic optimization methods provide an alternative to Bayesian optimization [121, 38, 158]. These methods sequentially build space-partitioning trees by splitting leaves with high function values or upper confidence bounds; the objective function is then evaluated at the center of the chosen leaves. Simultaneous optimistic optimization (SOO) can reach the global optimum without knowledge of the function's smoothness [158].

Though these optimistic optimization methods do not require any auxiliary optimization, these methods are not as competitive as Bayesian optimization in practical domains where prior knowledge is available. In Chapter 2, we introduce Bayesian multi-scale SOO algorithm that combines the tree partitioning idea of SOO with the surrogate model of Bayesian optimization to achieve competitive results while eliminating the need for auxiliary optimization.

1.6.3 Conditional Spaces

It is often the case that some variables will only influence the function being optimized when other variables take on certain values. These are called conditional variables and are said to be *active* or *inactive*. For example, when the function involves selecting between different algorithms as well as optimizing their hyper-parameters, then certain sets of hyper-parameters belonging to a given algorithm will be inactive if that algorithm is not selected [106, 20].

More formally, consider a variable $x_1 \in \mathcal{X}_2$ and another variable $x_2 \in \mathcal{X}_2$. x_1 is said to be a child of x_2 if it is only active when x_2 takes on certain values in $\mathcal{X}_2$. This conditional structure can be extended with multiple variables to form more complicated tree or directed acyclic graph structures. This greatly extends the capabilities of the Bayesian optimization framework, allowing it to chain together individual algorithms to form sophisticated pipelines that can be jointly optimized [211, 23].

Models such as random forests or the tree Parzen estimator (TPE) are naturally tailored to handle conditional spaces. Random forests are constructed using ensembles of decision trees that can learn to ignore inactive variables and the TPE itself is a graph-structured generative model that follows the conditional structure of the search space.

Gaussian processes are not immediately suitable for conditional spaces because standard kernels are not defined over variable-length spaces. A simple approach is to define a separate GP for each group of jointly active hyper-parameters [20], however this ignores dependencies between groups. Recent work has focused on defining a fixed-length embedding of conditional spaces where a standard kernel using Euclidean distance can be applied [207]. This is currently a very new area of research and more work needs to be done before GPs can work in conditional spaces as well as tree-based models.

1.6.4 Non-stationarity

A major assumption made by GP regression using the kernels suggested in Section 1.4.2 is that the underlying process is *stationary*. Formally, this assumption means that the kernel $k(\mathbf{x}, \mathbf{x}')$ can be equivalently written as a function of $\mathbf{x} - \mathbf{x}'$. Intuitively, a function whose length-scale does not change throughout the input space will be well modelled by a GP with a stationary kernel.

In real world problems we often expect that the true underlying process will be non-stationary. In these cases, the GP prior is misspecified, which means that it will require more data in order to produce reasonable posterior estimates. For Bayesian optimization this is an issue, as the entire goal is to minimize the function in as few evaluations as possible. Here, we will discuss some of the ways in which Bayesian optimization can be modified to deal with non-stationarity.

Non-stationary kernels One way to create a non-stationary process is to use a non-stationary kernel. One strategy is to convert a stationary kernel into a non-stationary one by transforming $\mathbf{x}$ using a parametric warping function $\mathbf{x}^{(w)} = w(\mathbf{x})$ and then applying a stationary kernel to $\mathbf{x}^{(w)}$ [186, 184]. If w is chosen appropriately, the data will follow a stationary process in the transformed space.

In Bayesian optimization, the inputs are traditionally projected onto the unit hypercube and this fact was exploited in [184], who chose the warping function to be the cumulative distribution function (CDF) of the beta distribution,

$$w_d(\mathbf{x}) = \frac{\mathbf{x}_d^{\alpha-1}(1-\mathbf{x}_d)^{\beta-1}}{B(\alpha,\beta)}, \tag{1.55}$$

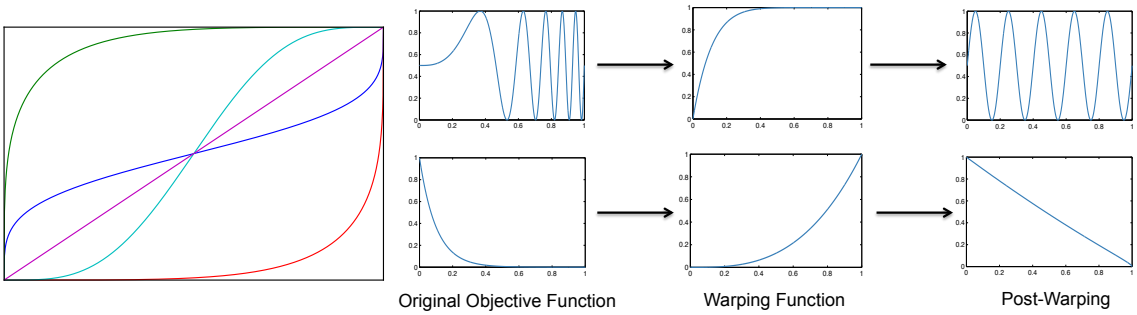


Figure 1.6: **Left:** Examples of beta CDF warpings under different settings of the shape parameters α and β. **Right:** Examples of functions after applying a beta CDF warping (originally from [184]). The regions where the CDF has a slope greater than 1 are expanded along the horizontal axis, while regions where the CDF has slope less than 1 are contracted.

where α and β are the shape parameters, and the B is the beta function. In this case, $w_d(\mathbf{x})$ is a warping function for the d^{th} dimension of $\mathbf{x}$, and a separate warping is applied to each dimension.

Examples of functions before and after applying beta warping are shown in Figure 1.6. Despite having only two parameters, the beta CDF is able to express a wide variety of transformations. These transformations contract portions of the input space, and expand others, which has the effect of decreasing and increasing the length scale in those portions, respectively. The beta warping approach has been shown to be highly effective on several benchmark problems as well as hyper-parameter optimization for machine learning models [184, 22].

While the beta CDF is not the only choice, it is appealing for a number of reasons. For hyper-parameter optimization, it mimics the kind of transformations practitioners tend to apply when applying a grid search, such as searching over learning rates in the log-domain. It is compactly parameterized, so that learning the shape parameters is not too much more expensive than learning other kernel parameters. Finally, it is an invertible transformation so that once the maximum of the acquisition function is found, it can easily be mapped back into the original space. For $\alpha = \beta = 1$, the transformation is the identity and the original, stationary GP is recovered.

Learning α and β via point estimates can be difficult when using gradient based optimization as the beta function and its derivatives with respect to α and β do not have simple closed-form solutions. An appealing alternative in this case is the Kumaraswamy distribution, whose CDF takes the form

$$w_d(\mathbf{x}) = 1 - (1 - \mathbf{x}_d^{\alpha})^{\beta}. \tag{1.56}$$

There are many other examples of non-stationary covariance functions [93, 167, 190, 176, 160, 29, 3, 71] that have been proposed for GP regression along with closely related output warping techniques [201] that can also model certain kinds of non-stationary processes.

Partitioning An alternative approach to modelling non-stationarity that has been useful in practice is to partition the space into distinct regions and then to model each region as a separate stationary process. In a random forest model [106, 34], this is achieved by finer partitioning in regions of the space where the function changes rapidly, and more granular partitioning in regions where the function changes slowly. Partitioning can also be an effective strategy for GPs. For example, [82] proposed the treed GP model, which partitions the data and then applies a separate GP to each region.

Heteroscedasticity Heteroscedasticity is a close analogue of non-stationarity, but refers to non-stationary behavior in the *noise* process governing the observation model, instead of the true process that we wish to capture. Standard GP regression using an isotropic noise kernel assumes by default that the noise process is constant everywhere, and is therefore stationary by definition. In practice, it is possible to have non-stationarity in both the true process and the noise process. Heteroscedasticity has been widely addressed in the GP literature, see e.g., [131, 117, 141].

For Bayesian optimization in particular, one approach to handling heteroscedastic noise was proposed in [7] using a partitioning approach. The idea is to build a partition using classification and regression trees (CART) [32]; however, splitting was restricted to occur at data points rather than between them. This ensured that the variance estimates of the GP would remain smooth between partitions.

Another form of non-stationarity that is closely related to heteroscedasticity is a non-stationary amplitude [1, 71]. This is where the magnitude of the output process changes as a function of the input. To our knowledge this has not been directly addressed in the Bayesian optimization literature. There have however been attempts to be robust to this effect by integrating out the amplitude parameter of the GP kernel. This was done numerically in [202] and analytically using conjugate priors in [196], resulting in a latent GP with t-distributed predictions and an input-dependent noise covariance.

1.6.5 Parallelization

Bayesian optimization is conventionally posed as a sequential problem where each experiment is completed before a new one is proposed. In practice it may be possible and

advantageous to run multiple function evaluations in parallel. Even if the number of experiments required to reach the minimum does not change, parallel approaches can yield a substantial reduction in terms of wall-clock time [108, 202].

Ginsbourger et al. [76] proposed several approaches based on imputing the results of currently running experiments. The idea is that given the current observations $\mathcal{D}_t = \{(x_t, y_t)\}$ and pending experiments $\mathcal{D}_p = \{x_p\}$, one can impute a set of set of experimental outcomes $\tilde{\mathcal{D}}_p = \{(x_p, \tilde{y}_p)\}$ and then perform a step of Bayesian optimization using the augmented dataset $\mathcal{D}_t \cup \tilde{\mathcal{D}}_p$.

One simple strategy is the *constant liar*, where a constant L is chosen such that $\tilde{y}_p = L,\ \forall p$. Another strategy is the *Kriging believer*, which uses the GP predictive mean $\tilde{y}_p = \mu_t(x_p)$. [202] used an approach where a set of S fantasies are sampled for each unfinished experiment from the full GP posterior predictive distribution. These are then combined to estimate the following parallel integrated acquisition function,

$$\alpha(x; \mathcal{D}_t, \mathcal{D}_p) = \int_{\mathbb{R}^J} \alpha(x; \mathcal{D}_t \cup \tilde{\mathcal{D}}_p) P(\tilde{y}_{1:J}; \mathcal{D}_t) dy_{p1:J}, \tag{1.57}$$

$$\approx \frac{1}{S} \sum_{s=1}^{S} \alpha(x; \mathcal{D}_t \cup \tilde{\mathcal{D}}_p^{(s)}), \tag{1.58}$$

$$\tilde{\mathcal{D}}_p^{(s)} \sim P(\tilde{y}_{1:J}; \mathcal{D}_t), \tag{1.59}$$

where J is the number of currently pending experiments. This approach has been shown to be very effective in practice when α is chosen to be EI. Similar approaches are proposed in [75, 51] and a similar parallel extension to GP-UCB is proposed [58].

Although the imputation approaches deal with parallel experiments, the nature in which they propose candidates is still inherently sequential. A truly parallel approach would simultaneously propose a set of candidates. Jones [112] and Hutter et al. [108] proposed an approach based on GP-UCB where α_{UCB} is optimized using a range of β_t values, which produces a set of points that favour a range of exploration and exploitation.

1.7 Theory of Bayesian optimization

There exist a vast literature on the theoretical properties of bandit algorithms in general. Theoretical properties of Bayesian optimization, however, have only been established recently. In this section, we focus on the results concerning Gaussian process based Bayesian optimization and defer detailed discussions of bandit algorithms to other dedicated surveys [36, 159].

There exist several early consistency proofs for Gaussian process based Bayesian optimization algorithms, in the one-dimensional setting [140] and one for a simplification of the algorithm using simplicial partitioning in higher dimensions [233]. The consistency of the algorithm using multivariate Gaussian processes has been established in [217].

More recently, [205] provided the first finite sample bound for Gaussian process based Bayesian optimization. In this work, the authors showed that the GP-UCB algorithm suffers from sub-linear cumulative regret in the stochastic setting. The regret bounds, however, allow only fixed hyper-parameters. In [39], Bull provided both upper and lower bounds of simple regret for the EGO algorithm [113] in the deterministic setting. In addition to regret bounds concerning fixed hyper-parameters, the author also provided simple regret bounds while allowing varying hyper-parameters.

Since the pioneering work of [205] and [39], there emerged a large body of results on this topic including, exponentially vanishing simple regret bounds in the deterministic setting [55]; bounds for contextual Gaussian process bandits [123]; Bayes regret bounds for Thompson sampling [115, 183]; bounds for high-dimensional problems with a underlying low-rank structure [59]; bounds for parallel Bayesian optimization [58]; and improved regret bounds using mutual information [52].

Despite the recent surge in theoretical contributions, there is still a wide gap between theory and practice. Regret bounds or even consistency results, for example, have not been established for approaches that use a full Bayesian treatment of hyper-parameters [202]. Such theoretical results could advance the field of Bayesian optimization and provide insight for practitioners.

1.8 History of Bayesian optimization and related approaches

Arguably the earliest work related to Bayesian optimization was that of William Thompson in 1933 where he considered the likelihood that one unknown Bernoulli probability is greater than another given observational data [210]. In his article, Thompson argues that when considering, for example, two alternative medical treatments one should not eliminate the worst one based on a single clinical trial. Instead, he proposes, one should estimate the probability that one treatment is better than the other and weigh future trials in favour of the seemingly better treatment while still trying the seemingly suboptimal one. Thompson rightly argues that by adopting a single treatment following a clinical trial, there is a fixed chance that all subsequent patients will be given suboptimal treatment. In contrast, by dynamically selecting a fraction of patients for each treatment, this sacrifice becomes vanishingly small.

In modern terminology, Thompson was directly addressing the *exploration–exploitation* trade-off, referring to the tension between selecting the best known treatment for every future patient (the greedy strategy) and continuing the clinical trial for longer in order to more confidently assess the quality of both treatments. This is a recurring theme not only in the Bayesian optimization literature, but also the related fields of sequential experimental design, multi-armed bandits, and operations research.

Although modern experimental design had been developed a decade earlier by Ronald Fisher's work on agricultural crops, Thompson introduced the idea of making design choices dynamically as new evidence becomes available; a general strategy known as sequential experimental design or, in the multi-armed bandit literature, adaptive or dynamic allocation rules [128, 78].

The term Bayesian optimization was coined in the seventies [156], but a popular version of the method has been known as efficient global optimization in the experimental design literature since the nineties [192]. Since the approximation of the objective function is often obtained using Gaussian process priors, the technique is also referred to as Gaussian process bandits [205].

In the nonparametric setting, Kushner [127] used Wiener processes for unconstrained one-dimensional optimization problems. Kushner's decision model was based on maximizing the probability of improvement. He also included a parameter that controlled the trade-off between 'more global' and 'more local' optimization, in the same spirit as the exploration–exploitation trade-off. Meanwhile, in the former Soviet Union, Močkus and colleagues developed a multidimensional Bayesian optimization method using linear combinations of Wiener fields [156, 154]. Both of these methods, probility of and expected improvement, were in studied in detail in [112].

At the same time, a large, related body of work emerged under the name *kriging*, in honour of the South African student who developed this technique at the University of the Witwatersrand [124], though largely popularized by Matheron and colleagues (e.g., [149]). In kriging, the goal is interpolation of a random field via a linear predictor. The errors on this model are typically assumed to *not* be independent, and are modelled with a Gaussian process.

Kriging has been applied to experimental design under the name DACE, after design and analysis of computer experiments, the title of a paper by Sacks et al. [185] (and more recently a book by Santner et al. [188]). In DACE, the regression model is a best linear unbiased predictor (BLUP), and the residual model is a noise-free Gaussian process. The goal is to find a design point or points that optimizes some criterion. Experimental design is

usually non-adaptive: the entire experiment is designed before data is collected. However, sequential design is an important and active subfield (e.g., [228, 40].

The efficient global optimization (EGO) algorithm is the combination of DACE model with the sequential expected improvement acquisition criterion. It was published in a paper by Jones et al. [113] as a refinement of the SPACE algorithm (stochastic process analysis of computer experiments) [191]. Since EGO's publication, there has evolved a body of work devoted to extending the algorithm, particularly in adding constraints to the optimization problem [9, 189, 30], and in modelling noisy functions [17, 101, 102].

In the bandits setting, Lai and Robbins [128] introduced upper confidence bounds (UCB) as approximate alternatives to Gittins indices in 1985. Auer studied these bounds using frequentist techniques, and in adversarial multi-armed bandit settings [12, 11].

The literature on multi-armed bandits is vast. The book of Cesa-Bianchi [44] is a good reference on the topic of online learning with experts and bandits in adversarial settings. There are many results on exploration [37, 68, 67] and contextual bandits [133, 150, 2]. These contextual bandits, may also be seen as myopic approximations to Markov decision processes.

1.9 Extensions and Open Questions

1.9.1 Constrained Bayesian optimization

In [73] a scenario was outlined in which a food company wished to design the best tasting cookie subject to the number of calories being below a certain level. This is an example of constrained optimization, where certain regions of the design space $\mathcal{X}$ are invalid. In machine learning, this can arise when certain hyper-parameter configurations result in models that diverge during training, or that run out of computer memory. When the constraints are known *a priori*, they can be incorporated into the optimization of the acquisition function. The more challenging case arises when it is not known in advance which configurations will result in a constraint violation. Several approaches deal with this problem by altering the acquisition function itself.

Gramacy and Lee [83] proposed the integrated expected conditional improvement (IECI) acquisition function:

$$\alpha_{\mathrm{IECI}}(\mathbf{x}) = \int_{\mathbf{x}'} (\alpha_{\mathrm{EI}}(\mathbf{x}', \mathcal{D}_t) - \alpha_{\mathrm{EI}}(\mathbf{x}', \mathcal{D}_t \cup \mathbf{x})|\mathbf{x}))h(\mathbf{x}')d\mathbf{x}. \tag{1.60}$$

This gives the change in expected improvement from observing $\mathbf{x}$ under the density h. Choosing h to model the probability of satisfying the constraint encourages IECI to favor regions with a high probability of being valid.

Snoek [204] and Gelbart et al. [73] proposed the weighted expected improvement criterion (wEI) that multiplies EI by the probability of satisfying the constraints:

$$\alpha_{\text{wEI}}(\mathbf{x}) = \alpha_{\text{EI}}(\mathbf{x}, \mathcal{D}_t)h(\mathbf{x}, \mathcal{D}_t). \tag{1.61}$$

Where $h(\mathbf{x}, \mathcal{D}_t)$ is a Gaussian process with a Bernoulli observation model. This reduces EI in regions that are likely to violate constraints.

A variant of wEI was proposed in [69] to deal with the case where the function is constrained to be less than some value λ. They used $h(\mathbf{x}, \mathcal{D}_t) = \mathbb{P}(f(x) < \lambda \mid \mathcal{D}_t)$, the posterior probability of satisfying this constraint under the Gaussian process model of the function.

Hernández-Lobato et al. [91] recently proposed a variation of the predictive entropy search acquisition function to deal with the decoupled case, where the function and constraints can be evaluated independently.

In a different approach, Gramacy et al. [81] adapted the augmented Lagrangian approach to the Bayesian optimization setting, with unconstrained Bayesian optimization approximately solving the inner loop of the algorithm.

1.9.2 Cost-sensitivity

In some cases, each function evaluation may return both a value along with an associated cost. In other words, it may be more expensive to evaluate the function in some parts of the design space than others. If there is a limited budget, then the search should be biased toward low-cost areas. In [202], the goal was to train a machine learning model and the cost was the time it took to train the model. They used expected improvement per second, $\text{EI}(\mathbf{x}, \mathcal{D}_t)/c(\mathbf{x})$ in order to bias the search toward good models with fast training times. Here, $c(\mathbf{x})$ was the estimated cost of querying the objective at $\mathbf{x}$ and was modelled using a Gaussian process with response $\log(c(\mathbf{x}))$.

1.9.3 High-dimensional problems

Despite many success stories, Bayesian optimization is restricted to problems of moderate dimension. To advance the state of the art, Bayesian optimization should be scaled to high-dimensional parameter spaces. This is a difficult problem: to ensure that a global optimum is found, we require good coverage of $\mathcal{X}$, but as the dimensionality increases, the number of evaluations needed to cover $\mathcal{X}$ increases exponentially.

For *linear* bandits, Carpentier et al. [43] recently proposed a compressed sensing strategy to attack problems with a high degree of sparsity. Also recently, Chen et al. [47] made

significant progress by introducing a two stage strategy for optimization and variable selection of high-dimensional GPs. In the first stage, sequential likelihood ratio tests with a couple of tuning parameters are used to select the relevant dimensions. This, however, requires the relevant dimensions to be axis-aligned with an ARD kernel. Chen et al. provide empirical results only for synthetic examples (of up to 400 dimensions), but they provide key theoretical guarantees.

Hutter et al. [106] used Bayesian optimization with random forests based on frequentist uncertainty estimates. Their method does not have theoretical guarantees for continuous optimization, but it achieved state-of-the-art performance for tuning up to 76 parameters of algorithms for solving combinatorial problems. Note that in constructing the trees that make the forest, one samples and selects the most promising features (dimensions). That is, random forests naturally select the relevant dimensions of the problem, and so not surprisingly have worked well in practice.

Many researchers have noted that for certain classes of problems most dimensions do not change the objective function significantly; examples include hyper-parameter optimization for neural networks and deep belief networks [21] and automatic configuration of state-of-the-art algorithms for solving $\mathcal{NP}$-hard problems [103]. That is to say these problems have low *effective dimensionality*. To take advantage of this property, Bergstra and Bengio [21] proposed to simply use random search for optimization – the rationale being that points sampled uniformly at random in each dimension can densely cover each low-dimensional subspace. As such, random search can exploit low effective dimensionality *without knowing which dimensions are important*. In Chapter 4, we exploit the same property, while still capitalizing on the strengths of Bayesian optimization. By combining randomization with Bayesian optimization, we are able to derive a new approach that outperforms each of the individual components.

1.9.4 Multi-Task

When tuning the hyper-parameters of a machine learning model on some data, it is unlikely that the hyper-parameters will change very much if new data is added to the original data, especially if the new data represents a small fraction of the total amount. Likewise, if one were to train a model for object recognition, then good hyper-parameter settings are likely to also be good on other object recognition datasets. Experts often exploit this property when applying their models to new datasets.

There have been several attempts to exploit this property within the Bayesian optimization framework [123, 106, 15, 208, 229, 66]. The idea is that there are several correlated functions, $\mathcal{T} = \{1, 2, \ldots, M\}$, called *tasks* and that we are interested in optimizing some

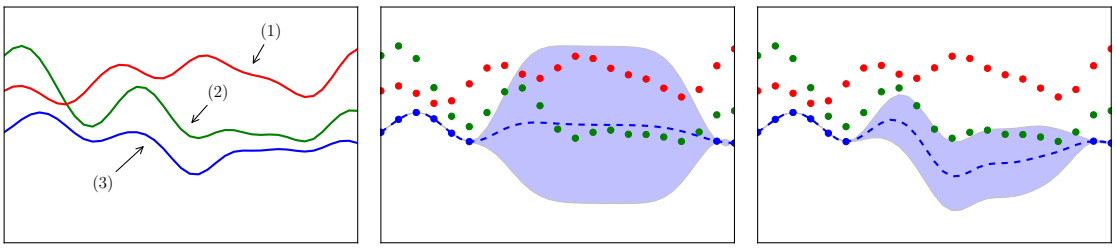


Figure 1.7: **Left:** Three correlated functions drawn from a multi-output GP. **Middle:** The GP posterior predictive distribution of function (3) when the functions are assumed to be independent. This is equivalent to ignoring observations from functions (1) and (2). **Right:** Posterior predictive distribution of function (3) when the correlations are taken into account. Here, the observations from functions (1) and (2) act as weak observations for function (3). This results in a much more accurate prediction.

subset of these tasks. In essence, the data from one task can provide information about another task.

One way to share information between tasks in a Bayesian optimization routine is to modify the underlying Gaussian process model. There has been a great deal of work on extending Gaussian processes to the multi-task scenario. These extensions are also known as multi-output Gaussian processes. The key is to define a valid covariance over input and task pairs, $k((\mathbf{x}, m), (\mathbf{x}', m'))$. One method is to use the intrinsic model of coregionalization (ICM) [80, 194, 28] that utilizes the product kernel,

$$k((\mathbf{x}, m), (\mathbf{x}', m')) = k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')k_{\mathcal{T}}(m, m'). \tag{1.62}$$

Where $m,\ m' \in \mathcal{T}$. $k_{\mathcal{T}}$ defines the covariance between tasks. There are many ways to parameterize the task covariance function [171]. Figure 1.7 illustrates how knowledge of correlations between tasks can be used with a multi-output GP to make more accurate predictions.

An alternative view of the ICM model is that it defines a latent process that is rotated and scaled to produce each of the individual tasks. The problem of defining a multi-output GP can then be viewed as learning a latent function, or a set of latent functions, that can be transformed to produce the output tasks. [15] proposed an approach that learns a latent ranking function at each iteration using pairs of observations from within each task. By learning a single ranking function that works across tasks, the tasks are effectively jointly embedded in a latent space that is invariant to potentially different output scales across tasks.

Each task may come with additional side information, or *context* features. In this case, it is possible to define a joint model that uses this context. This was considered for algorithm

configuration in [106] using a random forest model. When starting a new task, [66] uses task features to find similar tasks. The best inputs from the most similar tasks are then used as the initial design for the new task.

1.9.5 Freeze-Thaw

In some cases, the experiments selected by Bayesian optimization themselves require an inner loop of iterative optimization. For example, in the case of tuning machine learning hyper-parameters, each experiment consists of training a model before evaluating it. It is often possible to evaluate the model during training in order to get an estimate of how it is performing. When tuning hyper-parameters by hand, experts can use this information in order to estimate model performance at the end of training and can halt training early if this estimate looks unsatisfactory. This allows a far greater number of models to be trained in a given amount of time.

An attempt to incorporate this into the Bayesian optimization framework is given in [209]. They identify that many loss functions in machine learning follow an exponential decay pattern during training, and construct a basis set of exponentially decaying functions of the form $f(t, \lambda) = e^{-\lambda t}$, where λ represents the rate of decay over time, represented by t, in order to forecast model performance. It is possible to construct a nonstationary kernel from this basis set:

$$k(t, t') = \frac{\beta^\alpha}{(t + t' + \beta)^\alpha}, \tag{1.63}$$

where α and β are hyper-parameters that control the shape of the kernel. This kernel is used within a Gaussian process to jointly model $(\mathbf{x}, t)$ pairs. Given the ability to forecast curves, [209] then uses an entropy search-based acquisition function in order to determine whether to *freeze* a currently running experiment, *thaw* a previous experiment in order to resume training, or start a new experiment.

Rather than constructing a kernel, [61, 60] built a basis set manually based on previously collected training curves. This basis set is then used with Bayesian linear regression in order to forecast training curves, and an early stopping rule is given based on the probability of improvement using the forecasted value.

An alternative view of this procedure is to consider Gaussian process models that incorporate partial feedback. This view is used in [170], where they construct a Gaussian process with non-stationary noise process that starts high when the experiment begins, and decays over time.

Recently, [134] has proposed a very different approach (named HyperBand) to attack the same problem. In the paper, the authors adopt random search in combination with an

adaptive version of SuccessiveHalving (which is similar to Beam Search). Random search is used to generate hyper-parameter candidates. The adaptive SuccessiveHalving in turn weeds out the candidates that are not performing favorably. HyperBand has been shown to be very effective on test problems the authors considered and also holds the advantage of not using a kernel that models training curves which could be hard to design.

1.10 Outline of this work and contributions

In this work, we deal primarily with Gaussian process based Bayesian optimization. The introduction chapter is based on our tutorial paper which is joint work with Bobak Shahriari, Kevin Swersky, Nando de Freitas and Ryan Adams [197]. Bobak, especially, contributed much to the shaping of the tutorial paper and therefore the chapter. The remainder of the thesis is formed mostly of my own work and is divided into six chapters. The last chapter concludes the work. In this section, we briefly outline the rest of the chapters.

As discussed in Section 1.6.2, the optimization of acquisition functions plays a very important role in Bayesian optimization. Not only does the theoretical guarantees depends on the optimal optimization of the acquisition functions, but also, poor optimization of which often leads to bad practical performance. In Chapter 2, we introduce, the Bayesian multi-scale SOO (BaMSOO) algorithm which combines the tree partitioning idea of SOO [158] with the surrogate model of Bayesian optimization, eliminating the need for auxiliary optimization. We also provide theoretical guarantees for BaMSOO which do not require on the exact optimization of an acquisition function. Intuitively, the method implements SOO to optimize the objective function directly, but avoids querying points that are deemed unlikely to be optimal by the surrogate model's confidence bounds. This chapter is based off work previously published in [222].

In Chapter 3, we tackle the problem of convergence of Bayesian optimization with stochastic observations while allowing the kernel hyper-parameters to vary. Prior to this work, the work of Bull [39] provides the only known regret bound for Bayesian optimization when the hyper-parameters are unknown. By introducing lower and upper bounds on the possible range of hyper-parameters values, Bull obtains convergence rates for deterministic objective functions. Bull's results, however, is limited to the case of deterministic observations. Here, we extend the work of Bull to stochastic objective functions. Our results apply to sub-Gaussian noise, *e.g.*, symmetric Gaussian, Bernoulli, or uniform noise. To the best of our knowledge, this work proves the first convergence results for the EI family of acquisition functions.

Classical formulations of Bayesian optimization do not scale to high-dimensional problems. As discussed in Section 1.9.3, however, for a certain class of problems, most dimensions do not change the objective function significantly. Many important problems fall into this class including hyper-parameter optimization for neural networks and deep belief networks [21] as well as automatic configuration of state-of-the-art algorithms for solving $\mathcal{NP}$-hard problems [103]. In Chapter 4, we introduce a Bayesian optimization algorithm (which we name REMBO) that is designed to capitalize on this class of problems. By combining randomization with Bayesian optimization, we derive a new approach that does not depend on the extrinsic dimensionality of the problem and, given the right function class, is capable of optimizing in Billions of dimensions. We show both theoretical and empirical results to back up our claim. This chapter is based on the work of [223] as well as [226].

Chapter 5 focuses on the problem of portfolios of acquisition functions. Our proposed approach, entropy search portfolio (ESP), considers the use of an information-based metric. In contrast to the GP-Hedge portfolio [97], ESP selects among different candidates by considering the gain of information about the location of the optimum. To estimate the entropy of the probability distribution of optimal query points, we introduce a novel way of obtaining discrete search points by sampling directly from the probability distribution of a point being the minimizer following Bochner's Lemma [174]. We also provide empirical evidence that our approach results in gains not only over previous portfolio strategies but also over the fundamental strategies that make up the portfolio. This chapter is based on [198].

Finally, Chapter 6 introduces an application of Bayesian optimization to tuning (Riemannian manifold) Hamiltonian Monte Carlo ((RM)HMC) samplers. HMC is one of the most powerful and efficient class of MCMC samplers. It is, however, notoriously hard to tune. In this chapter, we first introduce the HMC samplers (as well as their Riemannian manifold counterparts) and discuss their properties. We then proceed to introduce a novel Adaptive HMC (AHMC) sampler which makes use of Bayesian optimization to automatically select hyper-parameters for HMC. Finally, we show state-of-the-art performance of AHMC on a number of test problems. We also provide a convergence analysis of the proposed sampler. This chapter is based on the work introduced in [224] and [232].

Chapter 2

Bayesian Multi-scale Optimistic optimization

In this chapter, we consider the problem of approximating the maximizer of a deterministic black-box function $f : \mathcal{X} \mapsto \mathbb{R}$. The function f can be evaluated point-wise, but it is assumed to be expensive to evaluate.

This global optimization problem can be treated within the framework of sequential design. In this context, by allowing $\mathbf{x}_t \in \mathcal{X}$ to depend on previous points and corresponding function evaluations $\mathcal{D}_{t-1} = \{(\mathbf{x}_1, f(\mathbf{x}_1)), \ldots, (\mathbf{x}_{t-1}, f(\mathbf{x}_{t-1}))\}$, the algorithm constructs a sequence $\mathbf{x}_{1:n} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ and returns the element $\mathbf{x}(n)$ of highest possible value. That is, it returns the value $\mathbf{x}(n)$ that minimizes the loss:

$$r_n = \sup_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) - f(\mathbf{x}(n)).$$

This measure is similar to *simple regret* in the bandit literature and is not to be confused with the concept of *cumulative regret* (see Section 1.3.4.2).

In this chapter, we consider solving the black-box optimization problem using Bayesian optimization. A central step of the Bayesian optimization framework is the maximization of the acquisition function (see Section 1.5 and 1.6.2). Naturally, an acquisition function is only useful if it is cheap to evaluate relative to the objective function f. Nevertheless, the acquisition function is often multimodal and maximizing it is not a trivial task. The maximum of the acquisition function is typically found by resorting to discretization or by means of an *auxiliary optimizer*. For example, [202] use discretization, [16] use adaptive grids, [35, 148] and [143] use the DIRECT algorithm of [111], [138, 202, 184] use a combination of discretization and quasi-Newton hill-climbing, [20] and [226] use the CMA-ES method of [87], [106] apply multi-start local search. (Approaches within the framework of Bayesian nonlinear experimental design, such as [90] for finding maxima and [125, 126, 96] for learning functions and Markov decision processes, have to rely on

expensive approximate inference for computing intractable integrals. An analysis of these approaches is beyond the scope of this work.)

The auxiliary optimization methodology is problematic for several reasons. First, it is difficult to assess whether the auxiliary optimizer has found the maximum of the acquisition function in practice. This creates important theoretical concerns about the behavior of Bayesian optimization algorithms because the typical theoretical convergence guarantees are only valid on the assumption that the optimum of the acquisition function can be found exactly; see for example [205, 216] and [39]. Second, running an auxiliary optimizer at each iteration of the Bayesian optimization algorithm can be unnecessarily costly. For any two consecutive iterations, the acquisition function may not change drastically. This questions the necessity of re-starting the auxiliary optimization at each iteration. Last but not least, failures in the optimization of acquisition functions could lead to sub-optimal solutions in practice.

Recent *optimistic optimization* methods provide a viable alternative to Bayesian optimization [121, 38, 158]. Instead of estimating a posterior distribution over the unknown objective function, these methods build space partitioning trees by expanding leaves with high function values or upper-bounds. The term optimistic, in this context, is used to refer to the fact that the algorithms expand at each round leaves that may contain the optimum. Remarkably, a variant of these methods, *Simultaneous Optimistic Optimization* (SOO) by [158], is able to optimize an objective function globally without knowledge of the function's smoothness. SOO is optimistic at all scales in the sense that it expands several leaves simultaneously, with at most one leaf per level. For this reason, instead of adopting the term "Simultaneous OO" we opt for the descriptive term "Multi-Scale OO".

We will describe SOO in more detail in Section 2.2. We also note that a stochastic variant of SOO has been recently proposed by [213], but we restrict the focus of this chapter to the deterministic case.

These optimistic optimization methods do not require the auxiliary optimization of acquisition functions. However, due to the lack of a posterior that interpolates between the sampled points, it is conceivable that these methods may not be as competitive as Bayesian optimization in practical domains where prior knowledge is available. This claim does not seem to have been backed up by empirical evidence in the past.

This chapter introduces a new algorithm, BaMSOO, which combines elements of Bayesian optimization and SOO. Importantly, it eliminates the need for auxiliary optimization of the acquisition function in Bayesian optimization. We derive theoretical guarantees for the method that do not depend on the assumption that the acquisition function needs to be optimized exactly. The method uses SOO to optimize the objective function directly, but

eliminates the need for SOO to sample points that are deemed unfit by Gaussian process posterior bounds. That is, BaMSOO uses the posterior distribution to reduce the number of function evaluations in SOO, thus increasing the efficiency of SOO substantially.

The experiments with benchmarks from the global optimization literature demonstrate that BaMSOO outperforms both GP-UCB and SOO. The chapter also introduces a novel application in the domain of knowledge discovery and information extraction. Finally, our theoretical results show that BaMSOO can attain, up to log factors, a polynomial finite sample convergence rate.

2.1 Bayesian optimization with GP confidence bounds

In this chapter, we report results for the GP-UCB acquisition function (see Section 1.5). The upper confidence bound (UCB), in this setup, is defined to be $\mathcal{U}(\mathbf{x}|\mathcal{D}_t) = \mu(\mathbf{x}|\mathcal{D}_t) + \sqrt{B_t}\sigma(\mathbf{x}|\mathcal{D}_t)$ [205, 55]. We also make use of the lower confidence bound (LCB) which is defined as $\mathcal{L}(\mathbf{x}|\mathcal{D}_t) = \mu(\mathbf{x}|\mathcal{D}_t) - \sqrt{B_t}\sigma(\mathbf{x}|\mathcal{D}_t)$. In these definitions, B_t is such that $f(\mathbf{x})$ is bounded above and below by $\mathcal{U}(\mathbf{x}|\mathcal{D}_t)$ and $\mathcal{L}(\mathbf{x}|\mathcal{D}_t)$ with high probability [55].

GP-UCB selects the next query point by optimizing the acquisition function: $\alpha_{\text{UCB}}(\mathbf{x}|\mathcal{D}_t)$ which in this case is defined to be the upper confidence bound: $\mathcal{U}(\mathbf{x}|\mathcal{D}_t)$. Note that our choice of utility favors the selection of points with high variance (points in regions not well explored) and points with high mean value (points worth exploiting). As mentioned in the introduction, the optimization of the closed-form acquisition function is often carried out by off-the-shelf global optimization procedures, such as DIRECT and CMA-ES.

Finite sample bounds for GP-UCB were derived by [205]. However, the bounds depend on the algorithm being able to optimize the UCB acquisition function, at each iteration, exactly. Unless the action set is discrete, it is unlikely that we will be able to find the global optimum of the UCB with a fixed budget optimization method. That is, we may not be able to guarantee that we can find the exact optimum of the UCB, and hence the theoretical bounds seem to make a very strong assumption in this regard.

2.1.1 Shrinking feasible regions

[55] introduced a different GP-based scheme to trade off exploration and exploitation. Instead of optimizing the acquisition function, they proposed to sample the objective function using a finite lattice within a feasible region R. The feasible region at the t^{th} iteration is defined as

$$R_t = \{\mathbf{x} : \mu_t(\mathbf{x}) + B_t\sigma_t(\mathbf{x}) > \sup_{\mathbf{x}\in R_{t-1}} \mu_t(\mathbf{x}) - B_t\sigma_t(\mathbf{x})\}.$$

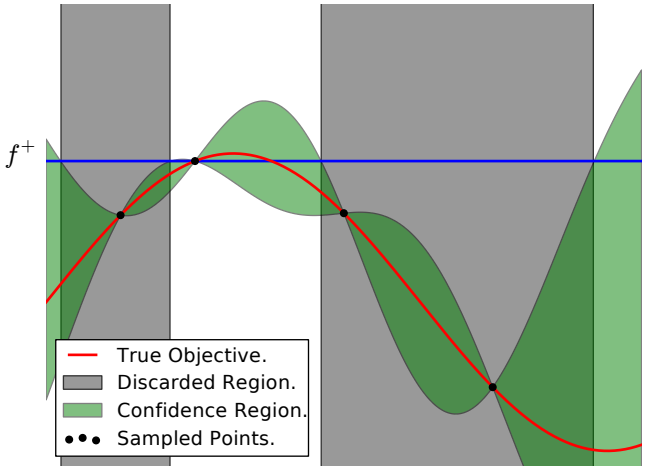


Figure 2.1: The global shrinking method of [55]. If the unknown objective function lies within the (green) confidence bounds with high probability, we can discard regions of the space where the upper bound is lower than the best lower bound encountered thus far.

That is, one should only search in the region where the upper bound is greater than the best lower bound encountered thus far, as illustrated in Figure 2.1. With high probability, the optimizer lies within R_t.

[55] proved that if we double the density of points in the lattice at each iteration, the feasible region shrinks very quickly. More precisely, they showed that the simple regret vanishes at an exponential rate and that the cumulative regret is bounded by a constant.

With this approach, they did not have to resort to optimizing an acquisition function. However, even in moderate dimensions, their algorithm is impractical since the lattice often becomes too large to be sampled in a reasonable amount of time.

In this chapter, we will argue that to overcome this problem, an optimistic strategy may have to be employed. Such a strategy enables us to sample the most promising regions first, so as to avoid the computational cost associated with covering the whole space. In the next section, we begin our discussion of optimistic strategies.

2.2 Simultaneous optimistic optimization

Deterministic optimistic optimization (DOO) and simultaneous optimistic optimization (SOO) are tree-based space partitioning methods for black-box function optimization [158, 159]. They were inspired by the UCT algorithm, which enjoyed great success in planning [121]. UCT was shown to have no finite-time guarantees by [53]. This prompted the

Algorithm 3 SOO

Evaluate $f(\mathbf{x}_{0,0})$
Initialize the tree $\mathcal{T}_1 = \{0, 0\}$
Set $n = 1$
while true **do**
  Set $\nu_{\max} = -\infty$
  for $h = 0 : \min\{\text{depth}(\mathcal{T}_n), h_{\max}(n)\}$ **do**
    Select $(h, j) = \arg\max_{j \in \{j | (h,j) \in L_n\}} f(\mathbf{x}_{h,j})$
    if $f(\mathbf{x}_{h,j}) > \nu_{\max}$ **then**
      Evaluate the children of (h, j)
      Add the children of (h, j) to $\mathcal{T}_n$
      Set $\nu_{\max} = f(\mathbf{x}_{h,j})$
      Set $n = n + 1$
    end if
  end for
end while

development of a range of optimistic, in the face of uncertainty, approaches. The term optimism, here, refers to the fact that the strategies expand at each round tree cells that may contain the optimum.

DOO and SOO partition the space $\mathcal{X}$ hierarchically by building a tree. Let us assume that each node of the tree has k children. A node (h, j) at level h of the tree has children $\{(h + 1, kj + i)\}_{0 \le i < k-1}$. The children partition the parent cell $X_{h,j}$ into cells $\{X_{h+1,kj+i},\ 0 \le i < k - 1\}$. The root cell is the entire space $\mathcal{X}$. A node is always evaluated at the center of the cell, which we denote as $\mathbf{x}_{h,j}$.

Instead of assuming that the target function is a sample from a GP, DOO and SOO assume the existence of a symmetric semi-metric ℓ such that $f(\mathbf{x}^*) - f(\mathbf{x}) \le \ell(\mathbf{x}, \mathbf{x}^*)$ where $\mathbf{x}^*$ is the maximizer of f. Although, SOO assumes that ℓ exists, it does not require explicit knowledge of it.

DOO on the other hand does require knowledge of ℓ. DOO builds a tree $\mathcal{T}_n$ incrementally, where n denotes the index over node expansions. DOO expands a leaf (h, j) from the set of leaves L_n (nodes whose children are not in $\mathcal{T}_n$) if it has the the highest upper bound: $f(\mathbf{x}_{h,j}) + \sup_{\mathbf{x} \in X_{h,j}} \ell(\mathbf{x}_{h,j}, \mathbf{x})$. This value for any cell containing $\mathbf{x}^*$ upper bounds the best function value f^*. The performance of DOO depends crucially on our knowledge of the true local smoothness of f. SOO aims to overcome the difficulty of having to know the true local smoothness.

SOO, as summarized in Algorithm 3, expands several leaves simultaneously. When a node is expanded, its children are evaluated. At each round, SOO expands at most one leaf per level, and a leaf is expanded only if it has the largest value among all leaves of the same or lower depths. The SOO algorithm takes as input a function $n \to h_{\max}(n)$, which

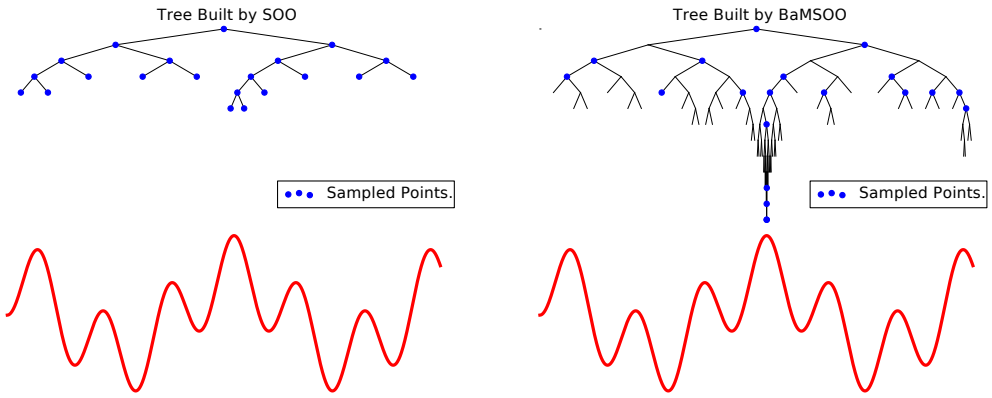


Figure 2.2: [LEFT]: The tree built by SOO when optimizing the function $f(x) = \frac{1}{2}\sin(15x)\sin(27x)$ in $[0, 1]$. [RIGHT]: The tree built by BaMSOO. The 20 blue dots represent nodes where the objective was evaluated. BaMSOO, in comparison, does not evaluate the objective function for points known to be sub-optimal with high probability. Hence, BaMSOO can achieve a better coverage of the search space with the same number of function evaluations as SOO.

limits the maximum height of the tree after n node expansions. $h_{\max}(n)$ defines a tradeoff between deep versus broad exploration. At the end of the finite horizon, SOO returns the $\mathbf{x}$ with the highest objective function value. Figure 2.2 illustrates the application of SOO to a simple 1-dimensional optimization problem.

2.3 BaMSOO

SOO offers a different way of trading off exploration and exploitation that does not require the optimization of an acquisition function. However, it does not utilize all the information brought in by the previously evaluated points effectively. To improve upon SOO in practice, we consider the additional assumption that the objective function is a sample from a GP prior.

We define the LCB and UCB to be $\mathcal{L}_N(\mathbf{x}|\mathcal{D}_t) = \mu(\mathbf{x}|\mathcal{D}_t) - B_N\sigma(\mathbf{x}|\mathcal{D}_t)$ and $\mathcal{U}_N(\mathbf{x}|\mathcal{D}_t) = \mu(\mathbf{x}|\mathcal{D}_t) + B_N\sigma(\mathbf{x}|\mathcal{D}_t)$ where $B_N = \sqrt{2\log(\pi^2 N^2/6\eta)}$ and $\eta \in (0, 1)$.

The BaMSOO algorithm is very similar to SOO. As with SOO, we only evaluate the cell at the center point. However, when a node's UCB is less than the function value of the best point already sampled, denoted f^+, we do not evaluate the objective function at this node because with high probability the center point is sub-optimal. Instead, we simply assign to this node its LCB value. Note that if the center-point of a cell is sub-optimal, the cell may still contain the optimizer. Hence this cell must also be further expanded in subsequent

iterations. To manage these two types of node in the pseudo-code (see Algorithm 4), we introduce a place-holder function g which is set to f when the UCB of the cell of interest is bigger than f^+, and it is set to the LCB of the node otherwise. For clarity, we remind the reader that the indices N, k, t and n are over node evaluations, branches (children), function evaluations and node expansions respectively.

In the pseudocode, we have highlighted in blue the additional lines of code brought in by BaMSOO. Effectively, BamSOO only involves a slight modification of SOO (Algorithm 2) provided we have GP routines to evaluate the LCB and UCB.

We found the assignment of the LCB values to nodes that do worse than f^+ to work well in practice. For this reason our presentation, experiments and theory focus on this choice.

BaMSOO improves upon SOO by making use of the available information more efficiently. Moreover, by using an optimistic proposal, it avoids the need to sample exhaustively before shrinking the feasible region as in [55]. Figure 2.2 illustrates how BaMSOO can cover the search space more effectively, even though it incurs the same number of expensive function evaluations as SOO.

2.4 Analysis

In this section, we provide an overview of the theoretical analysis of BaMSOO, which appears in Appendix A. Our discussion here will focus on our assumptions. At the end of this section, we will present the main result and sketch the proof coarsely.

We denote the global maximum by $f^* = \sup_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ and the maximizer by $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$.

We make similar assumptions to those made by [55]. As in their case, we make the global assumption that the objective function is a sample from a GP and a local assumption about the behavior of the objective near the optimum.

Assumption 1 (Conditions on the GP kernel)**.** *$\mathcal{X} \subseteq \mathbb{R}^D$ is a compact set, and κ is a kernel on $\mathbb{R}^D$ that is twice differentiable along the diagonal such that $\partial_{\mathbf{x}}\partial_{\mathbf{x}'}\kappa(\mathbf{x},\mathbf{x}')|_{\mathbf{x}=\mathbf{x}'}$ exists.*

Assumption 2 (Local smoothness of f)**.** *$f \sim GP(0,\kappa)$ is a continuous sample on $\mathcal{X}$ that has a unique global maximum $\mathbf{x}^*$, such that $f^* - c_1\|\mathbf{x}-\mathbf{x}^*\|_2^\alpha \leq f(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$ and $f(\mathbf{x}) \leq f^* - c_2\|\mathbf{x}-\mathbf{x}^*\|_2^2\ \forall \mathbf{x} \in \mathcal{B}(\mathbf{x}^*,\rho)$ for some constants c_1, c_2, $\rho > 0$ and $\alpha \in \{1,2\}$. Also $f^* - \max_{\mathbf{x}\in\mathcal{X}\setminus\mathcal{B}(\mathbf{x}^*,\rho)} f(\mathbf{x}) > \epsilon_0$ for some $\epsilon_0 > 0$.*

As argued by [55], in many practical cases the local conditions follow almost surely from the global condition. For example, if we were to employ the Gaussian kernel or a

Algorithm 4 BaMSOO

Set $g_{0,0} = f(\mathbf{x}_{0,0})$
Set $f^+ = g_{0,0}$
Initialize the tree $\mathcal{T}_1 = \{0, 0\}$
Set $t = 1$, $n = 1$, $N = 1$, and $\mathcal{D}_t = \{(\mathbf{x}_{0,0}, g(\mathbf{x}_{0,0}))\}$
while true **do**
  Set $\nu_{\max} = -\infty$.
  for $h = 0$ to $\min\{\text{depth}(\mathcal{T}_n), h_{\max}(n)\}$ **do**
    Select $(h, j) = \arg\max_{j \in \{j | (h,j) \in L_n\}} g(\mathbf{x}_{h,j})$
    if $g(x_{h,j}) > \nu_{\max}$ **then**
      for $i = 0$ to $k - 1$ **do**
        Set $N = N + 1$
        if $\mathcal{U}_N(\mathbf{x}_{h+1,kj+i} | \mathcal{D}_t) \geq f^+$ **then**
          Set $g(\mathbf{x}_{h+1,kj+i}) = f(\mathbf{x}_{h+1,kj+i})$
          Set $t = t + 1$
          $\mathcal{D}_t = \{\mathcal{D}_{t-1}, (\mathbf{x}_{h+1,kj+i}, g(\mathbf{x}_{h+1,kj+i}))\}$
        else
          Set $g(\mathbf{x}_{h+1,kj+i}) = \mathcal{L}_N(\mathbf{x}_{h+1,kj+i} | \mathcal{D}_t)$
        end if
        if $g(\mathbf{x}_{h+1,kj+i}) > f^+$ **then**
          Set $f^+ = g(\mathbf{x}_{h+1,kj+i})$
        end if
      end for
      Add the children of (h, j) to $\mathcal{T}_n$
      Set $\nu_{\max} = g(\mathbf{x}_{h,j})$
      Set $n = n + 1$
    end if
  end for
end while

kernel that is 6 times differentiable along the diagonal, we would have that the samples of the GPs are twice differentiable with probability one. The first case is trivial, while the second result was shown by [74, Theorem 5]. If the $\mathbf{x}^*$ lies in the interior of $\mathcal{X}$, then the Hessian of f at $\mathbf{x}^*$ would be almost surely non-singular as at least one of the eigenvalues of the Hessian is a co-dimension 1 condition in the space of all functions that are smooth at a given point [55]. In this case, we would have that

$$f^* - c_1 \|\mathbf{x} - \mathbf{x}^*\|_2^\alpha \leq f(\mathbf{x}) \leq f^* - c_2 \|\mathbf{x} - \mathbf{x}^*\|_2^2$$

with $\alpha = 2$.

If $\mathbf{x}^*$ lies on the boundary of $\mathcal{X}$ which we assume to be smooth, then $\nabla f(\mathbf{x}^*) \neq 0$ since the additional event of the vanishing of $\nabla f(\mathbf{x}^*)$ is a co-dimension d phenomenon in the space of functions with global maximum at $\mathbf{x}^*$ [55]. In this case, we would have that

$$f^* - c_1 \|\mathbf{x} - \mathbf{x}^*\|_2^\alpha \leq f(\mathbf{x}) \leq f^* - c_2 \|\mathbf{x} - \mathbf{x}^*\|_2^2$$

with $\alpha = 1$.

Finally, a sample from a GP on a compact domain has a unique maximum with probability one. This is because the space of continuous functions on a compact domain that attain their global maximum at more than one point have co-dimension 1 in the space of all continuous functions on that domain [55].

The subsequent assumptions are about the hierarchical partitioning of the search space. They are the same as Assumptions 3 and 4 in [158].

Assumption 3 (Bounded diameters)**.** *There exists a decreasing sequence $\delta(h) > 0$, such that for any depth $h \geq 0$, for any cell $X_{h,i}$ of depth h, we have $\sup_{\mathbf{x} \in X_{h,i}} \ell(\mathbf{x}_{h,i}, \mathbf{x}) \leq \delta(h)$. Here $\ell(\mathbf{x}, \mathbf{y}) := c_1 \|\mathbf{x} - \mathbf{y}\|_2^\alpha$ where $\alpha \in \{1, 2\}$ and $\delta(h) = c\gamma^h$ for some constant $c > 0$ and $\gamma \in (0, 1)$.*

Assumption 4 (Well-shaped cells)**.** *There exists $\nu > 0$ such that for any depth $h \geq 0$, any cell $X_{h,i}$ contains an ℓ-ball of radius $\nu\delta(h)$ centered in $X_{h,i}$.*

Note that depending on the value of α, γ would have to take on a different value for Assumptions 3 and 4 to be satisfied. Regardless of the choice of α and as illustrated in Example 1 of [38], Assumptions 3 and 4 are easy to satisfy in practice; for example when $\mathbf{x} \in [0, 1]^D$ and the split is done along the largest dimension of a cell. This is the case in all our experiments.

Assumption 2 together with Assumptions 3 and 4 impose a "*near optimality*" condition as defined by [158].

We can now present our main result, which is in the form of a corollary to Theorem 8 in Appendix A.

Corollary 1. *Let $d = -(D/4 - D/\alpha)$ and $h_{\max}(n) = n^\epsilon$. Given Assumptions $1 - 4$, we have that with probability at least $1 - \eta$, the loss of BaMSOO is $\mathcal{O}\left(n^{-\frac{1-\epsilon}{d}} \log^{\frac{\alpha}{4-\alpha}}(n^2/\eta)\right)$.*

It is worth pointing out that the result presented in Corollary 1 is based on the number of node expansions n instead of the number of function evaluations. The theory can therefore be strengthened.

If $\alpha = 2$ and $\epsilon = 1/2$, then the above result translates to $\mathcal{O}\left(n^{-\frac{2}{D}} \log(n^2/\eta)\right)$. If $\alpha = 1$ with ϵ being the same as before, then the rate of convergence becomes $\mathcal{O}\left(n^{-\frac{2}{3D}} \log^{\frac{1}{3}}(n^2/\eta)\right)$.

The structure of the proof follows that in [158]. Let $\mathbf{x}_h^*$ denote the optimal node at level h (that is, the node at height h in the branch that contains the optimum $\mathbf{x}^*$). Our proof shows that once $\mathbf{x}_h^*$ is expanded, it does not take long for $\mathbf{x}_{h+1}^*$ to be expanded. Once an optimal node $\mathbf{x}_h^*$ is expanded, by Assumptions 2 and 3, we have that the loss of BaMSOO is no worse than $\delta(h)$.

The main difficulty of the proof lies in the fact that we sometimes do not sample nodes when their UCB values are less than the best observed value. In this case, we can no longer make the claim that an optimal node is expanded soon after its parent. This is because when a node is not expanded, its LCB can be very low due to a high standard deviation. Fortunately, we can show that this is not the case for optimal nodes in the optimal region. This is accomplished by showing that the standard deviation at a point is no more than its distance to the nearest sampled point up to a constant factor (shown in Lemma 4). This enables us to show that every optimal node in the optimal region must have a low standard deviation. Given this result, we can adopt the proof structure outlined in [158].

2.5 Experiments with global optimization benchmarks

In this section, we validate the proposed algorithm with a series of experiments that compare the three algorithms (GP-UCB, SOO, BaMSOO) on global optimization benchmarks. We have omitted the feasible region shrinking algorithm (described in Section 2.1.1) as it is not practical for problems of even moderate dimensions. We have also omitted comparisons to PI and EI as these appear in [97] for the optimization benchmarks described in this chapter.

In our experiments, we used the same hyper-parameters in GP-UCB and BaMSOO for each test function. We also randomized the initial sample point for BaMSOO and GP-UCB so that they are not deterministic. To optimize the acquisition function for GP-UCB, we used DIRECT followed by a local optimization method using gradients.

We use 5 test functions: Branin, Rosenbrock, Hartmann3, Hartmann6, and Shekel. All of these test functions are common in the global optimization literature and with the exception of the Rosenbrock, they are all multi-modal. [1]

We rescaled the domain of each function to the $[0, 1]^D$ hypercube, and we used the log distance to the true optimum as our evaluation metric. This metric is defined as $\log_{10}(f^* - f^+)$ where f^+ is the best objective value sampled so far and f^* is the true maximum value of the objective. For each test function, we repeat our experiments 50 times for GP-UCB and BaMSOO and run SOO once as SOO is a deterministic strategy. We plot the mean and a confidence bound of one standard deviation of our metric across all the runs for all the tests.

For simplicity, we only consider binary-trees for space partitioning in SOO and BaMSOO. Specifically, the largest dimension in the parent's cell is split to create two children.

[1] Detailed information about the test functions is available at the following website: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm.

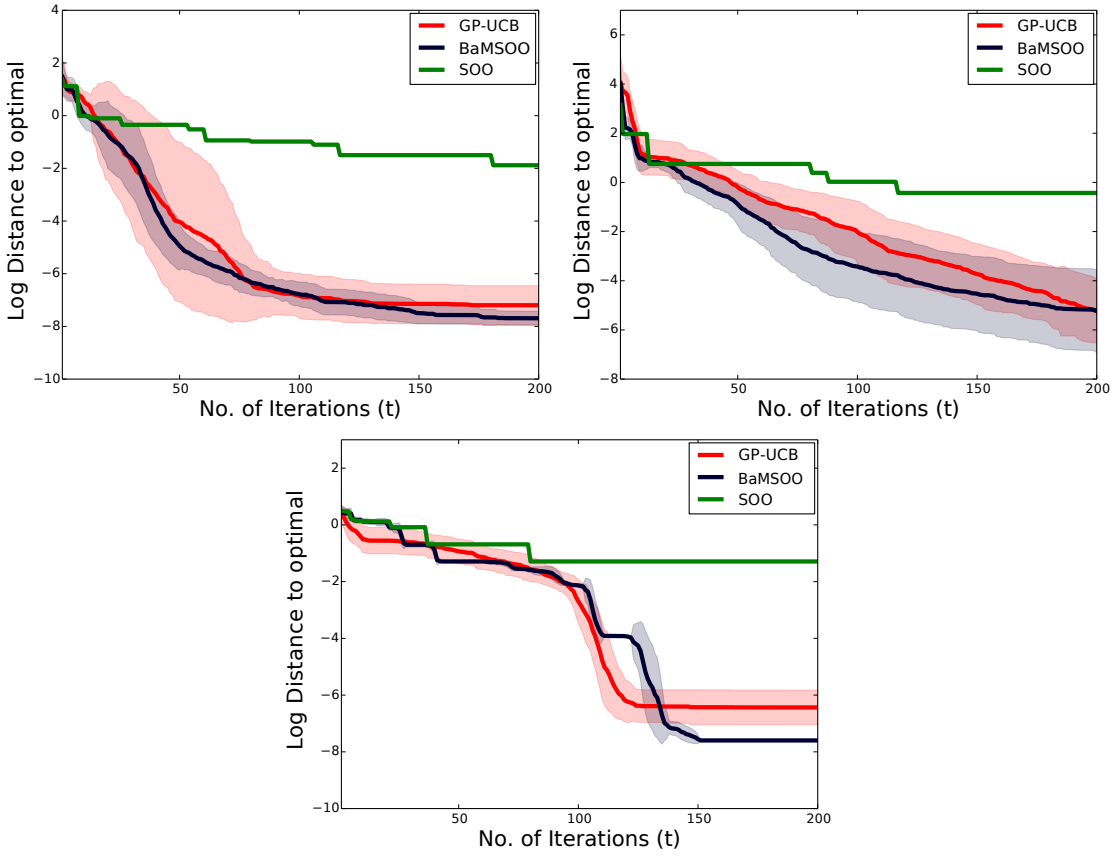


Figure 2.3: Comparison of GP-UCB, SOO, and BaMSOO on multi-modal test functions of low dimensionality (Branin, Rosenbrock and Hartmann3D). GP-UCB and BaMSOO perform similarly whereas SOO does poorly. The poor performance of SOO is caused by having weaker assumptions on the smoothness of the objective function. The good performance of GP-UCB indicates that when the dimensionality is low optimizing the acquisition function is reasonable.

First, we test the global optimization schemes on 3 test functions with low dimensionality: Branin, Rosenbrock and Hartmann3. The Branin function [112] is a common benchmark for Bayesian optimization and has 2 dimensions. The Rosenbrock function is a commonly used non-convex test function for local optimization algorithms, and although it is unimodal, its optimum lies in a long narrow valley, which makes the function hard to optimize. Finally, the Hartmann3 function is 3-dimensional and has four local optima.

As we can see from Figure 2.3, BaMSOO performs competitively against GP-UCB on these low dimensional test functions. Both BaMSOO and GP-UCB achieve very high accuracies of up to 10^{-8} in terms of the distance to the optimal objective value. In comparison, SOO, due to the lack of a strong prior assumption, cannot take advantage of the points sampled and thus is lagging behind.

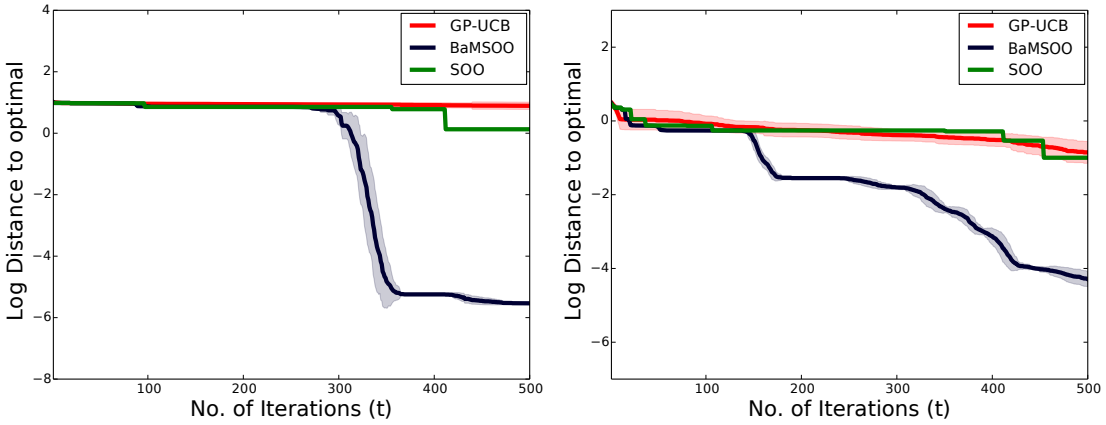


Figure 2.4: Comparison of GP-UCB, SOO, and BaMSOO on multi-modal test functions of moderate dimensionality: 4D Shekel function (LEFT) and 6D Hartmann function (RIGHT). Here, GP-UCB performs poorly. This is due in part to the hardness of optimizing the acquisition function.

Algorithm	Branin	Rosenbrock	Hartmann3	Hartmann6	Shekel
GP-UCB	29.9438	29.5716	34.0311	115.2402	100.7770
BaMSOO	3.0680	3.4693	3.9722	2.0918	3.8951
SOO	0.1810	0.1835	0.1871	0.4313	0.4350

Table 2.1: Time required for the test functions measured in seconds. SOO is very fast as it does not maintain a GP. BaMSOO maintains a GP to produce more accurate posterior estimates and is hence slower. The rejection of proposals also results in bigger trees, further slowing down the algorithm. GP-UCB is slow compared to the other two algorithms as it not only maintains a GP but also optimizes its acquisition function at each iteration.

In the experiments shown in Figure 2.4, we compare the approaches in consideration on the Shekel function and the Hartmann6 function. The Shekel function is 4-dimensional and has 10 local optima. The Hartmann6 function is 6-dimensional, as the name suggests, and has 6 local optima. On these higher dimensional problems, the performance of GP-UCB begins to dwindle. Despite the increase in dimensionality, BaMSOO is still able to optimize the test functions to a relatively high precision. SOO does not perform as well as BaMSOO again because of its weak assumptions. The poor performance of GP-UCB on these two test functions may be due in part to the inability of a global optimizer to optimize the acquisition function exactly in each iteration. As the dimensionality increases, so is the difficulty of optimizing a non-convex function globally as the cost of covering the space grows exponentially. The optimization of the acquisition function through algorithms like DIRECT demands the repartitioning of the space in each iteration. To reach a finer granu-

larity, we either have to sacrifice speed by building very fine partitions in each iteration or accuracy by using coarser partitions.

The proposed approach is not only competitive with GP-UCB in terms of effectiveness, it is also more computationally efficient. As we can see in Table 1, BaMSOO is about 10-40 times faster than GP-UCB on the test functions that we have experimented with. This is because instead of optimizing the acquisition function in each iteration the SOO algorithm, that sits inside, only optimizes once. BaMSOO, however, is much slower than SOO. This is because BaMSOO also employs a GP to reject points proposed by SOO. To sample one point, SOO may have to propose many points before one is accepted. For this reason, BaMSOO would build much bigger trees compared to SOO and it is therefore slower.

2.6 Application to term extraction

In this section, we evaluate the performance of the BaMSOO algorithm on optimizing the parameters in a term extraction algorithm. Term extraction is the process of analyzing a text corpus to find terms, where terms correspond to cohesive sequences of words describing entities of interest. Term extraction tools are widely used in industrial text mining and play a fundamental role in the construction of knowledge graphs and semantic search products. Recently [168] proposed a term extraction method, and showed that it outperforms state-of-the-art competitors, but their method has many free parameters that require manual adjustment. Here, we compare the performance of BaMSOO, GP-UCB and SOO in automatically tuning the 4 primary free parameters of the algorithm (support-thresholds). We define our deterministic objective function to be the F-score of the extracted terms, which is a weighted average of precision and recall. Precision is calculated using a predefined set of correct terms and recall is estimated by simply normalizing the number of extracted correct terms to be in the range [0,1]. We run the experiment on the GENIA corpus [118], which is a collection of 2000 abstracts from biomedical articles. The results of the experiment are shown in Figure 2.5. It is evident from this figure that BaMSOO outperforms GP-UCB and SOO in this application.

2.7 Discussion

This chapter introduced a new global optimization algorithm BaMSOO, which does not require the auxiliary optimization of either acquisition functions or samples from the GP. In trials with benchmark functions from the global optimization literature, the new algorithm

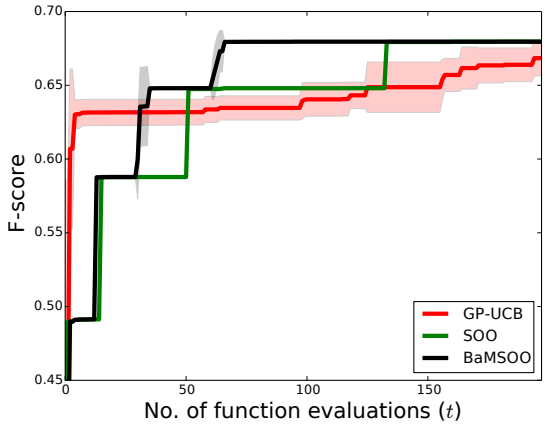


Figure 2.5: Comparison of GP-UCB, SOO, and BaMSOO on optimizing 4 parameters in term extraction from the GENIA corpus using a term extraction algorithm by [168]. In this plot, higher is better.

outperforms standard Bayesian optimization with GPs and SOO, while being computationally efficient. The chapter also provided a theoretical analysis proving that the loss of BaMSOO decreases polynomially.

The careful reader may have noticed that, despite the effectiveness of BaMSOO in the experiments, the convergence rate of BaMSOO is not as good as that of SOO for $\alpha = 2$. This is because we were only able to prove that the standard deviation at a point decreases linearly, instead of quadratically, when a nearby point is sampled (Lemma 4 in the Appendix). Since by assumption the objective function behaves quadratically in the optimal region, the linear decrease of the standard deviation gives rise to a sub-optimal convergence rate. It is also interesting to note that the same type of bound on the standard deviation was used by [39], who achieved similar convergence rates to the ones in this chapter. [55] showed that if the samples form a δ-cover on a subset of $\mathcal{D} \subseteq \mathcal{X}$, then the standard deviation of all points on $\mathcal{D}$ is bounded by a quadratic term $\frac{Q}{4}\delta^2$. Via this observation, the authors achieved a geometric convergence rate. The requirement of the δ-cover, however, renders their algorithm impractical. After the publication of our original paper, the authors of [116] improves BaMSOO and proves geometric convergence. Their technique is base on changing the partition of the space in BaMSOO according to the structure imposed by the Gaussian processes. Computationally, however, their approach is much more expensive since they require many more evaluations of UCB when "looking ahead" to decide the partition intervals.

BaMSOO currently only handles deterministic observations. Since noisy observations

are abundant in practice, extending BaMSOO to the stochastic case remains important future work.

Chapter 3

Theoretical Analysis of Bayesian Optimization with Unknown Gaussian Process Hyper-Parameters

In Bayesian optimization, Gaussian processes are one of the preferred priors for quantifying the uncertainty in the objective function [34]. However, estimating the hyper-parameters of the Gaussian process kernel with very few objective function evaluations is a daunting task, often with disastrous results as illustrated by a simple example in [19]. The typical estimation of the hyper-parameters by maximising the marginal likelihood [176, 113] can easily fall into traps; as shown in [39].

Several authors have proposed to integrate out the hyper-parameters using quadrature and Monte Carlo methods [166, 33, 202]; also see Section 1.6.1. Despite the advantages brought in by this more sophisticated treatment of uncertainty, Bayesian optimization can still fall in traps, as illustrated with a simple simulation example in this chapter.

To the best of our knowledge, the work of Bull [39] provides the only known regret bound for Bayesian optimization when the hyper-parameters are unknown. By introducing lower and upper bounds on the possible range of hyper-parameters values, Bull obtains convergence rates for deterministic objective functions, when estimating the hyper-parameters by maximum likelihood. Here, we extend the work of Bull to stochastic objective functions. Our results apply to sub-Gaussian noise, *e.g.*, symmetric Gaussian, Bernoulli, or uniform noise. This is we assume the observation at the t^{th} iteration is stochastic, $y_t = f(\mathbf{x}_t) + \epsilon_t$. Our goal is to maximise the cumulative rewards $\sum_{t=1}^{T} f(\mathbf{x}_t)$. That is, we wish to approach the performance of the optimiser $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ rapidly. Since the optimiser is unknown, we have to trade-off exploitation and exploration in the search process.

3.1 Preliminaries

In this chapter, we use slightly different notations than introduced in Chapter 1 such that out theoretical analysis is more accessible. In this section, we re-introduce some of the concepts covered previously and adapt them to the need of this chapter.

GP priors are completely characterised by a mean function $m(\cdot)$ and a covariance kernel $\kappa(\cdot,\cdot)$ on the index sets $\mathcal{X}$ and $\mathcal{X}\otimes\mathcal{X}$. In this chapter we consider covariance kernel $\kappa^{\boldsymbol{\theta}}(\mathbf{x}_i,\mathbf{x}_j)$ parametrised by $\boldsymbol{\theta}$. In this chapter, we assume a zero-mean prior. We consider the following types of covariance kernels

$$\begin{aligned}
\kappa^{\boldsymbol{\theta}}_{\text{SE}}(\mathbf{x},\mathbf{x}') &= \exp(-\tfrac{1}{2}r^2) && (3.1)\\
\kappa^{\boldsymbol{\theta}}_{\text{MATÉRN5}}(\mathbf{x},\mathbf{x}') &= \exp(-\sqrt{5}r)(1+\sqrt{5}r+\tfrac{5}{3}r^2) && (3.2)\\
\text{where } r &= (\mathbf{x}-\mathbf{x}')^{\mathsf{T}}\text{diag}(\boldsymbol{\theta}^2)^{-1}(\mathbf{x}-\mathbf{x}').
\end{aligned}$$

Both kernels are parametrised by d length-scale hyper-parameters θ_i. The astute reader may notice that the scaling parameter θ_0 is omitted in the above definitions. The equivalence of the parameter, however, is introduced in our choice acquisition functions defined in Equation (3.7).

We assume that the observations of the function at any point $\mathbf{x}_t$ are corrupted by σ-sub-Gaussian noise $y_t = f(\mathbf{x}_t)+\epsilon_t$. Our theoretical results cover this general type of noise, which encompasses symmetric Gaussian and Bernoulli noise. However, for ease of presentation, we will focus on the tractable case of Gaussian noise $\epsilon_t \sim \mathcal{N}(0,\sigma^2)$ in this section. We refer the reader to [33] for an example of discrete noise, which necessitates the introduction of approximate inference methods.

Given the data $\mathcal{D}_t = \{\mathbf{x}_{1:t},\mathbf{y}_{1:t}\}$, it is well known that the predictive posterior distribution of any evaluation point $\mathbf{x}$ is marginally Gaussian $f(\mathbf{x})|\mathcal{D}_t,\boldsymbol{\theta} \sim \mathcal{N}(\mu_t(\mathbf{x};\boldsymbol{\theta}),\sigma_t(\mathbf{x};\boldsymbol{\theta}))^2$, where

$$\begin{aligned}
\mu_t(\mathbf{x};\boldsymbol{\theta}) &= \mathbb{E}\left[f(\mathbf{x})|\mathcal{D}_t\right] = \mathbf{k}^{\boldsymbol{\theta}}_t(\mathbf{x})^{\mathsf{T}}(\mathbf{K}^{\boldsymbol{\theta}}_t+\sigma^2\mathbf{I})^{-1}\mathbf{y}_{1:t}, && (3.3)\\
\mathcal{K}^{\boldsymbol{\theta}}_t(\mathbf{x},\mathbf{x}') &= \mathbb{E}\left[f(\mathbf{x})f(\mathbf{x}')|\mathcal{D}_t\right] = \kappa^{\boldsymbol{\theta}}(\mathbf{x},\mathbf{x}') - \mathbf{k}^{\boldsymbol{\theta}}_t(\mathbf{x})^{\mathsf{T}}(\mathbf{K}^{\boldsymbol{\theta}}_t+\sigma^2\mathbf{I})^{-1}\mathbf{k}^{\boldsymbol{\theta}}_t(\mathbf{x}') && (3.4)\\
\sigma_t(\mathbf{x};\boldsymbol{\theta})^2 &= \mathcal{K}^{\boldsymbol{\theta}}_t(\mathbf{x},\mathbf{x}). && (3.5)
\end{aligned}$$

It is interesting to note that $\mathcal{K}^{\boldsymbol{\theta}}_t(\mathbf{x},\mathbf{x}')$ also defines a kernel. For more detailed descriptions of Gaussian processes and kernels, please refer to Section 1.4.

In this chapter, we consider the Expected Improvement (defined in Equation 1.46) as our acquisition function. The target of improvement τ is often set to be $f(\mathbf{x}^+)$ when we have deterministic observations. While this is reasonable for deterministic optimization, the

noise in the evaluation of the incumbent, $f(\mathbf{x}^+)$, causes it to be brittle in the stochastic case. In the stochastic setting, the improvement over the best mean value $\mu^+_{\boldsymbol{\theta}} = \max_{\mathbf{x}\in\mathcal{X}} \mu_t(\mathbf{x};\boldsymbol{\theta})$ seems to be a more reasonable alternative. For this choice, we obtain a similar expression for EI,

$$\alpha^{\mathrm{EI}(\mu)}_{\boldsymbol{\theta}}(\mathbf{x}|\mathcal{D}_t) = \mathbb{E}[\max\{0, f(\mathbf{x}) - \mu^+_{\boldsymbol{\theta}}\}|\mathcal{D}_t] = \sigma_t(\mathbf{x};\boldsymbol{\theta})[u\Phi(u) + \phi(u)], \tag{3.6}$$

where $u = \frac{\mu_t(\mathbf{x};\boldsymbol{\theta}) - \mu^+_{\boldsymbol{\theta}}}{\sigma_t(\mathbf{x};\boldsymbol{\theta})}$. In this chapter, we will consider a re-scaled version of this criterion:

$$\alpha^{\mathrm{EI}}_{\boldsymbol{\theta}}(\mathbf{x}|\mathcal{D}_t) = \mathbb{E}[\max\{0, f(\mathbf{x}) - \mu^+_{\boldsymbol{\theta}}\}|\mathcal{D}_t] = \nu\sigma_t(\mathbf{x};\boldsymbol{\theta})[\frac{u}{\nu}\Phi(\frac{u}{\nu}) + \phi(\frac{u}{\nu})] \tag{3.7}$$

where ν[1] is a parameter must be estimated. Intuitively, this parameter enables us to rescale the kernel. In the deterministic case, it plays an equivalent role to multiplying the kernel by an unknown coefficient (as do θ_0 defined in Section 1.4.2). (For notational simplicity, we are not making dependence of EI on ν explicitly in the expression $\alpha^{\mathrm{EI}}_{\boldsymbol{\theta}}(\mathbf{x}|\mathcal{D}_t)$.)

3.2 Theoretical analysis

Our theoretical analysis uses *regret* to measure convergence and *information gain* to measure how informative the samples are about $f(\cdot)$. It assumes that the noise process ϵ_t is *sub-Gaussian*, and that the function $f(\cdot)$ is smooth according to the *reproducing kernel Hilbert space (RKHS)* associated with the GP kernel $k^{\boldsymbol{\theta}}(\cdot,\cdot)$. Before presenting our main result, we briefly review these four background areas.

3.2.1 Background: Regret

As in [205], we will measure the performance of the Bayesian optimization algorithm using *regret*. The instantaneous regret at iteration t is defined as $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$. The corresponding cumulative regret after T iterations is $R_T = \sum_{t=1}^{T} r_t$. While the regret measures are never revealed to the algorithm, bounds on these enable us to assess how rapidly the algorithm is converging. See Section 1.3.4.2 for an introduction to regret in the context of Multi-armed bandits.

[1] The notation is overloaded. ν is defined as the smoothness parameter of the Matérn kernel in Section 1.4.2.

3.2.2 Background: Sub-Gaussian noise

We assume independent σ-sub-Gaussian noise. Formally, we say ϵ_t is σ-sub-Gaussian if there exists a $\sigma \geq 0$ such that

$$\mathbb{E}\left[\exp(\rho\epsilon_t)\right] \leq \exp\left(\frac{\rho^2\sigma^2}{2}\right) \ \forall \rho \in \mathbb{R}.$$

In other works, ϵ_t is σ-sub-Gaussian if its Laplace transform is dominated by the Laplace transform of a Gaussian random variable with zero mean and variance σ^2. It is easy to show that if ϵ_t is sub-Gaussian, then $\mathbb{E}[\epsilon_t] = 0$ and $\mathrm{Var}[\epsilon_t] \leq \sigma^2$.

There are many examples of sub-Gaussian variables, including zero-mean Gaussian random variables with variance σ^2, symmetric Bernoulli random variables and symmetric uniform distributions.

3.2.3 Background: Information gain

To measure the reduction in uncertainty about $f(\cdot)$ from observing $\mathbf{y}_\mathcal{A}$ for a set of sampling points $\mathcal{A} \subset \mathcal{X}$, we need to introduce the concept of *information gain*, which is defined as the mutual information between $f(\cdot)$ and a set of observations $\mathbf{y}_\mathcal{A}$:

$$\mathcal{I}(\mathbf{y}_\mathcal{A}; f(\cdot)) = H(\mathbf{y}_\mathcal{A}) - H(\mathbf{y}_\mathcal{A}|f(\cdot)). \tag{3.8}$$

This concept plays a central role in the results of [205], who also define the *maximum information gain* γ_T after T decision rounds as

$$\gamma_T = \max_{\mathcal{A}\subset\mathcal{X}:|\mathcal{A}|=T} \mathcal{I}(\mathbf{y}_{1:t}; f(\cdot)). \tag{3.9}$$

Note that for Gaussian distributions,

$$\gamma_T^{\boldsymbol{\theta}} = \max_{\mathcal{A}\subset\mathcal{X}:|\mathcal{A}|=T} \frac{1}{2}\log|\mathbf{I} + \sigma^{-2}\mathbf{K}_\mathcal{A}^{\boldsymbol{\theta}}|. \tag{3.10}$$

Our regret bounds will be given in terms of $\gamma_T^{\boldsymbol{\theta}}$. It should perhaps be clarified that the bounds apply to σ-sub-Gaussian noise, despite the appearance of the variable $\gamma_T^{\boldsymbol{\theta}}$ in their statements.

3.2.4 Background: Reproducing kernel Hilbert spaces

To discuss convergence, we must state formally what we mean by $f(\cdot)$ being smooth. In short, we assume that $f(\cdot)$ is an element of an RKHS with reproducing kernel $k(\cdot,\cdot)$. For an intuitive grasp of this formalization of smoothness, we need to briefly review some RKHS fundamentals. These fundamentals are also evoked in our proofs.

Let $L_{\mathbf{x}}$ be an evaluation functional: $L_{\mathbf{x}}f(\cdot) = f(\mathbf{x})$. A (real) RKHS $\mathcal{H}$ is a Hilbert space of real valued functions with the property that for each $\mathbf{x} \in \mathcal{X}$, the evaluation functional is bounded. That is, there exists a positive constant $M = M_{\mathbf{x}}$ such that $|L_{\mathbf{x}}f(\cdot)| = |f(\mathbf{x})| \leq M\|f(\cdot)\|_{\mathcal{H}}$ for all functions $f(\cdot) \in \mathcal{H}$, where $\|\cdot\|_{\mathcal{H}}$ denotes the norm in the Hilbert space. If $\mathcal{H}$ is an RKHS, by the Riesz Representation Theorem, there exists an element $k(\cdot, \mathbf{x}) \in \mathcal{H}$ with the property,

$$f(\mathbf{x}) = L_{\mathbf{x}}f(\cdot) = \langle k(\cdot, \mathbf{x}), f(\cdot)\rangle \tag{3.11}$$

for all $\mathbf{x} \in \mathcal{X}$ and $f(\cdot) \in \mathcal{H}$, where $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathcal{H}$.

To construct $\mathcal{H}$, we consider the linear manifold $\sum_{t=1}^{n} \lambda_t k(\cdot, \mathbf{x}_t)$ for all choices of n, $\lambda_1, \ldots, \lambda_n$ and $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$, with inner product

$$\langle \sum_{i=1}^{n} \lambda_i k(\cdot, \mathbf{x}_i), \sum_{j=1}^{n} \lambda_j k(\cdot, \mathbf{x}_j)\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i k(\mathbf{x}_i, \mathbf{x}_j)\lambda_j = \| \sum_{i=1}^{n} \lambda_i k(\cdot, \mathbf{x}_i)\|_{\mathcal{H}}^2 \geq 0. \tag{3.12}$$

The above norm is non-negative because of the positive-definiteness of $k(\cdot, \cdot)$. Clearly, for any element $f(\cdot)$ of this linear manifold,

$$f(\mathbf{x}_j) = \langle f(\cdot), k(\cdot, \mathbf{x}_j)\rangle = \langle \sum_{i=1}^{n} \lambda_i k(\cdot, \mathbf{x}_i), k(\cdot, \mathbf{x}_j)\rangle = \sum_{i=1}^{n} \lambda_i k(\mathbf{x}_i, \mathbf{x}_j) \tag{3.13}$$

A consequence of this is that for any Cauchy sequence $\{f_n(\cdot)\}$, we have the following bound by Cauchy-Schwartz: $|f_n(\mathbf{x}) - f(\mathbf{x})| = \langle f_n(\cdot) - f(\cdot), k(\cdot, \mathbf{x})\rangle \leq \|f_n(\cdot) - f(\cdot)\|_{\mathcal{H}}\|k(\cdot, \mathbf{x})\|_{\mathcal{H}}$. In words, norm convergence implies point-wise convergence.

The preceding steps illustrate that we can construct a unique RKHS for any positive definite kernel $k(\cdot, \cdot)$. The converse is also true (Moore-Aronszajn Theorem).

A positive definite function $k(\cdot, \cdot)$, under general conditions, has an eigenvector-eigenvalue decomposition. Suppose $k(\cdot, \cdot)$ is continuous and $\int_{\mathcal{X}} \int_{\mathcal{X}} k^2(\mathbf{x}, \mathbf{y})d\mathbf{x}d\mathbf{y} < \infty$, then there exists an orthonormal sequence of continuous eigenfunctions $q_1(\cdot), q_2(\cdot), \ldots$ and eigenvalues $\Delta_1 \geq \Delta_2 \geq \ldots \geq 0$, with $k(x, y) = \sum_{\nu=1}^{\infty} \Delta_\nu q_\nu(\mathbf{x})q_\nu(\mathbf{y})$. Next, consider the orthonormal expansion $f(\cdot) = \sum_{\nu=1}^{\infty} f_\nu q_\nu(\cdot)$ with coefficients $f_\nu = \int_{\mathcal{X}} f(\mathbf{x})q_\nu(\mathbf{x})d\mathbf{x}$. It is easy to prove that $f(\cdot)$ is an element of the RKHS associated with $k(\cdot, \cdot)$ if and only if

$$\|f(\cdot)\|_{\mathcal{H}}^2 = \sum_{\nu=1}^{\infty} \frac{f_\nu^2}{\Delta_\nu} < \infty. \tag{3.14}$$

To obtain the above finiteness condition, the coefficients f_ν of the expansion of $f(\cdot)$ must decay quickly. For the kernels we consider in this chapter, elements of RKHS can uniformly approximate any continuous function with compact support. Therefore, RKHS is well suited as a tool for analyzing convergence behaviors of Bayesian optimization algorithms.

3.2.5 Main result

In this section, we present our regret bound and sketch its proof. For space considerations, detailed proofs appear in Appendix B.

As discussed when presenting the algorithm, our theorem assumes bounds on the kernel hyper-parameters of the form $\boldsymbol{\theta}^L \leq \boldsymbol{\theta}_t \leq \boldsymbol{\theta}^U$ for all $t \geq 1$ with $f(\cdot) \in \mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})$. While we could recall all the conditions on the kernel function necessary for our theorem to apply, we simply restrict the family of kernels to one that satisfies the conditions detailed in [39]. Without loss of generality, we assume that $k(\mathbf{x}, \mathbf{x})$= 1.

Our theorem characterizing the growth in the cumulative regret R_T with the number of function evaluations T follows.

Theorem 1. *Let* $C_2 := \prod_{i=1}^{d} \frac{\theta_i^U}{\theta_i^L}$. *Suppose* $\boldsymbol{\theta}^L \leq \boldsymbol{\theta}_t \leq \boldsymbol{\theta}^U$ *for all* $t \geq 1$ *and* $f(\cdot) \in \mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})$. *If* $\left(\nu_t^{\boldsymbol{\theta}}\right)^2 = \boldsymbol{\Theta}\left(\gamma_{t-1}^{\boldsymbol{\theta}} + \log^{1/2}(2t^2\pi^2/3\delta)\sqrt{\gamma_{t-1}^{\boldsymbol{\theta}}} + \log(t^2\pi^2/3\delta)\right)$ *for all* $t \geq 1$. *Then with probability at least* $1-\delta$, *the cumulative regret obeys the following rate:*

$$R_T = \mathcal{O}\left(\sqrt{\beta_T \gamma_T^{\boldsymbol{\theta}^L} T}\right), \tag{3.15}$$

where $\beta_T = 2\log\left(\frac{T}{\sigma^2}\right)\gamma_{T-1}^{\boldsymbol{\theta}^L} + \sqrt{8\log\left(\frac{T}{\sigma^2}\right)}\log^{1/2}(4T^2\pi^2/6\delta)\left(\sqrt{C_2}\|f\|_{\mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})} + \sqrt{\gamma_{T-1}^{\boldsymbol{\theta}^L}}\right) + C_2\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})}$.

Our result is analogous to Theorem 3 of [205] which proves convergence rates for the GP-UCB algorithm in the agnostic setting. Their result, however, does not allow for the estimation of hyper-parameters. In addition, our algorithm does not require explicit knowledge of the RKHS norm of the objective function while GP-UCB does require this.

Using the results of Srinivas *et al.* we can further detail these rates as follows.

Theorem 2 (Theorem 5 of [205])**.** *Let* $\mathcal{X} \subseteq \mathbb{R}^d$ *be compact and convex,* $d \in \mathbb{N}$. *Assume the kernel function satisfies* $k(\mathbf{x}, \mathbf{x}') \leq 1$.

1. *Exponential spectral decay. For the squared Exponential kernel:* $\gamma_T^{\boldsymbol{\theta}} = \mathcal{O}\left((\log T)^{d+1}\right)$.
2. *Power law spectral decay. For Matérn kernels with degree of freedom* $\nu > 1$*:* $\gamma_T^{\boldsymbol{\theta}} = \mathcal{O}\left(T^{d(d+1)/(2\nu+d(d+1))}\log T\right)$.

The proof of Theorem 1 is provided in Appendix B. We sketch the main ideas here. Our proof methodology is inspired by the works of [205] and [39].

We start the proof-sketch by considering the instantaneous regret:

$$\begin{aligned} r_t &= f(\mathbf{x}^*) - f(\mathbf{x}_t) \\ &= (f(\mathbf{x}^*) - \mu^+_{\boldsymbol{\theta}_t}) - (f(\mathbf{x}_t) - \mu^+_{\boldsymbol{\theta}_t}) \end{aligned}$$

where $\mu^+_{\boldsymbol{\theta}_t} = \max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t)$ and $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha^{\text{EI}}_{\boldsymbol{\theta}_t}(\mathbf{x}|\mathcal{D}_{t-1})$. T

The first challenge of the proof is to bound the difference between the posterior mean of the GP and the objective function. Such a bound allows us to quantify the difference between our belief about the objective function and the true objective. Specifically, we bound $|\mu_{t-1}(\mathbf{x}, \boldsymbol{\theta}_t) - f(\mathbf{x})|\ \forall \mathbf{x} \in \mathcal{X}$ in each iteration with high probability. By way of the Cauchy-Schwarz inequality,

$$\begin{aligned} |\mu_{t-1}(\mathbf{x}, \boldsymbol{\theta}_t) - f(\mathbf{x})| &\leq \left(\mathcal{K}^{\boldsymbol{\theta}_t}_{t-1}(\mathbf{x}, \mathbf{x})\right)^{1/2} \|\mu_{t-1}(\cdot; \boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}^{\boldsymbol{\theta}_t}_{t-1}} \\ &\leq \sigma_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t) \|\mu_{t-1}(\cdot; \boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}^{\boldsymbol{\theta}_t}_{t-1}}. \end{aligned}$$

The first part of our proof (Section B.1 in the appendix) is then dedicated to providing a probabilistic bound for $\|\mu_{t-1}(\cdot; \boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}^{\boldsymbol{\theta}_t}_{t-1}}$ by means of concentration inequalities. In more detail, Lemma 14 and 15 bound separate terms that appear in $\|\mu_{t-1}(\cdot; \boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}^{\boldsymbol{\theta}_t}_{t-1}}$ using properties of reproducing kernel Hilbert spaces and concentration results for sub-Gaussian random variables [100]. Proposition 2 combines the aforementioned results via a union bound.

The second challenge of the proof is to relate EI with quantities that are easier to analyse, such as the posterior variance and the improvement function $I^{\boldsymbol{\theta}}_t(\mathbf{x}) = \max\{0, f(\mathbf{x}) - \mu^+_{\boldsymbol{\theta}}\}$. To bound the instantaneous regret, we observe that

$$(f(\mathbf{x}^*) - \mu^+_{\boldsymbol{\theta}_t}) - (f(\mathbf{x}_t) - \mu^+_{\boldsymbol{\theta}_t}) \leq I^{\boldsymbol{\theta}}_t(\mathbf{x}) + \left[(\mu^+_{\boldsymbol{\theta}_t} - \mu_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)) + \varphi^{\boldsymbol{\theta}_t}_t \sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)\right].$$

(Here $\varphi^{\boldsymbol{\theta}_t}_t$ is a quantity that arises in the concentration bound of Proposition 2.) The improvement function is upper-bounded by a constant times the expected improvement $\alpha^{\text{EI}}_{\boldsymbol{\theta}_t}(\mathbf{x}_t|\mathcal{D}_{t-1})$ via Lemma 20, which builds on results by [39]. The expected improvement is in turn bounded by a multiple of the posterior standard deviation $\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)$.

Next, we turn our attention to the term $(\mu^+_{\boldsymbol{\theta}_t} - \mu_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t))$. We bound this term in Lemma 21, which states that $\left|\mu_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t) - \mu^+_{\boldsymbol{\theta}_t}\right| \leq \sqrt{\log(t - 1 + \sigma^2) - \log(\sigma^2)}\nu\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)$.

By now, we have bounded the instantaneous regret in each iteration by a multiple of the posterior variance $\sigma^2_{t-1}(\mathbf{x}; \boldsymbol{\theta}^L)$. Finally, we can sum over t and use Lemma 18, which states that $\sum_{t=t_0}^{T} \sigma^2_{t-1}(\mathbf{x}; \boldsymbol{\theta}^L) \leq \frac{2}{\log(1+\sigma^2)}\gamma^{\boldsymbol{\theta}^L}_T$, and subsequently bound the cumulative regret by the maximal information gain, which as we said is related to the posterior variance of the GP (Lemma 5). To accommodate different hyper-parameters, we make use of Lemma 19 from [39].

Algorithm 5 Bayesian Optimization with Hyper-parameter Optimization.

input Threshold $t_\sigma > 0$, percentage of reduction parameter $p \in (0, 1)$, and $c_2 > c_1 > 0$.
input Lower and upper bounds $\boldsymbol{\theta}^L$, $\boldsymbol{\theta}^U$ for the hyper-parameters.
input Initial length scale hyper-parameter $\boldsymbol{\theta}^L \leq \boldsymbol{\theta}_1 \leq \boldsymbol{\theta}^U$.

Initialize $E = 0$
for $t = 1, 2, \ldots$ **do**
  Select $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha^{\text{EI}}_{\boldsymbol{\theta}_t}(\mathbf{x}|\mathcal{D}_{t-1})$
  if $\sigma^2_{t-1}(\mathbf{x}_t; \theta_t) < t_\sigma \sigma^2$ **then**
    $E = E + 1$
  else
    $E = 0$
  end if
  Augment the data $\mathcal{D}_t = \mathcal{D}_{t-1} \cup (\mathbf{x}_t, y_t)$
  if $E = 5$ **then**
    Restrict $\boldsymbol{\theta}^U$ such that $\theta^U_i = \max\left\{\min\left[p \max_j\{\theta^U_j\}, \theta^U_i\right], \theta^L_i\right\}$
    $E = 0$
  end if
  Choose hyper-parameters $\boldsymbol{\theta}_{t+1}$ such that $\boldsymbol{\theta}^L \leq \boldsymbol{\theta}_{t+1} \leq \boldsymbol{\theta}^U$.
  Choose hyper-parameter $\nu_t^{\boldsymbol{\theta}_{t+1}}$ such that $c_1 \xi_{t+1}^{\boldsymbol{\theta}_{t+1}} \leq \nu_{t+1}^{\boldsymbol{\theta}_{t+1}} \leq c_2 \xi_{t+1}^{\boldsymbol{\theta}_{t+1}}$, where $\xi_t^{\boldsymbol{\theta}_t}$ is defined in Equation (3.16).
end for

3.3 An algorithm inspired by the theory

Our main theorem (Theorem 1) establishes sufficient conditions to guarantee that the regret of a Bayesian optimization algorithm with EI and hyper-parameter estimation, vanishes as the number of function evaluations increases. To illustrate the value of Theorem 1, we use its guidelines to construct an algorithm in this section.

For Theorem 1 to hold, it is necessary that we adapt the hyper-parameters in a particular manner. First, we must ensure that that there exist upper-bounds on the hyper-parameters $\boldsymbol{\theta}$, which we group in the vector $\boldsymbol{\theta}^U$,such that the objective function $f(\cdot)$ is an element of the reproducing kernel Hilbert space induced by this narrower kernel $\mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})$. Figure 1 (right) shows what happens to the confidence intervals as the entries of $\boldsymbol{\theta}^U$ shrink with t, by narrowing the kernel.

In practice, it is difficult to assess this condition. To surmount this difficulty, we propose to reduce the upper bound of the length scales $\boldsymbol{\theta}^U$ when the algorithm becomes over confident. In particular, we adaptively reduce $\boldsymbol{\theta}^U$ whenever the model repeatedly samples points of low posterior variance in comparison to the noise variance σ^2. Once the algorithm optimizes to the precision of the noise variance, it suffers from a slower convergence rate.

By choosing to lower the upper bound as proposed in Algorithm 5, we essentially en-

able the algorithm to explore more, as opposed to over-exploiting a local mode. This is illustrated in Figure 1, which depicts the result of running the proposed algorithm and a standard Bayesian optimization scheme. We will explain the experiment in more detail at the end of this section.

As $\boldsymbol{\theta}^U$ is successively decreased, after a finite number of iterations, we can ensure that $f(\cdot) \in \mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})$ as long as there exists $\boldsymbol{\theta} \geq \boldsymbol{\theta}^L$ such that $f(\cdot) \in \mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})$. In practice, we advocate a conservative choice of $\boldsymbol{\theta}^L$ whenever we have little knowledge of the range of possible values of $\boldsymbol{\theta}$.

Theorem 1 also imposes a condition on ν. To satisfy it, we constrain $\nu_t^{\boldsymbol{\theta}_t}$ to be in the interval $c_1 \xi_t^{\boldsymbol{\theta}_t} \leq \nu_t^{\boldsymbol{\theta}_t} \leq c_2 \xi_t^{\boldsymbol{\theta}_t}$, where

$$\xi_t^{\boldsymbol{\theta}_t} = \left(\mathcal{I}_{\boldsymbol{\theta}_t}(\mathbf{y}_{t-1}; \mathbf{f}_{t-1}) + \log^{1/2}(2t^2\pi^2/3\delta) \sqrt{\mathcal{I}_{\boldsymbol{\theta}_t}(\mathbf{y}_{t-1}; \mathbf{f}_{t-1})} + \log(t^2\pi^2/3\delta) \right). \quad (3.16)$$

The careful reader may have noticed that the above condition does not match perfectly the condition detailed in Theorem 1. Upon closer examination, however, we see that replacing the maximum information gain $\gamma_T^{\boldsymbol{\theta}}$ with $\mathcal{I}_{\boldsymbol{\theta}}(\mathbf{y}_T; \mathbf{f}_T)$ does not break the convergence result. We have used $\gamma_T^{\boldsymbol{\theta}}$ in Theorem 1 instead of $\mathcal{I}_{\boldsymbol{\theta}}(\mathbf{y}_T; \mathbf{f}_T)$ simply to simplify the presentation.

In practice, we could use a number of strategies for estimating the hyper-parameters, provided they fall within the bounds set by Theorem 1. In particular, we could use maximum likelihood to estimate the hyper-parameters in this constrained space. Note that the ν parameter could also be treated as a kernel hyper-parameter (kernel scale), therefore removing the need of estimating it separately.

Finally, the astute reader would have noticed the parameters t_σ, p, c_2 and c_1 in the algorithm. If we want to achieve an accuracy comparable to the noise variance, we should set $t_\sigma = 1$. The other parameters simply determine how fast the algorithm converges and should be set to reasonable fixed values, e.g. $p = 0.5$, $c_2 = 1$ and $c_1 = 0.001$. Provided $t_\sigma > 0$, $p \in (0, 1)$ and $c_2 > c_1 > 0$, the theory is satisfied.

If we have strong beliefs about our GP prior model, it may seem unnecessary to estimate our parameters with Algorithm 5. When our prior belief is misplaced, however, we could fail to converge if we were to follow the traditional probabilistic approach. We provide an illustration of this effect by optimize the following stochastic function:

$$f(x) = 2\mathbf{k}_{SE}^{\theta_1}(x_1, x) + 4\mathbf{k}_{SE}^{\theta_2}(x_2, x) + \epsilon$$

over the interval $[0, 1]$, where $\theta_1 = 0.1$, $\theta_2 = 0.01$, $x_1 = 0.1$, $x_2 = 0.9$, and ϵ is zero-mean Gaussian with 10^{-2} standard deviation. Figure 3.1 compares Algorithm 5 against standard Bayesian optimization with the same EI function, but using slice sampling to infer the kernel hyper-parameters (without imposing the theoretical bounds on the hyper-parameters).

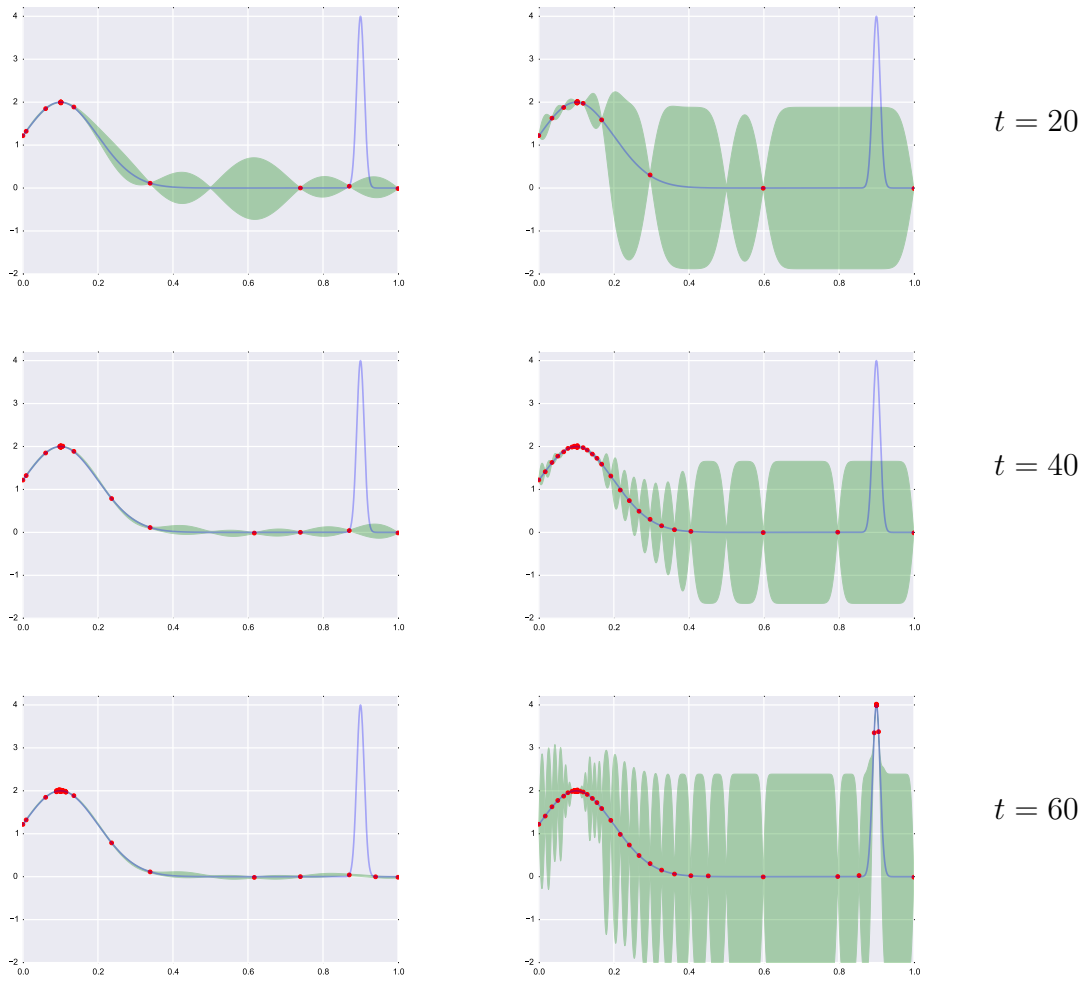


Figure 3.1: Convergence of EI with slice sampling over the kernel hyper-parameters [**left**] and EI using Algorithm 5 [**right**] at three function evaluation steps (t). The objective function (in blue) was constructed so that it has a trap. Unless EI with slice sampling hits the narrow optimum by random chance, it becomes too confident and fails to converge after 60 evaluations. In contrast, the confidence bounds for Algorithm 5 can increase enabling it to sample the function in a more reasonable way and thus find the optimum.

We see that, in the absence of reasonable prior beliefs, conditions like the ones detailed in our theoretical results are necessary to guarantee reasonable sampling of the objective function. (The same behavior for the plot on the left is observed if we replace slice sampling with maximum likelihood estimation of the hyper-parameters.) While heteroskedastic GP approaches could mitigate this problem, there are no theoretical results to guarantee this to the best of our knowledge.

3.4 Conclusion

Despite the rapidly growing literature on Bayesian optimization and the proliferation of software packages that learn the kernel hyper-parameters, to the best of our knowledge, only Bull [39] and us have attacked the question of convergence of GP-based Bayesian optimization with unknown hyper-parameters. Bull's results focused on deterministic objective functions. Our new results apply to the abundant class of noisy objective functions.

Chapter 4

Bayesian Optimization in a Billion Dimensions via Random Embeddings

Despite many success stories of Bayesian optimization, the approach is restricted to problems of moderate dimension, typically up to about 10. Of course, for a great many problems this is all that is needed. However, to advance the state of the art, we need to scale the methodology to high-dimensional parameter spaces. This is the goal of this paper.

It is difficult to scale Bayesian optimization to high dimensions. To ensure that a global optimum is found, we require good coverage of $\mathcal{X}$, but as the dimensionality increases, the number of evaluations needed to cover $\mathcal{X}$ increases exponentially. As a result, there has been little progress on this challenging problem, with a few exceptions. [20] introduced a non-standard Bayesian optimization method based on a tree of one-dimensional density estimators and applied it successfully to optimize the 238 parameters of a complex vision architecture [23]. [107] used random forests models in Bayesian optimization to achieve state-of-the-art performance in optimizing up to 76 mixed discrete/continuous parameters of algorithms for solving hard combinatorial problems, and to successfully carry out combined model selection and hyper-parameter optimization for the 768 parameters of the Auto-WEKA framework [211]. [64] showed that these two methods indeed yielded the best performance for high-dimensional hyper-parameter optimization (e.g., in deep belief networks). However, both are based on weak uncertainty estimates that can fail even for the optimization of very simple functions and lack theoretical guarantees.

In the *linear* bandits case, [43] recently proposed a compressed sensing strategy to attack problems with a high degree of sparsity. Also recently, [47] made significant progress by introducing a two stage strategy for optimization and variable selection of high-dimensional GPs. In the first stage, sequential likelihood ratio tests, with a couple of tuning parameters, are used to select the relevant dimensions. This, however, requires the relevant dimensions to be axis-aligned with an ARD kernel. Chen and colleagues provide empirical results

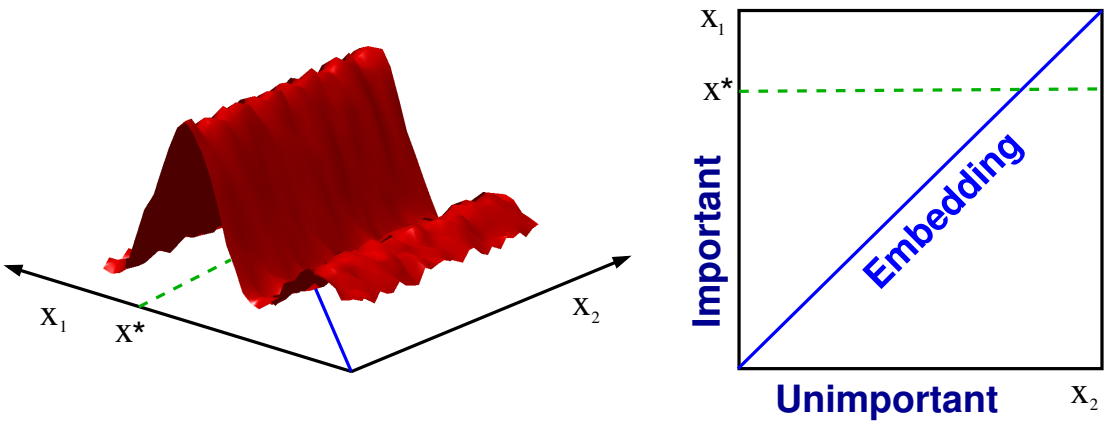


Figure 4.1: This function in D=2 dimensions only has d=1 *effective dimension*: the vertical axis indicated with the word important on the right hand side figure. Hence, the 1-dimensional embedding includes the 2-dimensional function's optimizer. It is more efficient to search for the optimum along the 1-dimensional random embedding than in the original 2-dimensional space. (For illustration purposes, the effective dimensions here are axis-aligned. Our proposed method, however, does not pose such requirements.)

only for synthetic examples (of up to 400 dimensions), but they provide key theoretical guarantees.

Many researchers have noted that for certain classes of problems most dimensions do not change the objective function significantly; examples include hyper-parameter optimization for neural networks and deep belief networks [21], as well as other machine learning algorithms and various state-of-the-art algorithms for solving $\mathcal{NP}$-hard problems [104]. That is to say these problems have *"low effective dimensionality"*. To take advantage of this property, [21] proposed to simply use random search for optimization – the rationale being that points sampled uniformly at random in each dimension can densely cover each low-dimensional subspace. As such, random search can exploit low effective dimensionality *without knowing which dimensions are important*. In this paper, we exploit the same property in a new Bayesian optimization variant based on random embeddings.

Figure 4.1 illustrates the idea behind random embeddings in a nutshell. Assume we know that a given $D = 2$ dimensional black-box function $f(x_1, x_2)$ only has $d = 1$ important dimensions, but we do not know which of the two dimensions is the important one. We can then perform optimization in the embedded 1-dimensional subspace defined by $x_1 = x_2$ since this is guaranteed to include the optimum.

We demonstrate that random embeddings enable us to scale Bayesian optimization to arbitrary D provided the objective function has low intrinsic dimensionality. Importantly, the algorithm associated with this idea, which we called REMBO, is not restricted to cases

with axis-aligned intrinsic dimensions but applies to any d-dimensional linear subspace. [59] recently proposed an adaptive, but more expensive, variant of REMBO with theoretical guarantees.

Our experiments (Section 4.2) also show that REMBO can solve problems of previously untenable high extrinsic dimensions, and that REMBO can achieve state-of-the-art performance for optimizing the 47 discrete parameters of a popular mixed integer linear programming solver.

4.1 Random Embedding for Bayesian Optimization

Before introducing our new algorithm and its theoretical properties, we need to define what we mean by effective dimensionality formally.

Definition 1. *A function $f : \mathbb{R}^D \to \mathbb{R}$ is said to have **effective dimensionality** d_e, with $d_e \leq D$, if*

- *there exists a linear subspace $\mathcal{T}$ of dimension d_e such that for all $\mathbf{x}_\top \in \mathcal{T} \subset \mathbb{R}^D$ and $\mathbf{x}_\perp \in \mathcal{T}^\perp \subset \mathbb{R}^D$, we have $f(\mathbf{x}_\top + \mathbf{x}_\perp) = f(\mathbf{x}_\top)$, where $\mathcal{T}^\perp$ denotes the orthogonal complement of $\mathcal{T}$; and*
- *d_e is the smallest integer with this property.*

*We call $\mathcal{T}$ the **effective subspace** of f and $\mathcal{T}^\perp$ the **constant subspace**.*

This definition simply states that the function does not change along the coordinates $\mathbf{x}_\perp$, and this is why we refer to $\mathcal{T}^\perp$ as the constant subspace. Given this definition, the following theorem shows that problems of low effective dimensionality can be solved via random embedding.

Theorem 3. *Assume we are given a function $f : \mathbb{R}^D \to \mathbb{R}$ with effective dimensionality d_e and a random matrix $\mathbf{A} \in \mathbb{R}^{D \times d}$ with independent entries sampled according to $\mathcal{N}(0, 1)$ and $d \geq d_e$. Then, with probability 1, for any $\mathbf{x} \in \mathbb{R}^D$, there exists a $\mathbf{y} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = f(\mathbf{A}\mathbf{y})$.*

Proof. Since f has effective dimensionality d_e, there exists an effective subspace $\mathcal{T} \subset \mathbb{R}^D$, such that $\text{rank}(\mathcal{T}) = d_e$. Furthermore, any $\mathbf{x} \in \mathbb{R}^D$ decomposes as $\mathbf{x} = \mathbf{x}_\top + \mathbf{x}_\perp$, where $\mathbf{x}_\top \in \mathcal{T}$ and $\mathbf{x}_\perp \in \mathcal{T}^\perp$. Hence, $f(\mathbf{x}) = f(\mathbf{x}_\top + \mathbf{x}_\perp) = f(\mathbf{x}_\top)$. Therefore, without loss of generality, it will suffice to show that for all $\mathbf{x}_\top \in \mathcal{T}$, there exists a $\mathbf{y} \in \mathbb{R}^d$ such that $f(\mathbf{x}_\top) = f(\mathbf{A}\mathbf{y})$.

Let $\mathbf{\Phi} \in \mathbb{R}^{D \times d_e}$ be a matrix, whose columns form an orthonormal basis for $\mathcal{T}$. Hence, for each $\mathbf{x}_\top \in \mathcal{T}$, there exists a $\mathbf{c} \in \mathbb{R}^{d_e}$ such that $\mathbf{x}_\top = \mathbf{\Phi}\mathbf{c}$. Let us for now assume

that $\mathbf{\Phi}^T\mathbf{A}$ has rank d_e. If $\mathbf{\Phi}^T\mathbf{A}$ has rank d_e, there exists a $\mathbf{y}$ such that $(\mathbf{\Phi}^T\mathbf{A})\mathbf{y} = \mathbf{c}$. The orthogonal projection of $\mathbf{A}\mathbf{y}$ onto $\mathcal{T}$ is given by

$$\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{A}\mathbf{y} = \mathbf{\Phi}\mathbf{c} = \mathbf{x}_\top.$$

Thus $\mathbf{A}\mathbf{y} = \mathbf{x}_\top + \mathbf{x}'$ for some $\mathbf{x}' \in \mathcal{T}^\perp$ since $\mathbf{x}_\top$ is the projection $\mathbf{A}\mathbf{y}$ onto $\mathcal{T}$. Consequently, $f(\mathbf{A}\mathbf{y}) = f(\mathbf{x}_\top + \mathbf{x}') = f(\mathbf{x}_\top)$.

It remains to show that, with probability one, the matrix $\mathbf{\Phi}^T\mathbf{A}$ has rank d_e. Let $\mathbf{A}_e \in \mathbb{R}^{D\times d_e}$ be a submatrix of $\mathbf{A}$ consisting of any d_e columns of $\mathbf{A}$, which are *i.i.d.* samples distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, $\mathbf{\Phi}^T\mathbf{a}_i$ are *i.i.d.* samples from $\mathcal{N}(\mathbf{0}, \mathbf{\Phi}^T\mathbf{\Phi}) = \mathcal{N}(\mathbf{0}_{d_e}, \mathbf{I}_{d_e\times d_e})$, and so we have $\mathbf{\Phi}^T\mathbf{A}_e$, when considered as an element of $\mathbb{R}^{d_e^2}$, is a sample from $\mathcal{N}(\mathbf{0}_{d_e^2}, \mathbf{I}_{d_e^2\times d_e^2})$. On the other hand, the set of singular matrices in $\mathbb{R}^{d_e^2}$ has Lebesgue measure zero, since it is the zero set of a polynomial (i.e. the determinant function) and polynomial functions are Lebesgue measurable. Moreover, the Normal distribution is absolutely continuous with respect to the Lebesgue measure, so our matrix $\mathbf{\Phi}^T\mathbf{A}_e$ is almost surely non-singular, which means that it has rank d_e and so the same is true of $\mathbf{\Phi}^T\mathbf{A}$, whose columns contain the columns of $\mathbf{\Phi}^T\mathbf{A}_e$. □

Theorem 3 says that given any $\mathbf{x} \in \mathbb{R}^D$ and a random matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$, with probability 1, there is a point $\mathbf{y} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = f(\mathbf{A}\mathbf{y})$. This implies that for any optimizer $\mathbf{x}^\star \in \mathbb{R}^D$, there is a point $\mathbf{y}^\star \in \mathbb{R}^d$ with $f(\mathbf{x}^\star) = f(\mathbf{A}\mathbf{y}^\star)$. Therefore, instead of optimizing in the high dimensional space, we can optimize the function $g(\mathbf{y}) = f(\mathbf{A}\mathbf{y})$ in the lower dimensional space. This observation gives rise to our new Random EMbedding Bayesian Optimization (REMBO) algorithm (see Algorithm 6). REMBO first draws a random embedding (given by $\mathbf{A}$) and then performs Bayesian optimization in this embedded space.

Algorithm 6 REMBO: Bayesian Optimization with Random Embedding. Blue text denotes parts that are changed compared to standard Bayesian Optimization.

1: Generate a random matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$
2: Choose the bounded region set $\mathcal{Y} \subset \mathbb{R}^d$
3: Initialize $\mathcal{D}_0$ as $\emptyset$.
4: **for** $t = 1, 2, \ldots$ **do**
5: Find $\mathbf{y}_{t+1} \in \mathbb{R}^d$ by optimizing the acquisition function u: $\mathbf{y}_{t+1} = \arg\max_{\mathbf{y}\in\mathcal{Y}} u(\mathbf{y}|\mathcal{D}_t)$.
6: Augment the data $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\mathbf{y}_{t+1}, f(\mathbf{A}\mathbf{y}_{t+1}))\}$.
7: Update the kernel hyper-parameters.
8: **end for**

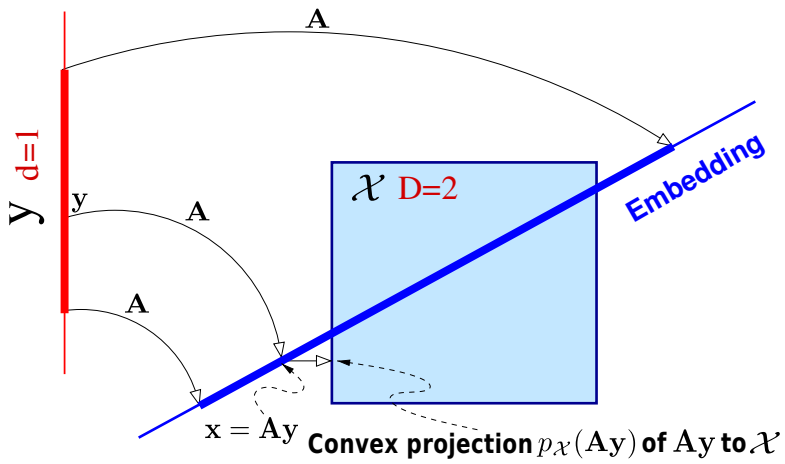


Figure 4.2: Embedding from $d = 1$ into $D = 2$. The box illustrates the 2D constrained space $\mathcal{X}$, while the thicker red line illustrates the 1D constrained space $\mathcal{Y}$. Note that if $\mathbf{A}\mathbf{y}$ is outside $\mathcal{X}$, it is projected onto $\mathcal{X}$. The set $\mathcal{Y}$ must be chosen large enough so that the projection of its image, $\mathbf{A}\mathcal{Y}$, onto the effective subspace (vertical axis in this diagram) covers the vertical side of the box. Despite the drawing, our proposed method does not require axis-aligned effective subspaces.

In many practical optimization tasks, the goal is to optimize f over a compact subset $\mathcal{X} \subset \mathbb{R}^D$ (typically a box), and f can often not be evaluated outside of $\mathcal{X}$. Therefore, when REMBO selects a point $\mathbf{y}$ such that $\mathbf{A}\mathbf{y}$ is outside the box $\mathcal{X}$, it projects $\mathbf{A}\mathbf{y}$ onto $\mathcal{X}$ before evaluating f. That is, $g(\mathbf{y}) = f(p_{\mathcal{X}}(\mathbf{A}\mathbf{y}))$, where $p_{\mathcal{X}} : \mathbb{R}^D \to \mathbb{R}^D$ is the standard projection operator for our box-constraint: $p_{\mathcal{X}}(\mathbf{y}) = \arg\min_{\mathbf{z}\in\mathcal{X}} \|\mathbf{z}-\mathbf{y}\|_2$; see Figure 4.2. We still need to describe how REMBO chooses the bounded region $\mathcal{Y} \subset \mathbb{R}^d$, inside which it performs Bayesian optimization. This is important because REMBO's effectiveness depends on the size of $\mathcal{Y}$. Locating the optimum within $\mathcal{Y}$ is easier if $\mathcal{Y}$ is small, but if we set $\mathcal{Y}$ too small it may not actually contain the global optimizer. In the following theorem, we show that we can choose $\mathcal{Y}$ in a way that only depends on the effective dimensionality d_e such that the optimizer of the original problem is contained in the low dimensional space with constant probability.

Theorem 4. *Suppose we want to optimize a function $f : \mathbb{R}^D \to \mathbb{R}$ with effective dimension $d_e \leq d$ subject to the box constraint $\mathcal{X} \subset \mathbb{R}^D$, where $\mathcal{X}$ is centered around $\mathbf{0}$. Suppose further that the effective subspace $\mathcal{T}$ of f is such that $\mathcal{T}$ is the span of d_e basis vectors, and let $\mathbf{x}^\star_\top \in \mathcal{T} \cap \mathcal{X}$ be an optimizer of f inside $\mathcal{T}$. If $\mathbf{A}$ is a $D \times d$ random matrix with independent standard Gaussian entries, there exists an optimizer $\mathbf{y}^\star \in \mathbb{R}^d$ such that $f(\mathbf{A}\mathbf{y}^\star) = f(\mathbf{x}^\star_\top)$ and $\|\mathbf{y}^\star\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2$ with probability at least $1 - \epsilon$.*

Proof. Since $\mathcal{X}$ is a box constraint, by projecting $\mathbf{x}^\star$ to $\mathcal{T}$ we get $\mathbf{x}^\star_\top \in \mathcal{T} \cap \mathcal{X}$. Also, since

$\mathbf{x}^\star = \mathbf{x}^\star_\top + \mathbf{x}_\perp$ for some $\mathbf{x}_\perp \in \mathcal{T}^\perp$, we have $f(\mathbf{x}^\star) = f(\mathbf{x}^\star_\top)$. Hence, $\mathbf{x}^\star_\top$ is an optimizer. By using the same argument as appeared in Proposition 1, it is easy to see that with probability 1 $\forall \mathbf{x} \in \mathcal{T}$ $\exists \mathbf{y} \in \mathbb{R}^d$ such that $\mathbf{A}\mathbf{y} = \mathbf{x} + \mathbf{x}_\perp$ where $\mathbf{x}_\perp \in \mathcal{T}^\perp$. Let $\mathbf{\Phi}$ be the matrix whose columns form a standard basis for $\mathcal{T}$. Without loss of generality, we can assume that

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{I}_{d_e} \\ \mathbf{0} \end{bmatrix}$$

Then, as shown in Proposition 3, there exists a $\mathbf{y}^\star \in \mathbb{R}^d$ such that $\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{A}\mathbf{y}^\star = \mathbf{x}^\star_\top$. Note that for each column of $\mathbf{A}$, we have

$$\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a}_i \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{I}_{d_e} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}\right).$$

Therefore $\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{A}\mathbf{y}^\star = \mathbf{x}^\star_\top$ is equivalent to $\mathbf{B}\mathbf{y}^\star = \bar{\mathbf{x}}^\star_\top$ where $\mathbf{B} \in \mathbb{R}^{d_e \times d_e}$ is a random matrix with independent standard Gaussian entries and $\bar{\mathbf{x}}^\star_\top$ is the vector that contains the first d_e entries of $\mathbf{x}^\star_\top$ (the rest are 0's). By Theorem 3.4 of [187], we have

$$\mathbb{P}\left[\|\mathbf{B}^{-1}\|_2 \geq \frac{\sqrt{d_e}}{\epsilon}\right] \leq \epsilon.$$

Thus, with probability at least $1 - \epsilon$, $\|\mathbf{y}^\star\| \leq \|\mathbf{B}^{-1}\|_2\|\bar{\mathbf{x}}^\star_\top\|_2 = \|\mathbf{B}^{-1}\|_2\|\mathbf{x}^\star_\top\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2$. □

Theorem 4 says that if the set $\mathcal{X}$ in the original space is a box constraint, then there exists an optimizer $\mathbf{x}^\star_\top \in \mathcal{X}$ that is d_e-sparse such that with probability at least $1 - \epsilon$, $\|\mathbf{y}^\star\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2$ where $f(\mathbf{A}\mathbf{y}^\star) = f(\mathbf{x}^\star_\top)$. If the box constraint is $\mathcal{X} = [-1, 1]^D$ (which is always achievable through rescaling), we have with probability at least $1 - \epsilon$ that

$$\|\mathbf{y}^\star\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\sqrt{d_e}.$$

Hence, to choose $\mathcal{Y}$, we must ensure that the ball of radius d_e/ϵ, centred at the origin, lies inside $\mathcal{Y}$.

In practice, we have found that it is very unlikely that the optimizer falls on the corner of the box constraint, implying that $\|x^\star_\top\| < \sqrt{d_e}$. Thus setting $\mathcal{Y}$ too big may be unnecessarily wasteful. To improve our understanding of this effect, we developed a simulation study, in which we drew random Gaussian matrices, used them to map various potential optimizers $\mathbf{x}^\star_\top$ to their corresponding points $\mathbf{y}^\star_\top \in \mathcal{Y}$, and studied the norms of $\mathbf{y}^\star_\top$.

Assume for simplicity of presentation that $\mathcal{Y}$ is axis-aligned and d_e-dimensional (the argument applies when $d > d_e$). The section of the random matrix $\mathbf{A}$ that maps points in $\mathcal{Y}$ to $\mathcal{T}$ is a random Gaussian matrix of dimension $d_e \times d_e$. Let us call this section of

the matrix $\mathbf{B}$. Since random Gaussian matrices are rotationally invariant in distribution, we have for any orthonormal matrix $\mathbf{O}$ and a random Gaussian matrix $\mathbf{B}$, $\mathbf{OB} \overset{d}{=} \mathbf{B}$. That is, $\mathbf{OB}$ and $\mathbf{B}$ are equal in distribution. Similarly, for $\mathbf{B}^{-1}$, $\mathbf{OB}^{-1} = \left(\mathbf{BO}^T\right)^{-1} \overset{d}{=} \mathbf{B}^{-1}$. Therefore, $\mathbf{B}^{-1}$ is also rotationally invariant. Hence, $\|\mathbf{B}^{-1}\mathbf{x}_\top\|_\infty \overset{d}{=} \|\mathbf{B}^{-1}\mathbf{x}'_\top\|_\infty$ as long as $\|\mathbf{x}_\top\|_2 = \|\mathbf{x}'_\top\|_2$. Following this equivalence for the supremum norm of projected vectors, it suffices to choose a point with the largest norm in $[-1, 1]^{d_e}$ in our simulations. We chose $\mathbf{x}_\top = [1, 1, \cdots, 1]$.

We conducted simulations for several embedding dimensions, $d_e \in \{1, 2, \cdots, 50\}$, by drawing 10000 random Gaussian matrices and computing $\|\mathbf{B}^{-1}\mathbf{x}\|_\infty$. We found that with empirical probability above $1 - \epsilon$ (for decreasing values of ϵ), it was the case that

$$\|\mathbf{B}^{-1}\mathbf{x}\|_\infty < \frac{1}{\epsilon} \max\{\log(d_e), 1\}.$$

These simulations indicate that we could set $\mathcal{Y} = \left[-\frac{1}{\epsilon}\max\{\log(d_e), 1\}, \frac{1}{\epsilon}\max\{\log(d_e), 1\}\right]^{d_e}$. We did this in our experiments and in particular chose $\epsilon = \log(d)/\sqrt{d}$, so that $\mathcal{Y}$ was $[-\sqrt{d}, \sqrt{d}]^d$. Note that Theorem 4 is not useful for this choice, which suggests that there is room to improve this aspect of our theory.

Some careful readers may wonder about the effect of the extrinsic dimensionality D. In the following theorem, we show that given the same intrinsic dimensions, the extrinsic dimensionality does not have an effect at all; in other words, REMBO is invariant to the addition of unimportant dimensions.

Theorem 5 (Invariance to addition of unimportant dimensions)**.** *Let $f : \mathbb{R}^{d_e} \to \mathbb{R}$ and for any $D \in \mathbb{N}$, $D \geq d_e$, define $f_D : \mathbb{R}^D \to \mathbb{R}$ such that f_D adds $D - d_e$ truly unimportant dimensions to f: $f_D(\mathbf{z}) = f(\mathbf{z}_{1:d_e})$. Let $\mathbf{A}_1 \in \mathbb{R}^{D_1 \times d}$ and $\mathbf{A}_0 \in \mathbb{R}^{(D_2 - D_1)\times d}$ be random Gaussian matrices with $D_2 \geq D_1 \geq d$ and let $\mathbf{A}_2 = \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_0\end{bmatrix}$. Then, REMBO run using the same dimension $d \geq d_e$ and bounded region $\mathcal{Y}$ yields exactly the same function values when run with $\mathbf{A}_1$ on f_{D_1} as when run with $\mathbf{A}_2$ on f_{D_2}.*

Proof. We only need to show that for each $\mathbf{y} \in \mathbb{R}^d$, we have $f_{D_1}(\mathbf{A}_1\mathbf{y}) = f_{D_2}(\mathbf{A}_2\mathbf{y})$ since this step of REMBO (line 6 of Algorithm 6) is the only one that differs between the two algorithm runs. When this function evaluation step yields the same results for every $\mathbf{y} \in \mathbb{R}^d$, then the two REMBO runs behave identically since the algorithm is otherwise identical and deterministic after the selection of $\mathbf{A}$ in Step 1. Since $\mathbf{A}_2 = \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_0\end{bmatrix}$, we have $\mathbf{A}_2\mathbf{y} = \begin{bmatrix}\mathbf{A}_1\mathbf{y} \\ \mathbf{A}_0\mathbf{y}\end{bmatrix}$. Since $D_2 \geq D_1 \geq d_e$, the first d_e entries of this $D_2 \times 1$ vector $\mathbf{A}_2\mathbf{y}$ are the first d_e entries of $\mathbf{A}_1\mathbf{y}$. We thus have $f_{D_1}(\mathbf{A}_1\mathbf{y}) = f([\mathbf{A}_1\mathbf{y}]_{1:d_e}) = f([\mathbf{A}_2\mathbf{y}]_{1:d_e}) = f_{D_2}(\mathbf{A}_2\mathbf{y})$. □

Finally, we show that REMBO is also invariant to rotations in the sense that given different rotation matrices, running REMBO would result in the same distributions of observed function values. The argument is made concise in the following results.

Lemma 1. *Consider function $f : \mathbb{R}^D \to \mathbb{R}$. Let $f_{\mathbf{R}} : \mathbb{R}^D \to \mathbb{R}$ be such that $f_{\mathbf{R}}(\mathbf{x}) = f(\mathbf{R}\mathbf{x})$ for some orthonormal matrix $\mathbf{R} \in \mathbb{R}^{D\times D}$. Then, REMBO run in bounded region $\mathcal{Y}$ yields exactly the same sequence of function values when run with $\mathbf{A}$ on f as when run with $\mathbf{R}^{-1}\mathbf{A}$ on $f_{\mathbf{R}}$ for a matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$.*

Proof. REMBO uses f and $\mathbf{A}$ (resp. $f_{\mathbf{R}}$ and $\mathbf{R}^{-1}\mathbf{A}$) only in one spot (in line 6). Thus, the proof is trivial by showing that $f(\mathbf{A}\mathbf{y}_{t+1}) = f_{\mathbf{R}}(\mathbf{R}^{-1}\mathbf{A}\mathbf{y}_{t+1})$ through simple algebra:

$$f_{\mathbf{R}}(\mathbf{R}^{-1}\mathbf{A}\mathbf{y}_{t+1}) = f(\mathbf{R}\mathbf{R}^{-1}\mathbf{A}\mathbf{y}_{t+1}) = f(\mathbf{A}\mathbf{y}_{t+1}).$$

□

Theorem 6 (Invariance to rotations)**.** *Consider function $f : \mathbb{R}^D \to \mathbb{R}$. Let $f_{\mathbf{R}} : \mathbb{R}^D \to \mathbb{R}$ be such that $f_{\mathbf{R}}(\mathbf{x}) = f(\mathbf{R}\mathbf{x})$ for some orthonormal matrix $\mathbf{R} \in \mathbb{R}^{D\times D}$. Then, given random Gaussian matrices $\mathbf{A}_1 \in \mathbb{R}^{D\times d}$ and $\mathbf{A}_2 \in \mathbb{R}^{D\times d}$, REMBO run in bounded region $\mathcal{Y}$ yields in distribution the same sequence of function values when run with $\mathbf{A}_1$ on f as when run with $\mathbf{A}_2$ on $f_{\mathbf{R}}$.*

Proof. Since $\mathbf{R}$ is orthonormal, we have $\mathbf{R}^{-1}\mathbf{A}_1 \overset{d}{=} \mathbf{A}_2$. Therefore, REMBO run in bounded region $\mathcal{Y}$ yields in distribution the same sequence of function values when run with $\mathbf{R}^{-1}\mathbf{A}_1$ on $f_{\mathbf{R}}$ as when run with $\mathbf{A}_2$ on $f_{\mathbf{R}}$. We have also by Lemma 1 that REMBO run in bounded region $\mathcal{Y}$ yields exactly the same sequence of function values when run with $\mathbf{A}_1$ on f as when run with $\mathbf{R}^{-1}\mathbf{A}_1$ on $f_{\mathbf{R}}$. The conclusion follows from combining the previous arguments. □

4.1.1 Increasing the Success Rate of REMBO

Theorem 4 only guarantees that $\mathcal{Y}$ contains the optimum with probability at least $1-\epsilon$; with probability $\delta \leq \epsilon$ the optimizer lies outside of $\mathcal{Y}$. There are several ways to guard against this problem. One is to simply run REMBO multiple times with different independently drawn random embeddings. Since the probability of failure with each embedding is δ, the probability of the optimizer not being included in the considered space of k independently drawn embeddings is δ^k. Thus, the failure probability vanishes exponentially quickly in the number of REMBO runs, k. Note also that these independent runs can be trivially parallelized to harness the power of modern multi-core machines and large compute clusters.

Another way of increasing REMBO's success rate is to increase the dimensionality d it uses internally. When $d > d_e$, with probability 1 we have $\binom{d}{d_e}$ different embeddings of dimensionality d_e. That is, we only need to select d_e columns of $\mathbf{A} \in \mathbb{R}^{D \times d}$ to represent the d_e relevant dimensions of $\mathbf{x}$. The algorithm can achieve this by setting the remaining $d - d_e$ sub-components of the d-dimensional vector $\mathbf{y}$ to zero. Informally, since we have more embeddings, it is more likely that one of these will include the optimizer. In our experiments, we will assess the merits and shortcomings of these two strategies.

4.1.2 Choice of Kernel

Since REMBO uses GP-based Bayesian optimization to search in the region $\mathcal{Y} \subset \mathbb{R}^d$, we need to define a kernel between two points $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathcal{Y}$. In this chapter, we consider squared exponential kernel:

Definition 2. *Let $K_{\text{SE}}(\mathbf{y}) = \exp(-\|\mathbf{y}\|^2/2)$. Given a length scale $\ell > 0$, we define the corresponding squared exponential kernel as*

$$k_\ell^d(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = K_{\text{SE}}\left(\frac{\mathbf{y}^{(1)} - \mathbf{y}^{(2)}}{\ell}\right)$$

Note, all other kernels introduced in Section 1.4.2 can also be easily adopted here. We do not consider these other kernels here for notational simplicity. It is possible to work with two variants of the above kernel. First, we can use $k_\ell^d(\mathbf{y}^1, \mathbf{y}^2)$ as in Definition 2. We refer to this kernel as the low-dimensional kernel. We can also adopt an implicitly defined high-dimensional kernel on $\mathcal{X}$:

$$k_\ell^D(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = K_{\text{SE}}\left(\frac{p_{\mathcal{X}}(\mathbf{A}\mathbf{y}^{(1)}) - p_{\mathcal{X}}(\mathbf{A}\mathbf{y}^{(2)})}{\ell}\right),$$

where $p_{\mathcal{X}} : \mathbb{R}^D \to \mathbb{R}^D$ is the projection operator for our box-constraint as above (see Figure 4.2).

Note that when using this high-dimensional kernel, we are fitting the GP in D dimensions. However, the search space is no longer the box $\mathcal{X}$, but it is instead given by the much smaller subspace $\{p_{\mathcal{X}}(\mathbf{A}\mathbf{y}) : \mathbf{y} \in \mathcal{Y}\}$. Importantly, in practice it is easier to maximize the acquisition function in this subspace.

Both kernel choices have strengths and weaknesses. The low-dimensional kernel has the benefit of only requiring the construction of a GP in the space of intrinsic dimensionality d, whereas the high-dimensional kernel requires the GP to be constructed in a space

of extrinsic dimensionality D. However, the low-dimensional kernel may waste time exploring in the region of the embedding outside of $\mathcal{X}$ (see Figure 2) because two points far apart in this region may be projected via $p_{\mathcal{X}}$ to nearby points on the boundary of $\mathcal{X}$. The high-dimensional kernel is not affected by this problem because the search is conducted directly on $\{p_{\mathcal{X}}(\mathbf{Ay}) : \mathbf{y} \in \mathcal{Y}\}$ with distances calculated in $\mathcal{X}$ and not in $\mathcal{Y}$.

The choice of kernel also depends on whether our variables are continuous, integer or categorical. The categorical case is important because we often encounter optimization problems that contain discrete choices. We define our kernel for categorical variables as:

$$k_{\lambda}^{D}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \exp\left(-\frac{\lambda}{2} h(s(\mathbf{Ay}^{(1)}), s(\mathbf{Ay}^{(2)}))^2\right),$$

where $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathcal{Y} \subset \mathbb{R}^d$, the function s maps continuous d-dimensional vectors to discrete D-dimensional vectors, and h defines the distance between two discrete vectors. In more detail, $s(\mathbf{x})$ first uses $p_{\mathcal{X}}$ to project $\mathbf{x}$ to $\bar{\mathbf{x}} \in [-1, 1]^D$. For each dimension $\bar{x}_i$ of $\bar{\mathbf{x}}$, s then maps $\bar{x}_i$ to a discrete value by scaling and rounding. In our experiments, following [102], we defined $h(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = |\{i : x_i^{(1)} \neq x_i^{(2)}\}|$ so as not to impose an artificial ordering between the values of categorical parameters. In essence, we measure the distance between two points in the low-dimensional space as the Hamming distance between their mappings in the high-dimensional space.

4.1.3 Regret Bounds

In this section, we provide regret results for REMBO in the special case that (1) the embedded subspace has the same dimension as the effective dimension and (2) the embedded subspace contains a maximum of the function f inside the box $\mathcal{X}$. More specifically, here we will analyze a simplified version of the algorithm that performs Bayesian optimization only inside the box $\mathcal{X}$ rather than considering its extension beyond $\mathcal{X}$ and projecting onto the boundary of $\mathcal{X}$ as done in our actual implementation.

We acknowledge that this mismatch between the theoretical results and our actual algorithm is rather unsatisfactory. However, some of the obstacles that stand in the way of a complete analysis of the algorithm are currently insurmountable, since they would require the development of new tools that are far beyond the scope of this paper. We point these out at the end of this section and hope that our partial result will motivate the development of such tools, which might not otherwise receive any attention from the community.

We begin our mathematical treatment with the definitions of *simple regret* and the *skew squared exponential (SSE)* kernel.

Definition 3. *Given a function $f : \mathcal{X} \to \mathbb{R}$ and a sequence of points $\{\mathbf{x}_t\}_{t=1}^{\infty} \subseteq \mathcal{X}$, the* simple regret **with respect to the set** $\mathcal{X}$ *at time T is defined to be $r_f(T) = \sup_{\mathcal{X}} f - \max_{t=1}^{T} f(\mathbf{x}_t)$.*

Definition 4. *Given a symmetric, positive-definite matrix $\mathbf{\Delta}$ and K_{SE}, we define the corresponding* skew squared exponential *kernel as*

$$k_{\mathbf{\Delta}}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = K_{SE}\left(\mathbf{\Delta}^{-1/2}(\mathbf{y}^{(1)} - \mathbf{y}^{(2)})\right).$$

Given $\mathbf{\Delta}$, and $\mathcal{X} \subseteq \mathbb{R}^d$, we denote the Reproducing Kernel Hilbert Spaces (RKHSs) corresponding to $k_{\mathbf{\Delta}}$ by $\mathcal{H}_{k_{\mathbf{\Delta}}}$ [206, Definition 4.18]. Note that for the setting $\mathbf{\Delta} = \ell^2 I$ we recover the squared exponential kernel k_ℓ^d introduced earlier. To simplify notation, we introduce yet another definition.

Definition 5. *Consider the function $f : \mathcal{S} \to \mathbb{R}$, where $\mathcal{S} \subset \mathbb{R}^D$ is a d-dimensional subspace of $\mathbb{R}^D$. Let $\mathbf{\Phi} \in \mathbb{R}^{D \times d}$ be a matrix whose columns form an orthonormal basis for $\mathcal{S}$. We define the* canonical representation $\overline{f} : \mathbb{R}^d \to \mathbb{R}$ *of f as $\overline{f}(\mathbf{x}) = f(\mathbf{\Phi}\mathbf{x})$.*

Our main result below shows that the simple regret of a variation of REMBO vanishes with rate $\mathcal{O}(t^{-\frac{1}{d}})$ with high probability. This REMBO variant uses a fixed kernel parameter length scale ℓ and, more importantly, restricts its search to its embedding inside the box $\mathcal{X}$.

We only make the assumption that the cost function restricted to $\mathcal{T}$ is governed by a skew squared exponential kernel, a much weaker assumption than the standard assumption that the cost function is governed by an axis aligned kernel in D dimensions (see, e.g., [39] and Chapter 3). Despite the fact that the cost function restricted to $\mathcal{T}$ is governed by a skew squared exponential kernel, the result shows that we can control the regret using the low-dimensional squared exponential kernel.

Theorem 7. *Let $\mathcal{X} \subset \mathbb{R}^D$ be a compact subset with non-empty interior that is convex and contains the origin and $f : \mathbb{R}^D \to \mathbb{R}$, a function with effective dimension d. Suppose that the canonical representation of the restriction of f to its effective subspace $\mathcal{T}$, denoted $\overline{f|_{\mathcal{T}}}$, is an element of the RKHS $\mathcal{H}_{k_{\mathbf{\Delta}}}(\mathbb{R}^d)$ with $\mathbf{\Delta}$ symmetric and positive definite and also satisfying $0 < r^2 < \lambda_{\min}(\mathbf{\Delta}) \leq \lambda_{\max}(\mathbf{\Delta}) < R^2$ for constants r and R, where $\lambda_{\min}(\mathbf{\Delta})$ and $\lambda_{\max}(\mathbf{\Delta})$ are the extreme eigenvalues of $\mathbf{\Delta}$.*

Let $\mathbf{A}$ be a $D \times d$ matrix, whose elements are drawn from the normal distribution $\mathcal{N}(0, 1)$. Then, given any $\epsilon > 0$, we can choose a length-scale $\ell = \ell(\epsilon)$ such that running REMBO with kernel k_ℓ^d on the restriction of f to the image of $\mathbf{A}$ inside $\mathcal{X}$ has simple regret **with respect to the set** $\operatorname{Im} \mathbf{A} \cap \mathcal{X}$ *in $\mathcal{O}(t^{-\frac{1}{d}})$ with probability $1 - \epsilon$.*

This theorem does not follow directly from the results of [39], since the kernel is not aligned with the axes, both in the high-dimensional space and the lower dimensional embedding.

Please refer to Appendix C for the proof of this theorem. The general idea of the proof is as follows. If we have a squared exponential kernel k_ℓ, with a smaller length scale than a given kernel $k_{\mathbf{\Delta}}$, then an element f of the RKHS of $k_{\mathbf{\Delta}}$ is also an element of the RKHS of k_ℓ (see Lemma 24 in the Appendix for more details). So, when running expected improvement, one can safely use k_ℓ instead of $k_{\mathbf{\Delta}}$ as the kernel and still obtain a regret bound. Most of the proof is dedicated to finding a length scale ℓ that fits "underneath" our kernel, so we can replace our kernel with k_ℓ, to which we can apply the results of [39].

Note that in the above theorem we make the assumption that the embedded dimension and the effective dimension are equal to each other. Given bounds such as Proposition 1 of [55], we strongly believe that a similar result holds when the embedded dimension is higher than the effective dimension; however, the analysis of that setting remains elusive due to the fact that none of the methods available in the literature on regret bounds for Bayesian optimization algorithms can handle kernels that have flat dimensions (i.e. when $\mathbf{\Delta}$ is not positive-definite), and adapting them to such a case requires tools from statistics that have yet to be developed. Given that, theoretical bounds for the case with $d_e < d$ are outside the scope of this work.

Moreover, note that this theorem provides a sublinear regret result for REMBO with respect to the whole set $\mathcal{X}$ only in the situation that Im $\mathbf{A}$ intersects the maximum locus of the function f *inside* the set $\mathcal{X}$. Note that Theorem 4 provides a lower bound on the probability of this happening in the special case that the effective subspace is axis-aligned. Proving regret bounds for situations in which the image of $\mathbf{A}$ only contains a maximum outside of $\mathcal{X}$ would require dealing with the RKHS of non-stationary kernels, since the projection operator $p_{\mathcal{X}}$ can have non-constant Jacobian. This is related to the situation with treed GPs [82], with the additional, immensely complicating ingredient that is the continuity assumption imposed along the boundaries of the various partitions of the space (since $p_{\mathcal{X}}$ is continuous). Similar to the theory of Partial Differential Equations, where boundary conditions are the hardest part of the problem, we anticipate this modification to be a non-trivial, albeit very interesting, undertaking, and pose it as an open problem to the community.

Remark 1. *The above theorem would also hold for a class of stationary kernels which includes the popular Matérn kernel. For conciseness of presentation, we do not include this result, but refer the curious reader to [39] for more details.*

Algorithm 7 Bayesian Optimization with Hyper-parameter Optimization.

input Threshold t_σ.
input Upper and lower bounds $U > L > 0$ for hyper-parameter.
input Initial length scale hyper-parameter $\ell \in [L, U]$.

Initialize $C = 0$
for $t = 1, 2, \ldots$ **do**
  Find $\mathbf{x}_{t+1}$ by optimizing the acquisition function u: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} u(\mathbf{x}|\mathcal{D}_t)$.
  if $\sqrt{\sigma^2(\mathbf{x}_{t+1})} < t_\sigma$ **then**
    $C = C + 1$
  else
    $C = 0$
  end if
  Augment the data $\mathcal{D}_{t+1} = \{\mathcal{D}_t, (\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1}))\}$
  if $t \mod 20 = 0$ **or** $C = 5$ **then**
    if $C = 5$ **then**
      $U = \max\{0.9\ell, L\}$
      $C = 0$
    end if
    Learn the hyper-parameter by optimizing the log marginal likelihood by using DIRECT and CMA-ES: $\ell = \arg\max_{l\in[L,U]} \log p(\mathbf{f}_{1:t+1}|\mathbf{x}_{1:t+1}, l)$
  end if
end for

4.1.4 Hyper-parameter Optimization

For Bayesian optimization (and therefore REMBO), it is difficult to manually estimate the true length scale hyper-parameter of a problem at hand. To avoid any manual steps and to achieve robust performance across diverse sets of objective functions, in this chapter we adopt an the hyper-parameter optimization scheme proposed in chapter 3. Since in this chapter, we only consider deterministic observations for simplicity, we adopt a slight simplification of Algorithm 5.

Here, we propose to optimize the length scale parameter ℓ by maximizing the marginal likelihood subject to an upper bound U which is decreased when the algorithm starts exploiting too much. Full details are given in Algorithm 7. We say that the algorithm is exploiting when the standard deviation at the maximizer of the acquisition function $\sqrt{\sigma^2(\mathbf{x}_{t+1})}$ is less than some threshold t_σ for 5 consecutive iterations. Intuitively, this means that the algorithm did not emphasize exploration (searching in new parts of the space, where the predictive uncertainty is high) for 5 consecutive iterations. When this criterion is met, the algorithm decreases its upper bound U multiplicatively and re-optimizes the hyper-parameter subject to the new bound. Even when the criterion is not met the

hyper-parameter is re-optimized every 20 iterations. For each optimization of the acquisition function, the algorithm runs both DIRECT [111] and CMA-ES [88] and uses the result of the best of the two options. The astute reader may wonder about the difficulty of optimizing the acquisition functions as discussed in Chapter 2. For REMBO, however, we have not found the optimization of the acquisition function to be a problem since we only need to optimize it in the low-dimensional space and our acquisition function evaluations are cheap, allowing us tens of thousands of evaluations in seconds that (empirically) suffice to cover the low-dimensional space well. If necessary, BaMSOO, the innovation of which is completely orthogonal to REMBO, can be easily applied.

The motivation of this algorithm (as with Algorithm 5) is to rather error on the side of having too small a length scale: given a squared exponential kernel k_ℓ, with a smaller length scale than another kernel k, one can show that any function f in the RKHS characterized by k is also an element of the RKHS characterized by k_ℓ. Thus, when running expected improvement, one can safely use k_ℓ instead of k as the kernel of the GP and still preserve convergence [39]. We argue that (with a small enough lower bound L) the algorithm would eventually reduce the upper bound enough to allow convergence. Also, the algorithm would not explore indefinitely as L is required to be positive. In our experiments, we set the initial constraint $[L, U]$ to be $[0.01, 50]$ and set $t_\sigma = 0.002$.

We want to stress the fact that the above argument is only known to hold for a class of kernels over continuous domains (e.g. squared exponential and Matérn class kernels). Although we believe that a similar argument could be made for integer and categorical kernels, rigorous arguments concerning convergence under these kernels remain a challenge in Bayesian optimization.

4.2 Experiments

We now study REMBO empirically. We first use synthetic functions of small intrinsic dimensionality $d_e = 2$ but extrinsic dimension D up to 1 billion to demonstrate REMBO's independence of D. Then, we apply REMBO to automatically optimize the 47 parameters of a widely-used mixed integer linear programming solver and demonstrate that it achieves state-of-the-art performance. However, we also warn against the blind application of REMBO. To illustrate this, we study REMBO's performance for tuning the 14 parameters of a random forest body part classifier used by Kinect. In this application, all the $D = 14$ parameters appear to be important, and while REMBO (based on $d = 3$) finds reasonable solutions (better than random search and comparable to what domain experts

achieve), standard Bayesian optimization can outperform REMBO (and the domain experts) in such moderate-dimensional spaces. More optimistically, this random forest tuning application shows that REMBO does not fail catastrophically when it is not clear that the optimization problem has low effective dimensionality.

4.2.1 Experimental Setup

For all our experiments, we used a single robust version of REMBO that automatically sets its GP's length scale parameter as described in Section 4.1.4. The code for REMBO, as well as all data used in our experiments is publicly available at `https://github.com/ziyuw/rembo`.

Some of our experiments required substantial computational resources, with the computational expense of each experiment depending mostly on the cost of evaluating the respective black-box function. While the synthetic experiments in Section 4.2.2 only required minutes for each run of each method, optimizing the mixed integer programming solver in Section 4.2.4 required 4-5 hours per run, and optimizing the random forest classifier in Section 4.2.5 required 4-5 days per run. In total, we used over half a year of CPU time for the experiments in this paper. In the first two experiments, we study the effect of our two methods for increasing REMBO's success rate (see Section 4.1.1) by running different numbers of independent REMBO runs with different settings of its internal dimensionality d.

4.2.2 Bayesian Optimization in a Billion Dimensions

The experiments in this section employ a standard $d_e = 2$-dimensional benchmark function for Bayesian optimization, embedded in a D-dimensional space. That is, we add $D - 2$ additional dimensions which do not affect the function at all. More precisely, the function whose optimum we seek is $f(\mathbf{x}_{1:D}) = g(x_i, x_j)$, where g is the Branin function

$$g(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$$

and where i and j are selected once using a random permutation. To measure the performance of each optimization method, we used the *optimality gap*: the difference of the best function value it found and the optimal function value.

We evaluate REMBO using a fixed budget of 500 function evaluations that is spread across multiple interleaved runs — for example, when using $k = 4$ interleaved REMBO runs, each of them was only allowed 125 function evaluations. We study the choices of k and d by considering several combinations of these values. The results in Table 4.1

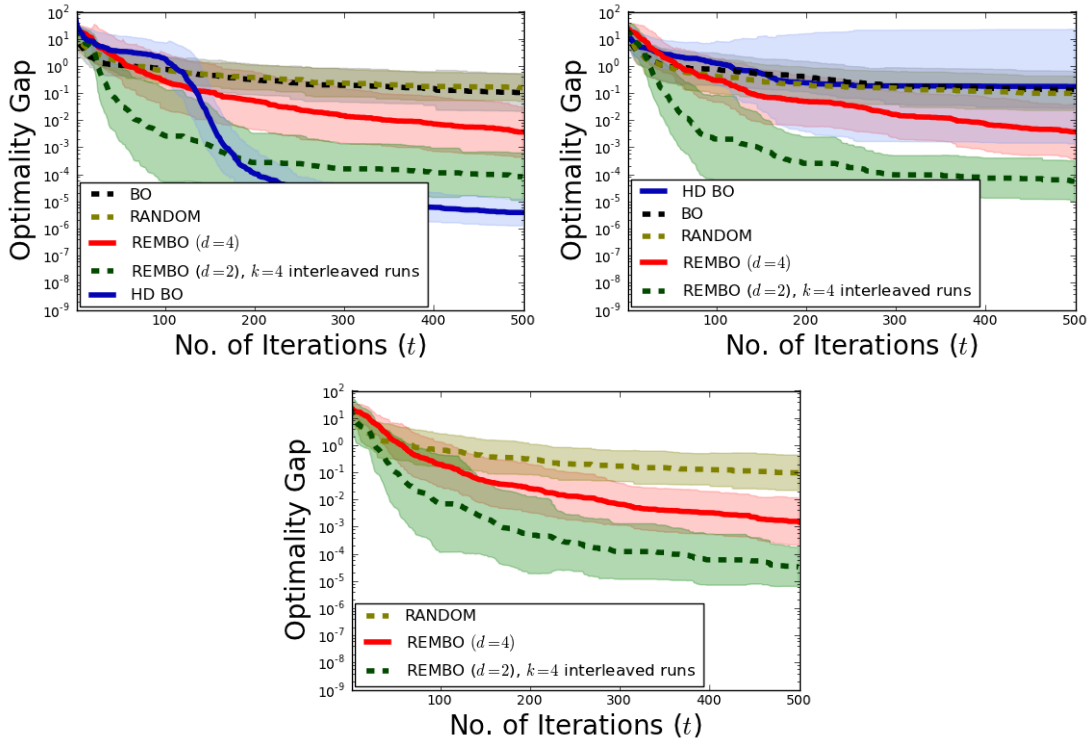


Figure 4.3: Comparison of random search (RANDOM), Bayesian optimization (BO), method by [47] (HD BO), and REMBO. Left: $D = 25$ extrinsic dimensions; Right: $D = 25$, with a rotated objective function; Bottom: $D = 10^9$ extrinsic dimensions. We plot means and $1/4$ standard deviation confidence intervals of the optimality gap across 50 trials.

demonstrate that interleaved runs helped improve REMBO's performance. We note that in 13/50 REMBO runs, the global optimum was indeed not contained in the box $\mathcal{Y}$ REMBO searched with $d = 2$; this is the reason for the poor mean performance of REMBO with $d = 2$ and $k = 1$. However, the remaining 37 runs performed very well, and REMBO thus performed well when using multiple interleaved runs: with a failure rate of 13/50=0.26 per independent run, the failure rate using $k = 4$ interleaved runs is only $0.26^4 \approx 0.005$. One could easily achieve an arbitrarily small failure rate by using many independent parallel runs. Using a larger d is also effective in increasing the probability of the optimizer falling into REMBO's box $\mathcal{Y}$ but at the same time slows down REMBO's convergence (such that interleaving several short runs loses its effectiveness).

Next, we compared REMBO to standard Bayesian optimization (BO) and to random search, for an extrinsic dimensionality of $D = 25$. Standard BO is well known to perform well in low dimensions, but to degrade above a tipping point of about 15-20 dimensions. Our results for $D = 25$ (see Figure 4.3, left) confirm that BO performed rather poorly

k	$d = 2$	$d = 4$	$d = 6$
10	0.0022 ± 0.0035	0.1553 ± 0.1601	0.4865 ± 0.4769
5	0.0004 ± 0.0011	0.0908 ± 0.1252	0.2586 ± 0.3702
4	0.0001 ± 0.0003	0.0654 ± 0.0877	0.3379 ± 0.3170
2	0.1514 ± 0.9154	0.0309 ± 0.0687	0.1643 ± 0.1877
1	0.7406 ± 1.8996	0.0143 ± 0.0406	0.1137 ± 0.1202

Table 4.1: Optimality gap for $d_e = 2$-dimensional Branin function embedded in $D = 25$ dimensions, for REMBO variants using a total of 500 function evaluations. The variants differed in the internal dimensionality d and in the number of interleaved runs k (each such run was only allowed $500/k$ function evaluations). We show mean and standard deviations of the optimality gap achieved after 500 function evaluations.

just above this critical dimensionality (merely tying with random search). REMBO, on the other hand, still performed very well in 25 dimensions.

One important advantage of REMBO is that — in contrast to the approach of [47] — it does not require the effective dimension to be coordinate aligned. To demonstrate this fact empirically, we rotated the embedded Branin function by an orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{D \times D}$. That is, we replaced $f(\mathbf{x})$ by $f(\mathbf{R}\mathbf{x})$. Figure 4.3 (middle) shows that REMBO's performance is not affected by this rotation.

Finally, since REMBO is independent of the extrinsic dimensionality D as long as the intrinsic dimensionality d_e is small, it performed just as well in $D = 1\,000\,000\,000$ dimensions (see Figure 4.3, right). To the best of our knowledge, the only other existing method that can be run in such high dimensionality is random search.

For reference, we also evaluated the method of [47] for these functions, confirming that it does not handle rotation gracefully: while it performed best in the non-rotated case for $D = 25$, it performed worst in the rotated case. It could not be used efficiently for more than $D = 1,000$. Based on a Mann-Whitney U test with Bonferroni multiple-test correction, all performance differences were statistically significant, except Random vs. standard BO. Finally, comparing REMBO to the method of [47], we also note that REMBO is much simpler to implement and that its results are very reliable (with interleaved runs).

4.2.3 Synthetic Discrete Experiment

In this section, we test the high-dimensional kernel with a synthetic experiment. Specifically, we again optimize the Branin function, but restrict its domain to 225 discrete points on a regular grid. As above, we added 23 additional irrelevant dimensions to make the problem 25-dimensional in total.

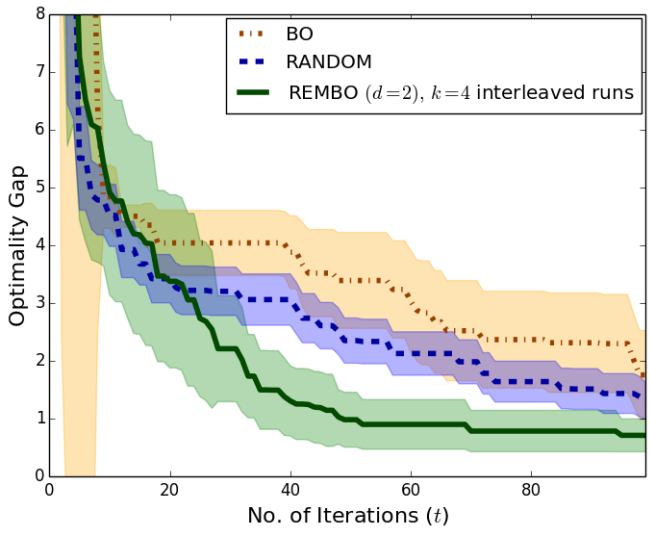


Figure 4.4: Comparison of random search (RANDOM), Bayesian optimization (BO), and REMBO. $D = 25$ extrinsic dimensions. We plot means and $1/4$ standard deviation confidence intervals of the optimality gap across 50 trials.

We used a small fixed budget of 100 function evaluations for all algorithms involved as the problem would require no more than 225 evaluations to be solved completely. We used $k = 4$ interleaved runs for REMBO. We again compare REMBO to random search and standard BO. For REMBO, we use the high-dimensional kernel to handle the discrete nature of the problem. The result of the comparison is summarized in Figure 4.4. Standard BO again suffered from the high extrinsic dimensionality and performed slightly worse than random search. REMBO, on the other hand, performed well in this setting.

4.2.4 Automatic Configuration of a Mixed Integer Linear Programming Solver

State-of-the-art algorithms for solving hard computational problems tend to parameterize several design choices in order to allow a customization of the algorithm to new problem domains. Automated methods for algorithm configuration have recently demonstrated that substantial performance gains of state-of-the-art algorithms can be achieved in a fully automated fashion [155, 109, 105, 214, 20, 221]. These successes have led to a paradigm shift in algorithm development towards the active design of highly parameterized frameworks that can be automatically customized to particular problem domains using optimization [99, 23, 211]. The resulting algorithm configuration problems have been shown to have low dimensionality [104], and here, we demonstrate that REMBO can exploit this low dimensionality even in the discrete spaces typically encountered in algorithm config-

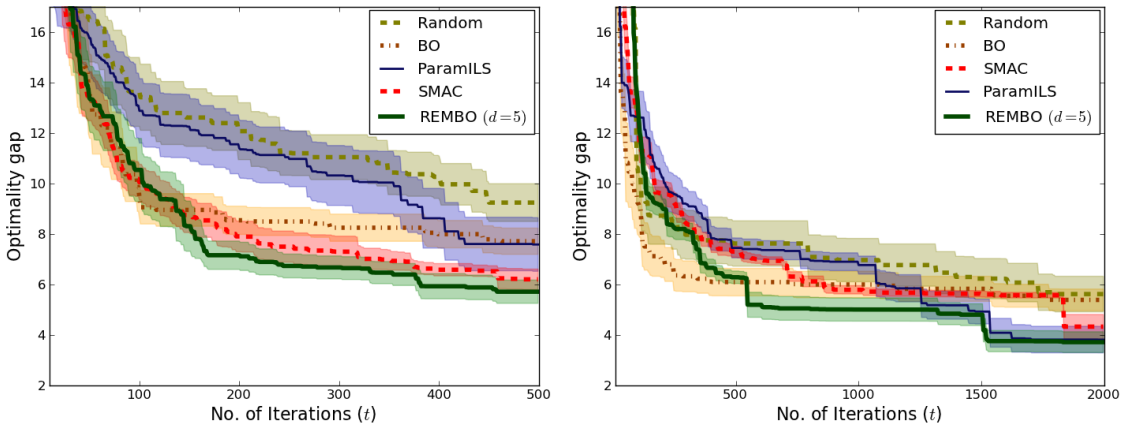


Figure 4.5: Performance of various methods for configuration of `lpsolve`; we show the optimality gap `lpsolve` achieved with the configurations found by the various methods (lower is better). Left: a single run of each method; Right: performance with $k = 4$ interleaved runs.

uration. We use a configuration problem obtained from [105], aiming to configure the 40 binary and 7 categorical parameters of lpsolve [24] , a popular mixed integer programming (MIP) solver that has been downloaded over 40 000 times in the last year. The objective is to minimize the optimality gap `lpsolve` can obtain in a time limit of five seconds for a MIP encoding of a wildlife corridor problem from computational sustainability [79]. Algorithm configuration usually aims to improve performance for a representative set of problem instances, and effective methods need to solve two orthogonal problems: searching the parameter space effectively and deciding how many instances to use in each evaluation (to trade off computational overhead and over-fitting). Our contribution is for the first of these problems; to focus on how effectively the different methods search the parameter space, we only consider configuration on a single problem instance.

Due to the discrete nature of this optimization problem, we could only apply REMBO using the high-dimensional kernel for categorical variables $k_\lambda^D(\mathbf{y}^{(1)}, \mathbf{y}^{(2)})$ described in Section 4.1.2. While we have not proven any theoretical guarantees for discrete optimization problems, REMBO appears to effectively exploit the low effective dimensionality of at least this particular optimization problem.

Figure 4.5 (left) compares BO, REMBO, and the baseline random search against ParamILS [109] and SMAC [107]. ParamILS and SMAC were specifically designed for the configuration of algorithms with many discrete parameters and define the current state of the art for this problem. Nevertheless, here SMAC and our vanilla REMBO method performed best. Based on a Mann-Whitney U test with Bonferroni multiple-test correction, they both

yielded statistically significantly better results than both Random and standard BO; no other performance differences were significant. The figure only shows REMBO with $d = 5$ to avoid clutter, but we did not optimize this parameter; the only other value we tried ($d = 3$) resulted in indistinguishable .

As in the synthetic experiment, REMBO's performance could be further improved by using multiple interleaved runs. However, as shown by [108], multiple independent runs can also improve the performance of SMAC and especially ParamILS. Thus, to be fair, we re-evaluated all approaches using interleaved runs. Figure 4.5 (right) shows that ParamILS and REMBO benefitted most from interleaving $k = 4$ runs. However, the statistical test results did not change, still showing that SMAC and REMBO outperformed Random and BO, with no other significant performance differences.

4.2.5 Automatic Configuration of Random Forest Kinect Body Part Classifier

We now evaluate REMBO's performance for optimizing the 14 parameters of a random forest body part classifier. This classifier closely follows the proprietary system used in the Microsoft Kinect [199] and is available at `https://github.com/david-matheson/rftk`.

We begin by describing some details of the dataset and classifier in order to build intuition for the objective function and the parameters being optimized. The data we used consists of pairs of depth images and ground truth body part labels. Specifically, we used 1 500 pairs of 320x240 resolution depth and body part images, each of which was synthesized from a random pose of the CMU mocap dataset. Depth, ground truth body parts and predicted body parts (as predicted by the classifier described below) are visualized for one pose in Figure 4.6 (left). There are 19 body parts plus one background class. For each of these 20 possible labels, the training data contained 25 000 pixels, randomly selected from 500 training images. Both validation and test data contained *all* pixels in the 500 validation and test images, respectively.

The random forest classifier is applied to one pixel P at a time. At each node of each of its decision trees, it computes the depth difference between two pixels described by offsets from P and compares this to a threshold. At training time, many possible pairs of offsets are generated at random, and the pair yielding highest information gain for the training data points is selected. Figure 4.6 (right) visualizes a potential feature for the pixel in the green box: it computes the depth difference between the pixels in the red box and the white box, specified by respective offsets u and v. At training time, u and v are drawn from two independent 2-dimensional Gaussian distributions, each of which is parameterized by its



Figure 4.6: Left: ground truth depth, ground truth body parts and predicted body parts; Right: features specified by offsets u and v.

Table 4.2: Parameter ranges for random forest classifier. For the purpose of optimization, the maximum tree depth and the number of potential offsets were transformed to log space.

Parameter	**Range**
Max. tree depth	[1 60]
Min. No. samples for non leaf nodes	[1 100]
No. potential offsets to evaluate	[1 5000]
Bootstrap for per tree sampling	[T F]

two mean parameters μ_1 and μ_2 and three covariance terms Σ_{11}, Σ_{12}, and Σ_{22} ($\Sigma_{21} = \Sigma_{12}$ because of symmetry). These constitute 10 of the parameters that need to be optimized, with range [-50,50] for the mean components and [1, 200] for the covariance terms. Low covariance terms yield local features, while high terms yield global features. Next to these ten parameters, the random forest classifier has four other standard parameters, outlined in Table 4.2. It is well known in computer vision that many of the parameters described here are important. Much research has been devoted to identifying their best values, but results are dataset specific, without definitive general answers.

The objective in optimizing these RF classifier parameters is to find a parameter setting that learns the best classifier in a given time budget of five minutes. To enable competitive performance in this short amount of time, at each node of the tree only a random subset of data points is considered. Also note that the above parameters do not include the number of trees T in the random forest; since performance improves monotonically in T, we created as many trees as possible in the time budget. Trees are constructed depth first and returned in their current state when the time budget is exceeded. Using a fixed budget results in a subtle optimization problem because of the complex interactions between the various

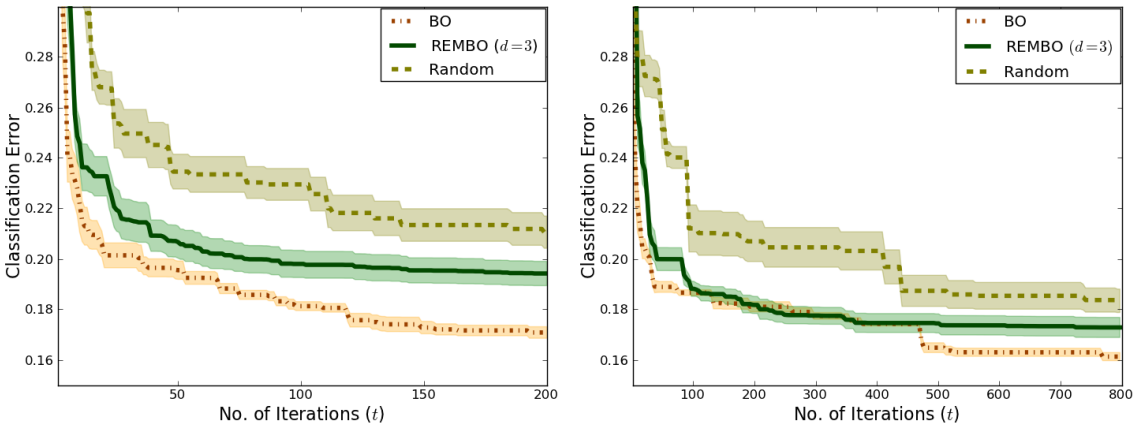


Figure 4.7: Performance of various methods for optimizing RF parameters for body part classification. For all methods, we show RF accuracy (mean $\pm$ 1/4 standard deviation across 10 runs) for all 2.2 million non background pixels in the 500-pose validation set, using the RF parameters identified by the method. The results on the test set were within 1% of the results on the validation set. Left: performance with a single run of each method; Right: performance with $k = 4$ interleaved runs.

parameters (maximum depth, number of potential offsets, number of trees and accuracy).

It is unclear a priori whether a low-dimensional subspace of these 14 interacting parameters exists that captures the classification accuracy of the resulting random forests. We performed large-scale computational experiments with REMBO, random search, and standard Bayesian optimization (BO) to study this question. In this experiment, we used the high-dimensional kernel for REMBO to avoid the potential over-exploration problems of the low-dimensional kernel described in Section 4.1.2. We believed that $D = 14$ dimensions would be small enough to avoid inefficiencies in fitting the GP in D dimensions. This belief was confirmed by the observation that standard BO (which operates in $D = 14$ dimensions) performed well for this problem.

Figure 4.7 (left) shows the results that can be obtained by a single run of random search, BO, and REMBO. Remarkably, REMBO clearly outperformed random search, even based on as few as $d = 3$ dimensions.[1] However, since the extrinsic dimensionality was "only" a moderate $D = 14$, standard Bayesian optimization performed well, and since it was not limited to a low-dimensional subspace it outperformed REMBO. Nevertheless, several REMBO runs actually performed very well, comparably with the best runs of BO. Consequently, when running $k = 4$ interleaved runs of each method, REMBO performed almost

[1]Due to the large computational expense of this experiment (in total over half a year of CPU time), we only performed conclusive experiments with $d = 3$; preliminary runs of REMBO with $d = 4$ performed somewhat worse than those with $d = 3$ for a budget of 200 function evaluations, but were still improving at that point.

as well as BO, matching its performance up to about 450 function evaluations (see Figure 4.7, right).

We conclude that the parameter space of this RF classifier does not appear to have a clear low effective dimensionality; since the extrinsic dimensionality is only moderate, this leads REMBO to perform somewhat worse than standard Bayesian optimization, but it is still possible to achieve reasonable performance based on as little as $d = 3$ dimensions.

4.3 Conclusion

We have demonstrated that it is possible to use random embeddings in Bayesian optimization to optimize functions of extremely high extrinsic dimensionality D provided that they have low intrinsic dimensionality d_e. Moreover, our resulting REMBO algorithm is coordinate independent and it only requires a simple modification of the original Bayesian optimization algorithm; namely multiplication by a random matrix. We proved REMBO's independence of D theoretically and empirically validated it by optimizing low-dimensional functions embedded in previously untenable extrinsic dimensionalities of up to 1 billion. We also theoretically and empirically showed REMBO's rotational invariance. Finally, we demonstrated that REMBO achieves state-of-the-art performance for optimizing the 47 discrete parameters of a popular mixed integer programming solver, thereby providing further evidence for the observation (already put forward by Bergstra, Hutter and colleagues) that, for many problems of great practical interest, the number of important dimensions indeed appears to be much lower than their extrinsic dimensionality.

We note that the central idea of our work – using an otherwise unmodified optimization procedure in a randomly embedded space – in principle could be applied to arbitrary optimization procedures. Evaluating the effciency of this technique for other procedures is an interesting topic for future work.

Chapter 5

An Entropy Search Portfolio for Bayesian Optimization

The key idea underlying Bayesian optimization is that one can place a prior over the space of probable functions before optimization. This prior can then be sequentially combined with data as it is observed in order to form an updated posterior over likely functions. In particular, the point selected by the algorithm at each iteration will explicitly depend on the posterior. While this general principle encompasses all of Bayesian optimization, naturally, how well the algorithm performs, i.e. how quickly it is able to find the global maximizer in practice, hinges on exactly *how* the posterior is utilized. We will refer to the method which maps observed data to a potential query point as an *acquisition strategy*. Please see Section 1.5 for a broad introduction.

One early acquisition strategy utilized in the literature, *Probability of Improvement (PI)*, selects the candidate point which has maximum probability of improving over the best point seen so far (the incumbent). The point with the highest probability of improvement, however, frequently lies close to the current incumbent. As a result, in practice the use of PI often exhibits very greedy behavior. Instead, one can take into account the *expected* improvement, resulting in the *Expected Improvement (EI)* strategy [156]. More recently a wide variety of more advanced techniques have been proposed which include Bayesian upper confidence bounds [205, 55] and information-theoretic approaches [219, 90]. However, no single acquisition strategy provides better performance over all problem instances. In fact, a frequent occurrence is that the optimality of various strategies can swap during the course of optimization. To remedy this [97] proposes the use of a portfolio containing multiple acquisition strategies. A key ingredient of this approach is the meta-criterion by which a portfolio method selects among different strategies. This function is analogous to an acquisition function, but at a higher level. Whereas acquisition functions assign utility

to points in the input space, a meta-criterion assigns utility to candidates suggested by base strategies.

Our contribution in this chapter is the proposal of an information-theoretic meta-criterion similar to that used by [219, 90], which we call the Entropy Search Portfolio (ESP). Information-theoretic methods aim to reduce the entropy of the location of the minimum of the black-box function. The estimation of the probability of any point being the minimizer as well as the entropy of this probability distribution often requires discretization of the state-space. Previous work [90] approached this problem using slice sampling of an acquisition function such as PI or EI. In practice, we have found this process to be inefficient and numerically unstable. To improve on this, we introduce a novel way of obtaining discrete search points by sampling directly from the probability distribution of a point being the minimizer following Bochner's Lemma [174].

By applying Bochner's Lemma, we are also able to draw and optimize samples of a Gaussian process over a continuous space $\mathcal{X}$ approximately. This gives us an important extension of Thompson sampling, a popular bandit strategy that has so far been confined to the multi-armed bandit setting (discrete space $\mathcal{X}$) [46, 98]. This gives us an additional acquisition strategy to include in our portfolio.

Finally, we provide empirical evidence that our approach results in gains not only over previous portfolio strategies but also over the fundamental strategies that make up the portfolio.

5.1 Entropy search over portfolios

As described in the previous chapters, we are interested in finding the global minimizer $\mathbf{x}_\star = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ of a function f over some bounded domain, typically $\mathcal{X} \subset \mathbb{R}^{\mathrm{d}^d}$. Also we use a Gaussian process (GP) prior for f [176]. In this chapter, as in the previous chapters, we focus on two commonly used kernels that we make use of are the squared exponential and Matérn[1] kernels

$$\kappa_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\tfrac{1}{2}r^2), \quad \text{where } r = (\mathbf{x} - \mathbf{x}')^\mathsf{T} \mathrm{diag}(\boldsymbol{\theta}^2)^{-1}(\mathbf{x} - \mathbf{x}'), \tag{5.1}$$

$$\kappa_{\text{MATÉRN5}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{5}r)(1 + \sqrt{5}r + \tfrac{5}{3}r^2). \tag{5.2}$$

Notice, in this chapter we deviate from previous notations for the hyper-parameters for notational convenience.

In this work we also make use of a portfolio, or collection of base strategies $\mathcal{A} = \{\alpha_k\}_{k=1}^K$. Algorithm 8 outlines the framework for Bayesian optimization with portfolios,

[1] Technically the Matérn is a class of kernels; here we use a smoothness parameter of $\frac{5}{2}$.

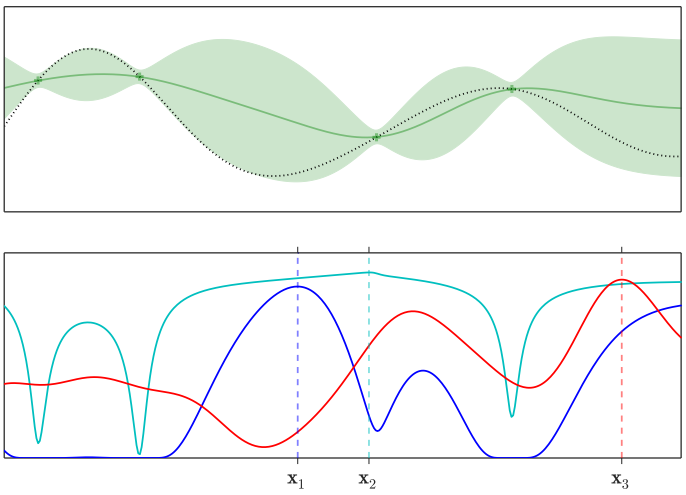


Figure 5.1: Visualization of Bayesian optimization with acquisition function portfolios. **Top panel:** visualization of our probabilistic model at a particular time. **Bottom panel:** plot of the three acquisition functions in this portfolio, in this case EI (blue), PI (cyan), and Thompson (red). The purpose of Algorithm 9 is to select between these "experts".

and Figure 5.1 provides a visualization to help follow the discussion. Each strategy can be thought of as an expert which recommends that its candidate point $\mathbf{x}_{kt} = \alpha_k(\mathcal{D}_t)$ be selected at iteration t. We denote this point as $\mathbf{x}_k$ when the time index is unambiguous. Our task is to select the most promising candidate according to some meta-criterion or MetaPolicy as it is denoted in Line 3 of Algorithm 8. In particular, the approach we introduce uses a probabilistic model of the location of the *unknown* global minimizer $\mathbf{x}_\star$. Given data $\mathcal{D}$, let

$$\mathbb{P}(\,\mathrm{d}\mathbf{x}_\star|\mathcal{D}) = \mathbb{P}\left(\arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \in \,\mathrm{d}\mathbf{x}_\star \middle| \mathcal{D}\right) \tag{5.3}$$

denote the posterior over minimizer locations, with density $p(\mathbf{x}_\star|\mathcal{D})$. This distribution is induced by our Gaussian process regression. In Section 5.1.1 we propose our meta-criterion $u(\mathbf{x}_k|\mathcal{D})$ which corresponds to the expected entropy of the distribution (5.3) after selecting input $\mathbf{x}_k$. In other words the candidate selected by this criterion is the one that results in the greatest decrease in uncertainty (in terms of entropy) about the minimizer. We will also see that an important step of this algorithm relies on sampling from (5.3). In Section 5.1.2 we show how to approximately sample from this distribution, an approach that also acts as an extension of Thompson sampling to continuous domains with GP priors. Finally, in Section 5.1.3 we summarize our selection strategy, the pseudocode for which is given in Algorithm 9. We also discuss how to effectively marginalize over GP hyper-parameters within this strategy.

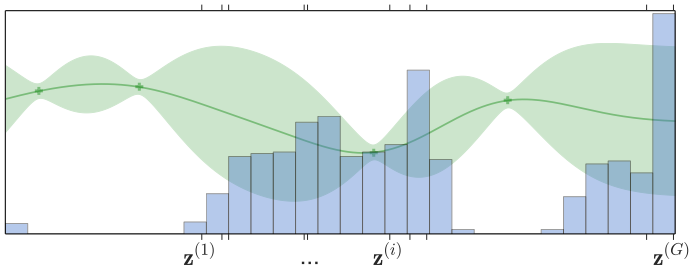


Figure 5.2: Histogram of 1000 approximate samples from pmin. Here we also show G draws from $p(\mathbf{x}_\star|\mathcal{D})$ that we denote $\mathbf{z}^{(i)}$ and fix for the remainder of this time step t and use in the METAPOLICY.

Algorithm 8 Portfolio Bayesian Optimization

Require: a noisy BLACKBOX function,
initial data $\mathcal{D}_0 = \{\}$
a portfolio $\mathcal{A} = \{\alpha_k\}$

for $t = 1, \ldots, T$ **do**
  collect candidates $\mathbf{x}_{kt} = \alpha_k(\mathcal{D}_{t-1})$
  set $\mathbf{x}_t = \text{METAPOLICY}(\{\mathbf{x}_{kt}\}, \mathcal{D}_{t-1})$
  set $y_t = \text{BLACKBOX}(\mathbf{x}_t)$
  augment data $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$
end for
Return $\tilde{\mathbf{x}}_T = \arg\max_{\mathbf{x}\in\mathcal{X}} \mu_T(\mathbf{x})$

Algorithm 9 METAPOLICY
Entropy Search Portfolio

Require: candidates $\{\mathbf{x}_k\}$, observations $\mathcal{D}$

$\mathbf{z}^{(i)} \sim p(\mathbf{x}_\star|\mathcal{D}), \quad i = 1, \ldots, G$
for $k = 1 : K$ **do**
  for $n = 1 : N$ **do**
    $y_k^{(n)} \sim p(y|\mathbf{x}_k, \mathcal{D})$
    $\mathcal{D}_k^{(n)} = \mathcal{D} \cup \{(\mathbf{x}_k, y_k^{(n)})\}$
    $\mathbf{f}_{kn}^{(s)} \sim p(\mathbf{f}|\tilde{\mathcal{D}}_k^{(n)})$ for $s = 1 : S$
    $\hat{p}_{ikn} = \frac{1}{S}\sum_s \mathbb{I}\left[i = \arg\min_j [\mathbf{f}_{kn}^{(s)}]_j\right]$
  end for
  $u_k = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{G} \hat{p}_{ikn} \log \hat{p}_{ikn}$
end for
Return $\mathbf{x}_{k_\star}$ **where** $k_\star = \arg\max_k u_k$

5.1.1 Entropy as a meta-criterion

We now consider our proposed selection criterion which corresponds to the expected entropy that arises from selecting the recommendation of the kth acquisition strategy. The steps of the algorithm can be followed in Algorithm 9.

Our meta-criterion corresponds to the expectation $\mathbb{E}_{p(y_k|\mathcal{D},\mathbf{x}_k)}[H[p(\mathbf{x}_\star|\tilde{\mathcal{D}}_k)]]$ where $\tilde{\mathcal{D}} = \mathcal{D} \cup \{(\mathbf{x}_k, y_k)\}$ is a random variable containing one additional fixed $\mathbf{x}_k$ and a corresponding random y_k distributed according to the posterior predictive distribution $p(y_k|\mathcal{D}, \mathbf{x}_k)$. For continuous densities p the differential entropy can be written as $H[p(\mathbf{x})] = -\int p(\mathbf{x}) \log p(\mathbf{x}) \, \mathrm{d}\mathbf{x}$. However, while the expectation with respect to the predictive distribution is easy to approximate via sampling and Monte Carlo integration, the entropy computation itself is more complicated due to the difficulty of evaluating $p(\mathbf{x}_\star|\tilde{\mathcal{D}}_k)$. Instead we approximate this density with a discrete distribution $\hat{p}$ restricted to a finite set of *representer points* denoted $\{\mathbf{z}^{(i)}\}_{i=1}^G$ sampled from some alternative measure. Not only should the distribution of $\{\mathbf{z}^{(i)}\}$ share the same support as the $p(\mathbf{x}_\star|\tilde{\mathcal{D}}_k)$, but in order to obtain good performance it is desirable for it to closely match this target distribution.

In the next section we show how to produce approximate samples from $p(\mathbf{x}_\star|\mathcal{D})$ directly rather than using a heuristic surrogate measure or even a regular grid as is proposed in previous work [90, 219]. This step corresponds to Line 1 of Algorithm 9 after which the $\{\mathbf{z}^{(i)}\}$ are kept fixed for the remainder of the algorithm.

We can now write our selection criterion as

$$u(\mathbf{x}_k|\mathcal{D}) = \mathbb{E}_{p(y_k|\mathcal{D},\mathbf{x}_k)}\left[H\left[\hat{p}(\mathbf{x}_\star|\tilde{\mathcal{D}}_k)\right]\right] \approx \frac{1}{N}\sum_{n=1}^{N} H\left[\hat{p}(\mathbf{x}_\star|\tilde{\mathcal{D}}_k^{(n)})\right] \tag{5.4}$$

where $H[\cdot]$ now represents the discrete entropy of its argument. Note that the Monte Carlo approximation given above is the result of the N data terms $\tilde{\mathcal{D}}_k^{(n)}$ each of which includes a single additional sample $y_k^{(n)} \sim p(\cdot|\mathbf{x}_k, \mathcal{D})$. As a result of our discrete approximation we can now write

$$H\left[\hat{p}(\mathbf{x}_\star|\tilde{\mathcal{D}}_k^{(n)})\right] = -\sum_i \hat{p}(\mathbf{x}_\star = \mathbf{z}^{(i)}|\tilde{\mathcal{D}}_k^{(n)}) \log \hat{p}(\mathbf{x}_\star = \mathbf{z}^{(i)}|\tilde{\mathcal{D}}_k^{(n)}) \tag{5.5}$$

for which we are left with the problem of computing $\hat{p}(\mathbf{x}_\star = \mathbf{z}^{(i)}|\tilde{\mathcal{D}}_k^{(n)})$. Recall, however, that $\hat{p}$ is the probability distribution over minimizers of a GP-distributed function where the minimizers are restricted to a finite set. This can easily be sampled exactly. Let the random variable $[\mathbf{f}_{kn}]_i = f(\mathbf{z}^{(i)})$, $i = 1, \ldots, G$, be a vector of latent function values evaluated at the representer points and conditioned on data $\tilde{\mathcal{D}}_k^{(n)}$. This vector simply has a Gaussian distribution and as a result we can produce S samples $\mathbf{f}_{kn}^{(s)} \sim p(\cdot|\tilde{\mathcal{D}}_k^{(n)})$ from the resulting

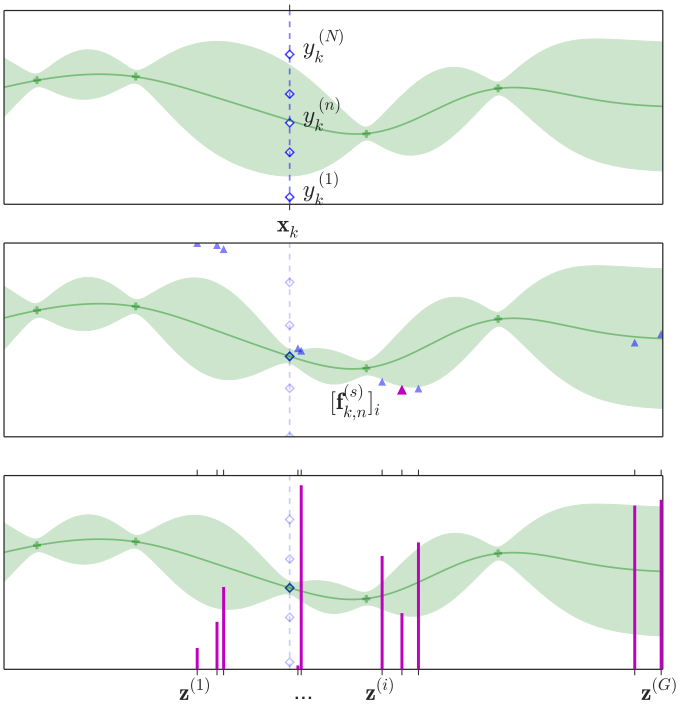


Figure 5.3: Visualization of the Entropy Search Portfolio. **Top panel:** for each candidate $\mathbf{x}_k$ (blue dashed line) we draw N hallucinations (blue diamonds) from the conditional. In practice this is done with quasi-Monte-Carlo to reduce variance. **Middle panel:** for each hallucination $y_k^{(n)}$ we augment the GP model (green line and shaded area) and sample it S times at the discrete points $\mathbf{z}^{(i)}$ to obtain the $\mathbf{f}_{kn}^{(s)}$ (blue triangles) for $s = 1, \ldots, S$. We find the minimizer of each vector $\mathbf{f}_{kn}^{(s)}$ (magenta triangle). **Bottom panel:** finally, we bin the S minimizers into a discrete empirical distribution $\hat{p}$ depicted here as a magenta histogram.

GP posterior. The probabilities necessary to compute the entropy can then be approximated by the relative counts

$$\hat{p}_{ikn} = \frac{1}{S} \sum_s \mathbb{I}\left[i = \arg\min_j [\mathbf{f}_{kn}^{(s)}]_j\right]$$

that $\mathbf{z}^{(i)}$ is the minimizer among the sampled functions. Finally, by combining these ideas we can write our entropy-based meta-criterion as

$$u(\mathbf{x}_k|\mathcal{D}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{G} \hat{p}_{ikn} \log \hat{p}_{ikn}.$$

Pseudocode computing this quantity is given in Algorithm 9 and we provide a corresponding visualization in Figure 5.3.

5.1.2 Sampling posterior global minimizers

The meta-criterion introduced in the previous section relied upon producing samples from (5.3), i.e. the posterior over global minima. While sampling from (5.3) is difficult in general, we can gain intuition by considering the finite domain setting. If the domain $\mathcal{X}$ is restricted to a finite set of m points, the latent function f takes the form of an m-dimensional vector $\mathbf{f}$. The probability that the ith element of $\mathbf{f}$ is optimal can then be written as $\int p(\mathbf{f}|\mathcal{D}) \prod_{j \neq i} \mathbb{I}[f_i \leq f_j] \, d\mathbf{f}$. This suggests the following generative process: i) draw a sample from the posterior distribution $p(\mathbf{f}|\mathcal{D})$ and ii) return the index of the maximum element in the sampled vector. This process is known as Thompson sampling or probability matching when used as an arm-selection strategy in multi-armed bandits [46]. This same approach could be used for sampling the maximizer over a continuous domain $\mathcal{X}$. At first glance this would require constructing an infinite-dimensional object representing the function f. To avoid this, one could sequentially construct f while it is being optimized. However, evaluating such an f would ultimately have cost $\mathcal{O}(m^3)$ where m is the number of function evaluations necessary to find the optimum. Instead, we propose to sample and optimize an analytic approximation to f. We will briefly derive this approximation below.

Given a shift-invariant kernel κ, Bochner's theorem [27] asserts the existence of its Fourier dual $s(\mathbf{w})$, which is equal to the spectral density of κ. Letting $p(\mathbf{w}) = s(\mathbf{w})/\alpha$ be the associated normalized density, we can write the kernel as the expectation

$$\kappa(\mathbf{x}, \mathbf{x}') = \alpha \, \mathbb{E}_{p(\mathbf{w})}[e^{-i\mathbf{w}^\mathsf{T}(\mathbf{x}-\mathbf{x}')}] = 2\alpha \, \mathbb{E}_{p(\mathbf{w},b)}[\cos(\mathbf{w}^\mathsf{T}\mathbf{x} + b)\cos(\mathbf{w}^\mathsf{T}\mathbf{x}' + b)] \,, \tag{5.6}$$

where $b \sim \mathcal{U}[0, 2\pi]$. Let $\boldsymbol{\phi}(\mathbf{x}) = \sqrt{2\alpha/m}\cos(\mathbf{W}\mathbf{x} + \mathbf{b})$ denote an m-dimensional feature mapping and where $\mathbf{W}$ and $\mathbf{b}$ consist of m stacked samples from $p(\mathbf{w}, b)$. The kernel κ can then be approximated by the inner product of these features, $\kappa(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\boldsymbol{\phi}(\mathbf{x}')$. This approach was used by [174] as an approximation method in the context of kernel methods. The feature mapping $\boldsymbol{\phi}(\mathbf{x})$ allows us to approximate the Gaussian process prior for f with a linear model $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\mathsf{T}\boldsymbol{\psi}$ where $\boldsymbol{\psi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard Gaussian. By conditioning on $\mathcal{D}$, the posterior for $\boldsymbol{\psi}$ is also multivariate Gaussian, $\boldsymbol{\psi}|\mathcal{D} \sim \mathcal{N}(\mathbf{A}^{-1}\boldsymbol{\Phi}^\mathsf{T}\mathbf{y}, \sigma^2\mathbf{A}^{-1})$ where $\mathbf{A} = \boldsymbol{\Phi}^\mathsf{T}\boldsymbol{\Phi} + \sigma^2\mathbf{I}$, $\mathbf{y}$ is the vector of the output data, and $\boldsymbol{\Phi}$ is a matrix of features evaluated on the input data.

Let $\boldsymbol{\phi}^{(i)}$ and $\boldsymbol{\psi}^{(i)}$ be a random set of features and the corresponding posterior weights sampled both according to the generative process given above. They can then be used to construct the function $f^{(i)}(\mathbf{x}) = \boldsymbol{\phi}^{(i)}(\mathbf{x})^\mathsf{T}\boldsymbol{\psi}^{(i)}$, which is an approximate posterior sample of f—albeit one with a finite parameterization. We can then maximize this function to obtain

$\mathbf{z}^{(i)} = \arg\min_{\mathbf{x}\in\mathcal{X}} f^{(i)}(\mathbf{x})$, which is approximately distributed according to $p(\mathbf{x}_\star|\mathcal{D})$. To produce samples from this process it is also necessary to derive the spectral density for any kernel of interest. This quantity is given by the kernel's Fourier transform $s(\mathbf{w}) = \frac{1}{(2\pi)^d}\int e^{i\mathbf{w}^\intercal\boldsymbol{\tau}}\kappa(\boldsymbol{\tau},\mathbf{0})\,\mathrm{d}\boldsymbol{\tau}$. For the squared-exponential and Matérn kernels we consider here this gives rise to weights which take the form

$$\mathbf{w}_{\text{SE}} \sim \mathcal{N}(0, \operatorname{diag}(\boldsymbol{\theta}^2)^{-1}) \quad \text{and} \quad \mathbf{w}_{\text{MATÉRN}} \sim \mathcal{T}(0, \operatorname{diag}(\boldsymbol{\theta}^2)^{-1}, \tfrac{5}{2}), \tag{5.7}$$

i.e. these are distributed as a normal and Student's T respectively, which are easy to sample from.

Finally, while this approach can be used to produce samples which implement the entropy meta-criterion of the previous section, it can also be used directly as an acquisition strategy. Consider the randomized strategy given by

$$\alpha_{\text{THOMPSON}}(\mathcal{D}) = \mathbf{x}_\star^{(0)} \quad \text{where} \quad \mathbf{x}_\star^{(0)} \sim p(\mathbf{x}_\star|\mathcal{D}).$$

Given the above description we can see that for finite $\mathcal{X}$ this is exactly the Thompson sampling algorithm. However, our derivation extends this approach to continuously varying functions.

5.1.3 Algorithm summary

In Algorithm 8 we describe general pseudocode for performing Bayesian optimization using a portfolio of acquisition functions. Here METAPOLICY denotes a general selection criterion such as the GP-HEDGE approach of [97]. In Algorithm 9 we summarize ESP, the criterion developed in this work. The cost of this approach will be dominated by solving the linear system on line 6 of Algorithm 9. As a result the complexity will be on the order $\mathcal{O}(KNSt^3)$, however this is only a constant-factor slowdown when compared to standard Bayesian optimization algorithms and can easily be parallelized.

In this chapter, we take a full Bayesian approach in marginalizing hyper-parameters as described in Section 1.6.1. Note, however, that the randomized Thompson sampling strategy, introduced in Section 5.1.2, only requires a single hyper-parameter sample as we can see this as a joint sample from the hyper-parameters and the function minimizer.

Our approach has additional complexity in that our candidate selection criterion depends on the hyper-parameter samples as well. This occurs explicitly in lines (4–6) of Algorithm 9 which sample from the GP posterior. This can be solved simply by adding an additional loop over the hyper-parameter samples. Our particular use of representer points in line 1 also depends on these hyper-parameters. We can solve this problem by equally allocating our representers between the M different hyper-parameter samples.

5.2 Experiments

In this section, we evaluate the proposed method on several problems: two synthetic test functions commonly used for benchmarking in global optimization, two real-world datasets from the geostatistics community, and two simulated control problems. We compare ESP against two other portfolio methods, namely the Hedge portfolio of [97] and RP, a random portfolio baseline which selects among its acquisition functions randomly.

All three portfolios consist of three commonly used base acquisition functions: EI, PI, and Thompson sampling. For EI we used the implementation available in the *spearmint* package[2], while the latter two were implemented in the same framework. All three methods were included in the portfolios. Note that we do not compare against GP-UCB (Gaussian Process Upper Confidence Bounds) [205] as the bounds do not apply directly when integrating over GP hyper-parameters. This would require an additional derivation which we do not consider here.

The performance of the three constituent acquisition functions are provided in our figures for three reasons. First, the results show that indeed no single acquisition function is the top performer across all six objective functions that we optimized. Second, the purpose of the portfolio was to mitigate against a poor choice of acquisition function; therefore our target performance should be the best performing constituent. Surprisingly however, we find that, in two cases[3], the portfolio beat the top acquisition function. Recall that the portfolios do not know, *a priori*, which of their constituents will perform the best. Finally, this is also the first evaluation of Thompson sampling in continuous search spaces, despite Thompson sampling being a popular strategy elsewhere.

5.2.1 Model details and hyper-parameter marginalization

As noted in Section 5.1.3, rather than obtain maximum likelihood point estimates of the GP hyper-parameters (*i.e.,* kernel length-scale and amplitude, and prior constant mean), we marginalize them using 10 Markov chain Monte Carlo (MCMC) samples after each observation. Both improvement-based methods, EI and PI, can simply be evaluated with each sample and averaged. Meanwhile, in keeping with the Thompson strategy, our Thompson implementation only uses a single sample, namely the last one of the MCMC chain. As reported above, our proposed ESP method splits its 500 representer points equally among the hyper-parameter samples. In other words, for each of the 10 MCMC samples we draw 50 points. In addition, in order to estimate $u(\mathbf{x}_k|\mathcal{D})$, for each of 5 simulated $y_k^{(n)}$ and each

[2] https://github.com/JasperSnoek/spearmint

[3] Branin and Brenda, and, to a lesser extent, Agromet.

GP hyper-parameter, ESP draws 1000 samples from $p(\mathbf{x}_\star|\tilde{\mathcal{D}}_k^{(n)})$, and then computes and averages the entropy estimates.

In the following experiments we evaluate the performance of each algorithm as a function of the number of function evaluations. When the true optimum is known, as in the case of the Branin and Hartmann 3 experiments, we measure performance by the absolute error. For all other minimization experiments, performance is given by the minimum function value obtained up to that iteration. Each experiment was repeated 25 times with different random seeds. For each method, we plot the mean performance over the repetitions as well as a shaded area one standard error away from the mean.

5.2.2 Global optimization test functions

We begin with two synthetic functions commonly used for benchmarking global optimization methods: Branin and Hartmann 3 [137]. They are two- and three-dimensional, respectively, and are both continuous, bounded, and multimodal. The experiments were run up to a final horizon of $T = 100$. Figure 5.4 reports the observed performance measured in absolute error on a logarithmic scale.

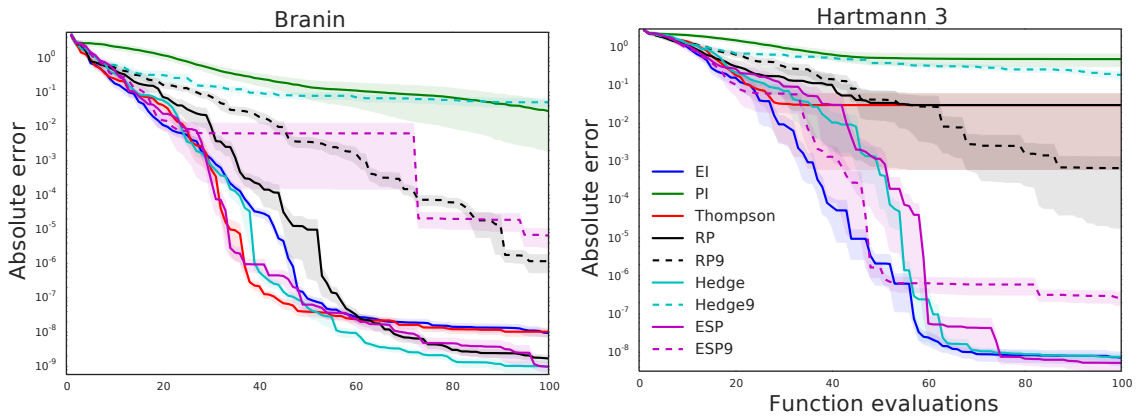


Figure 5.4: Absolute error of the best observation for the Branin and Hartmann 3 synthetic functions. The 9 additional random experts in RP9, GPHedge9, and ESP9 affect the RP and GPHedge methods much more dramatically than ESP on Hartmann 3. On Branin, a single outlier run out of 25 is negatively affecting the averaged results. When this run is masked, ESP9 achieves $< 10^{-4}$ error by evaluation 40.

It is interesting to note that despite the poor performance of PI in this task, all portfolios still eventually outperform the base strategies. As expected, ESP makes the best use of its base experts and outperforms all other methods in both examples.[4] In addition, note that

[4]Note that PI performs poorly in this multi-modal example because of PI's greediness. This issue could in principle be addressed using a *minimum improvement* parameter [112, 34]. However we set this parameter

each of the two domains have their own champion, EI being a clear winner on Hartmann 3, and Thompson being the more attractive option on Branin. This observation motivates the use of portfolios of acquisition functions.

In our experiments, we do not include information-theoretic acquisition functions as baselines. Previous results show, however, the current state-of-the-art information-theoretic acquisition function (PES; see Section 1.5.3) does better compared to other acquisition functions on Branin while doing significantly worse than EI and PES on Hartmann 3 [92]. This furthers the evidence that no single acquisition function is superior to all others. Since PES takes different approximations when computing the entropy than that proposed here, it is likely that including PES in the portfolio would lead to further improvements.

In these synthetic experiments, we also demonstrate the robustness of ESP with respect to the inclusion of poor base strategies. We do so by adding 9 random experts to each portfolio (we denote these ESP9, RP9, etc.). These so-called random experts select a point uniformly at random in the bounding box $\mathcal{X}$. We expect this sort of random search to be comparable to the other base methods in the initial stage of the optimization and eventually provide too much exploration and not enough exploitation. Note that a few random experts in a portfolio could actually be beneficial in providing a constant source of purely exploratory candidates; precisely how many, however, is an interesting question we do not discuss in the present work. Nevertheless, for the dimensionality and difficulty of these examples, we propose 9 additional random experts as being too many and indeed we observe empirically that they substantially deteriorate performance for all portfolios.

In particular, we see that, especially on Hartmann 3, ESP is virtually unaffected until it reaches 6 digits of accuracy. Meanwhile, the progress made by RP is hindered by the random experts which it selects as frequently as the legitimate acquisition functions. Significantly worse is the performance of GPHedge which, due to the initial success of the random experts, favors these until the horizon is reached. Note in contrast that ESP does not rely on any expert's past performance, which makes it robust to lucky guesses and time-varying expert performances. GPHedge could possibly be improved by tuning the reward function it uses internally but this is a difficult and problem dependent task. On Branin, a single outlier run is deteriorating the performance of ESP9. When masked, the mean performance achieves 10^{-4} by evaluation 40.

to its default value of 0 in our experiments since our focus in this work is in selecting between acquisition functions rather than tuning them. However, one could imagine in future work using parameterized families of strategies as a way of enlarging portfolios.

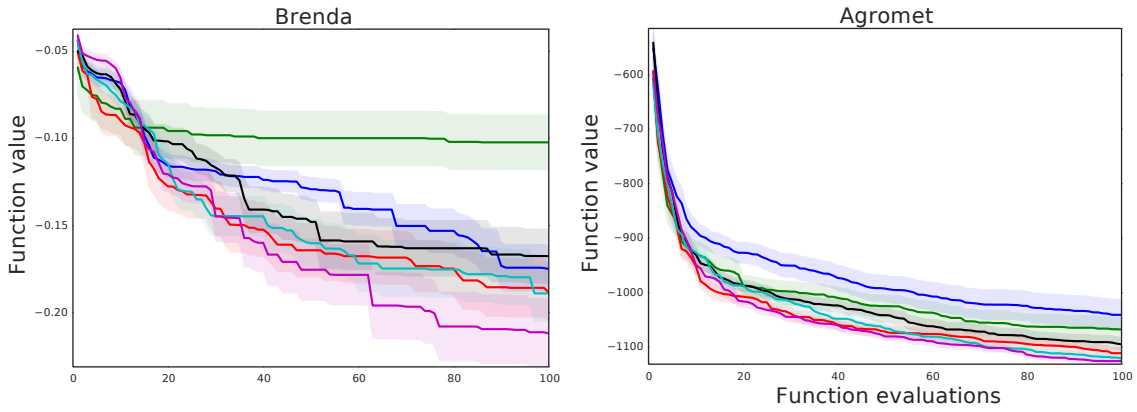


Figure 5.5: Best observed evaluation on mining datasets Brenda and Agromet. ESP outperforms the other portfolio methods while RP is dominated by all others.

5.2.3 Geostatistics datasets

The next set of experiments were carried out on two datasets from the geostatistics community, referred to here as Brenda and Agromet [50].[5] Since these datasets consist in finite sets of points, we transformed each of them into a function that we can query at arbitrary points via nearest neighbor interpolation. This produces a jagged piecewise constant function, which is outside the smoothness class of our surrogate models and hence a relatively difficult problem.

Brenda is a dataset of 1,856 observations in the depths of a copper mine in British Columbia, Canada, that was closed in 1990. The search domain is therefore three-dimensional and the objective function represents the quantity of underground copper. Agromet is a dataset of 18,188 observations taken in Natal Highlands, South Africa, where gold grade of the soil is the objective. The search domain here is two-dimensional.

We note that PI, which has so far been an under-achiever, outperforms EI on Agromet. This is further motivation for the use of portfolios, because our experience on the two synthetic experiments would have suggested that PI is a poor strategy. On both geological examples, RP is out-performed by GPHedge and ESP. We observe that ESP performs particularly well on Brenda, and on Agromet ESP is first, albeit by a very small margin. Recall that our original intention in using portfolios was to guard against choosing a poor acquisition function; however, on these examples ESP even outperforms the best of its constituent strategies (Thompson sampling in these two examples). This suggests that in addition to improving robustness to poor acquisition functions, portfolios could be used to boost overall performance.

[5]Both datasets can be found at `kriging.com/datasets`.

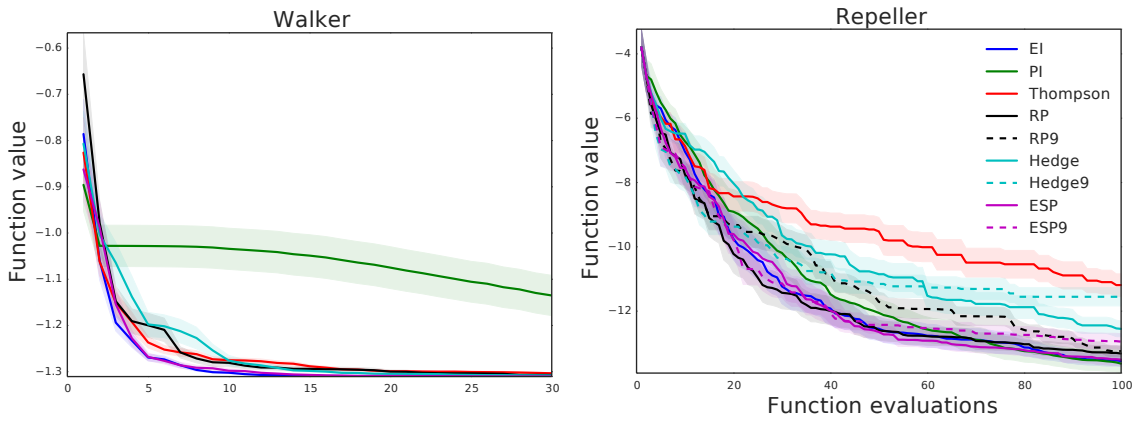


Figure 5.6: Best observed evaluation on the Walker and Repeller control tasks. Here we see ESP outperforming the other methods on Walker. On Repeller, ESP is once again competitive with the best methods and exhibits much more robustness to poor experts as evidenced by ESP9 following the performance of ESP so closely.

5.2.4 Control tasks

Our final experiments compare the portfolio methods on two control tasks. Walker is an eight-dimensional control problem where the inputs are fed into a simulator which returns the walking speed of a bipedal robot [227]. The code for the simulator is available online. This example was also used in a Bayesian optimization context by [92]. Repeller is a nine-dimensional control problem considered in previous work on Bayesian optimization portfolios [97, 96]. This problem simulates a particle in a two-dimensional world being dropped from a fixed starting region while accelerated downward by gravity. The task is to direct the falling particle through circular regions of low loss by placing 3 repellers with 3 degrees of freedom each, namely their position in the plane and repulsion strength. Figure 5.6 reports our results on both control tasks.

On these tasks, GPHedge performed poorly, while RP performs well. Meanwhile, ESP is the clear top performing portfolio method on Walker and competitive, if not tied, with RP on Repeller. We have also added the RP9, GPHedge9, and ESP9 methods in this experiment to once again demonstrate the robustness of ESP relative to its competitors. Indeed, both RP9 and GPHedge9 exhibit noticeably poorer performance than RP and GPHedge, while in contrast ESP9 is not significantly affected.

5.3 Conclusion

We revisited the idea of portfolios for Bayesian optimization. We demonstrated empirically that our proposed information-theoretic ESP meta-criterion can indeed have robust behav-

ior across functions of different dimensionality and modality even when the members of its portfolio do not exhibit this behavior. Furthermore, ESP is more robust to poorly performing experts than other portfolio mechanisms. We also provided a principled mechanism for sampling representer points as a way of approximating $p(\mathbf{x}_\star|\mathcal{D})$ which is more principled than previous approaches based on slice sampling from surrogate measures such as the expected improvement (EI). Finally, we showed that this sampling mechanism can itself be used as an acquisition strategy, thereby extending the popular Thompson sampling approach to continuous spaces with kernel methods; we include the first empirical evaluation of this method within our results.

Chapter 6

Adaptive Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (HMC) [62] is widely-known as a powerful and efficient sampling algorithm, having been demonstrated to outperform many existing MCMC algorithms, especially in problems with high-dimensional, continuous, and correlated distributions [48, 161]. Despite this flexibility, HMC has not been widely adopted in practice, due principally to the sensitivity and difficulty of tuning its hyper-parameters. In fact, tuning HMC has been reported by many experts to be more difficult than tuning other MCMC methods [110, 161]. In this chapter we aim to remove this obstacle in the use of HMC by providing an automated method for determining these tunable parameters, paving the way for a more widespread application of HMC in statistics and machine learning.

There are few existing works dealing with the automated tuning of HMC. Two notable approaches are: the No U-turn sampler (NUTS) [95], which is an adaptive algorithm for HMC that aims to find the best parameter settings by tracking the sample path and preventing HMC from retracing its steps in this path; and Riemann manifold HMC (RMHMC) [77], which provides adaptations using the Riemannian geometry of the problem.

In this chapter, we follow the approach of *adapting* Markov chains in order to improve the convergence of both HMC and RMHMC. Our adaptive strategy is based on Bayesian optimization. Bayesian optimization has been proposed previously for the adaptation of general MCMC samplers by [143] and [86]. To guarantee convergence, these works were limited to a finite adaptation of the Markov chain. This finite adaptation can result in the sampler being trapped in suboptimal parameter settings, leading to inefficient sampling, and we address this limitation in this chapter.

We describe Hamiltonian-based Monte Carlo samplers in section 6.1, and then make the following contributions:

- We present an algorithm for adaptive HMC in which we allow for *infinite adaptation* of the Markov chain, thus avoiding parameter traps due to finite adaptation (section 6.2).
- Importantly, we prove that the adaptive MCMC samplers we present are *ergodic* in this infinite adaptation setting (section 6.3).
- We provide a comprehensive set of experiments demonstrating that the adaptive schemes perform better across a diverse set of problems (section 6.4 and 6.5).

6.1 Hamiltonian-based Monte Carlo Sampling

Hamiltonian (or Hybrid) Monte Carlo [62, 161], has become established as a powerful, general purpose Markov chain Monte Carlo (MCMC) algorithm for sampling from general, continuous distributions. Its efficiency is due to the fact that it makes use of gradient information from the target density to allow for an ergodic Markov chain capable of large transitions that are accepted with high probability. This efficiency and flexibility is demonstrated by the wide range of applications to which HMC has been applied, including: Bayesian generalized linear models [110], Bayesian neural networks [162], Gaussian process regression and classification [176], exponential family PCA and factor analysis [157], and restricted Boltzmann machines [175], amongst others.

For HMC, we are required to specify a potential energy function, which is the log of the joint distribution we wish to sample from, $U(\mathbf{x}) = -\log p(\mathbf{x})$ and a kinetic energy function, most typically, $K(\mathbf{p}) = \mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}/2$, with momentum vector $\mathbf{p}$ and a positive definite mass matrix $\mathbf{M}$. For standard HMC, the mass matrix is set to the identity. We defer the technical details of HMC to existing work [161], and present only the algorithm here (Algorithm 10).

HMC requires the selection of two free parameters: a step-size ϵ and a number leapfrog steps L. The accepted guidance is to choose the step-size so as to ensure that the sampler's rejection rate is between 25%-35%. It is also preferable to have a large L, since this reduces the random walk behavior of the sampler [161], but too large an L results in unnecessary computation. In this chapter, we consider a slight variation of the HMC algorithm: instead of performing L leapfrog steps at each iteration, we perform a random number of leapfrog steps, chosen from the discrete uniform distribution over $\{1, \cdots, L\}$, i.e. $L_r \sim \mathcal{U}(1, L)$ steps. This approach amounts to using a mixture of L different HMC transition kernels, thus preserving detailed balance [4].

Algorithm 10 Hamiltonian Monte Carlo Algorithm

Given: M, L, ϵ, and $\mathbf{x}^1$.
for $t = 1, 2, \cdots$ **do**
  Sample $\mathbf{p}^t \sim \mathcal{N}(\mathbf{0}, M)$ and $L_r \sim \mathcal{U}(1, L)$
  Let $\mathbf{x}_0 = \mathbf{x}^t$ and $\mathbf{p}_0 = \mathbf{p}^t + \frac{\epsilon}{2} \frac{\partial U}{\partial \mathbf{x}}\big|_{\mathbf{x}_0}$
  for $l = 1, 2, \cdots, L_r$ **do**
    $\mathbf{x}_l = \mathbf{x}_{l-1} + \epsilon M^{-1} \mathbf{p}_{l-1}$
    $\mathbf{p}_l = \mathbf{p}_{l-1} + \epsilon \frac{\partial U}{\partial \mathbf{x}}\big|_{\mathbf{x}_l}$
  end for
  $\mathbf{p}_l = \mathbf{p}_{l-1} - \frac{\epsilon}{2} \frac{\partial U}{\partial \mathbf{x}}\big|_{\mathbf{x}_l}$
  Draw $\mathbf{u} \sim \mathcal{U}(0, 1)$
  if $\mathbf{u} < \min[1, e^{U(\mathbf{x}^t) + K(\mathbf{p}^t) - U(\mathbf{x}_l) - K(\mathbf{p}_l)}]$ **then**
    Let $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1}) = (\mathbf{x}_l, \mathbf{p}_l)$
  else
    Let $(\mathbf{x}^{t+1}, \mathbf{p}^{t+1}) = (\mathbf{x}^t, \mathbf{p}^t)$
  end if
end for

HMC is known to be highly sensitive to the choice of ϵ and L. We demonstrate HMC's sensitivity to these parameters by sampling from a bivariate Gaussian with correlation coefficient 0.99. We consider three settings

$$(\epsilon, L) = \{(0.16, 40), (0.16, 50), (0.15, 50)\}$$

and show the behavior of the sampler as well as the autocorrelation plot in figure 6.1. While the first setting exhibits good behavior and low auto-correlation, small changes to these settings results in poor mixing and high auto-correlation, as seen in the other graphs. Theoretical results concerning the optimal acceptance rate for HMC exist, having been described by [25] and [161], with both concluding a rate around 0.65 as optimal. Such results, however, would not help in choosing the best sampler out of the three in Figure 6.1, since all three samplers in this demonstration have an acceptance rate around 0.7, leaving little guidance for finding the most efficient sampler.

To address the problem of choosing these parameters, we will introduce a method for automatically and adaptively tuning the parameters of HMC, reducing the need for time-consuming, expert tuning. An existing approach for automatic tuning of HMC was introduced by [95], referred to as the No U-turn sampler (NUTS). NUTS allows for automatic tuning of both HMC's parameters by tuning the stepsize ϵ during the burn-in phase, after which it is fixed and the number of leapfrog steps is adjusted thereafter for every iteration. ϵ is chosen using a stochastic approximation method referred to as dual averaging, and L is chosen for every sample using a recursive algorithm in which the number of leapfrog steps is allowed to increase until the proposal trajectory taken by the sampler begins to move

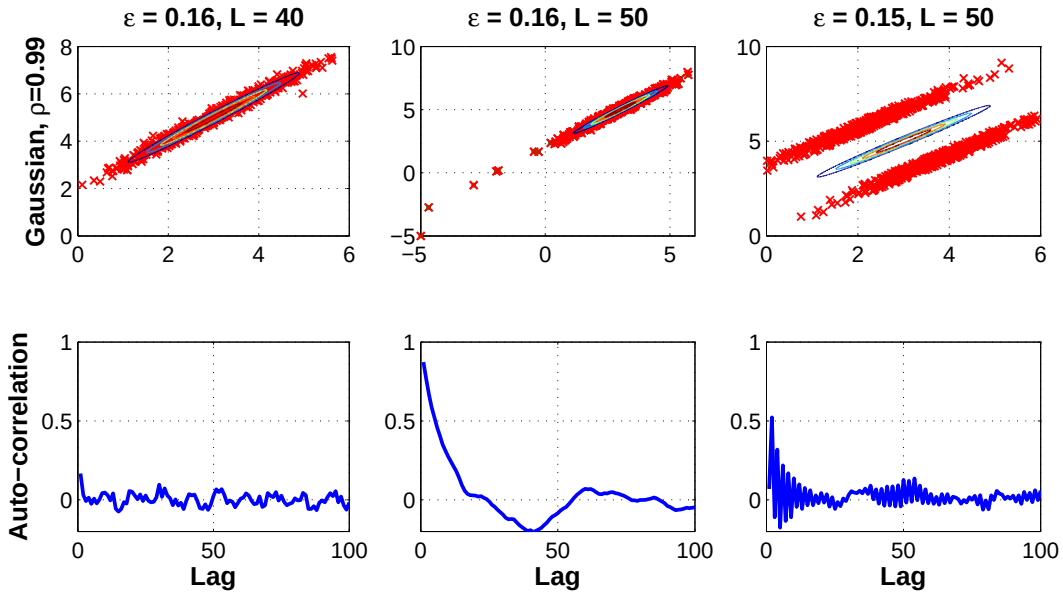


Figure 6.1: 1000 samples from a bivariate Gaussian distribution generated using HMC. We show the trajectory and auto-correlation of the samples for 3 parameter settings.

back towards its initial point, thus preventing U-turns and allowing for the good mixing of the chain.

Riemann manifold HMC (RMHMC) [77] is a sampling method derived from HMC, and provides an adaptation mechanism by exploiting the Riemannian geometry of the parameter space. Rather than adapting ϵ and L, RMHMC accounts for the local structure of the joint density by adapting the mass matrix $\mathbf{M}$ used in HMC. Since RMHMC automatically adapts its mass matrix, the stepsize ϵ is usually fixed and the number of leapfrog steps L, which is a single scalar, can be chosen using the rejection rate. While the sensitivity to these parameters is greatly reduced, they must still be set and there is no general guidance on how these parameters should be chosen, making it desirable to have a fully automatic method for RMHMC as well.

6.2 Adaptive Hamiltonian Monte Carlo

In order to adapt the MCMC parameters L and ϵ for HMC, we need to (i) define an objective function and (ii) choose a suitable optimization method.

As pointed out in [169], a natural objective function for adaptation is the asymptotic efficiency of an MCMC sampler, $(1 + 2\sum_{k=1}^{\infty} \rho_k)^{-1}$, where ρ_k is the auto-correlation of the sampler with lag k. Despite its appeal, this measure is problematic because the higher order auto-correlations are hard to estimate. To circumvent this problem, Pasarica and Gel-

man (2010) introduced an objective measure called the expected squared jumping distance (ESJD):

$$\text{ESJD}(\boldsymbol{\gamma}) = \mathbb{E}_{\boldsymbol{\gamma}} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2,$$

where $\boldsymbol{\gamma} = (L, \epsilon)$ denotes the set of parameters for HMC. Maximizing the above objective is equivalent to minimizing the first-order auto-correlation ρ_1. In practice, the above intractable expectation with respect to the Markov chain is approximated by an empirical estimator, as outlined in [169].

The ESJD measure is efficient in situations where the higher order auto-correlations increase monotonically with respect to ρ_1. However, it is not suitable for tuning HMC samplers since by increasing the number of leapfrog steps one can almost always generate better samples. What we need is a measure that also takes computing time into consideration. With this goal in mind, we introduce the following objective function:

$$f(\gamma) = \frac{\text{ESJD}(\boldsymbol{\gamma})}{\sqrt{L}} = \frac{\mathbb{E}_{\boldsymbol{\gamma}} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2}{\sqrt{L}}.$$

This function simply normalizes the ESJD by the number of leapfrog steps L, thus taking both statistical efficiency and computation into consideration. Most of our experiments will use this measure as we have found it to work very well in practice. Many works in the adaptive MCMC literature have considered matching empirical and theoretical acceptance rates in order to adapt MCMC samplers; see for example [5] or [218]. We have found this strategy to perform poorly in the case of HMC, where samplers with the same acceptance rate can exhibit different mixing behavior (figure 6.1).

Now that we are armed with an objective function, we need to address the issue of optimization. Since the objective is only available point-wise (that is, it can be evaluated but its exact form is intractable), researchers typically use stochastic approximation. We use Bayesian optimization to optimize the objective. A discussion contrasting these two alternatives is presented in [85].

Bayesian optimization is an efficient gradient-free optimization tool well suited for expensive black box functions. Our objective function (normalized ESJD) is of this nature. As mentioned earlier, normalized ESJD involves an intractable expectation that can be approximated by sample averages, where the samples are produced by running HMC for a few iterations. Each set of HMC samples for a specific set of hyper-parameters $\boldsymbol{\gamma} \in \Gamma$ results in a noisy evaluation of the normalized ESJD: $r(\boldsymbol{\gamma}) = f(\boldsymbol{\gamma}) + \varepsilon$, where we assume that the measurement noise is Gaussian $\varepsilon \sim \mathcal{N}(0, \sigma_\eta^2)$.

Following the standard Bayesian optimization methodology, we set Γ to be a box constraint such that

$$\Gamma = \{(\epsilon, L) : \epsilon \in [b_l^\epsilon, b_u^\epsilon], L \in [b_l^L, b_u^L]\}$$

Algorithm 11 Adaptive HMC.

Given: Γ, m, k, α, and $\boldsymbol{\gamma}_1$.
for $i = 1, 2, \ldots,$ **do**
  Run HMC for m iterations with $\boldsymbol{\gamma}_i = (\epsilon_i, L_i)$.
  Obtain the objective function value r_i using the drawn samples.
  Augment the data $\mathcal{D}_i = \{\mathcal{D}_{i-1}, (\boldsymbol{\gamma}_i, r_i)\}$.
  if $r_i > \sup_{j \in \{1,\cdots,i-1\}} r_j$ **then**
    $s = \frac{\alpha}{r_i}$
  end if
  Draw $u \sim \mathcal{U}([0, 1])$
  let $p_i = (\max\{i - k + 1, 1\})^{-0.5}$, with $k \in \mathbb{N}^+$.
  if $u < p_i$ **then**
    $\boldsymbol{\gamma}_{i+1} := \arg\max_{\boldsymbol{\gamma} \in \Gamma} u(\boldsymbol{\gamma}, s|\mathcal{D}_i)$.
  else
    $\boldsymbol{\gamma}_{i+1} := \boldsymbol{\gamma}_i$
  end if
end for

for some interval boundaries $b_l^\epsilon \leq b_u^\epsilon$ and $b_l^L \leq b_u^L$. The parameter L is discrete. The parameter ϵ is continuous, but since it is one-dimensional, we can discretize it using a very fine grid.

As in previous chapters, we use Gaussian process based Bayesian optimization. In this chapter, we adopt a squared exponential kernel function with

$$\kappa_{\text{SE}}(\boldsymbol{\gamma}_i, \boldsymbol{\gamma}_j) = \exp\left(-\frac{1}{2}\boldsymbol{\gamma}_i^T \Sigma^{-1} \boldsymbol{\gamma}_j\right)$$

where Σ is a positive definite matrix. We set $\Sigma = \text{diag}\left(\left[\alpha(b_u^\epsilon - b_l^\epsilon)\right]^2; \left[\alpha(b_u^L - b_l^L)\right]^2\right)$, where $\alpha = 0.2$. This hyper-parameter setting works well in our experiments and by not adapting the hyper-parameters, our algorithm runs considerably faster. For more complicated settings, hyper-parameters could be handled as in described in Section 1.6.1.

We adopt a variant of the Upper Confidence Bound (UCB) acquisition function [205], modified to suit our application:

$$\alpha(\boldsymbol{\gamma}, s|\mathcal{D}_i) = \mu_i(\boldsymbol{\gamma}, s) + p_i \beta_{i+1}^{\frac{1}{2}} \sigma_i(\boldsymbol{\gamma}).$$

As in standard UCB, we set $\beta_{i+1} = 2\log\left(\frac{(i+1)^{\frac{d}{2}+2}\pi^2}{3\delta}\right)$, where d is the dimension of Γ and δ is set to 0.1. The parameter p_i ensures that the diminishing adaptation condition for adaptive MCMC [179] is satisfied. Specifically, we set $p_i = (\max\{i - k + 1, 1\})^{-0.5}$ for some $k \in \mathbb{N}^+$. As p_i goes to 0, the probability of Bayesian optimization adapting $\boldsymbol{\gamma}$ vanishes as shown in Algorithm 11.

It could be argued that this acquisition function could lead to premature exploitation, which may prevent Bayesian optimization from locating the true optimum of the objective function. There is some truth to this argument. Our goal when adapting the Markov chain, however, is less about finding the absolute best hyper-parameters but more about finding sufficiently good hyper-parameters given finite computational resources. Given enough time, we could slow the annealing schedule thus allowing Bayesian optimization to explore the hyper-parameter space fully. However, under time constraints we must use faster annealing schedules. As p_i decreases, it becomes increasingly difficult for Bayesian optimization to propose new hyper-parameters for HMC. Consequently, the sampler ends up using the same set of hyper-parameters for many iterations. With this in mind, we argue, it is more reasonable to exploit known good hyper-parameters rather than exploring for better ones. This intuition matches our experience when conducting experiments.

The acquisition function also includes a scalar scale-invariance parameter s, such that $\mu_i(\boldsymbol{\gamma}, s) = \mathbf{k}^T(\mathbf{K} + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{r}_i s$. This parameter is estimated automatically so as to rescale the rewards to the same range each time we encounter a new maximal reward. Prior mean functions (see Section 1.4.3) could be adopted in place of s. We have found, however, for the task we consider in this chapter this simple trick is adequate.

Gaussian processes require the inversion of the covariance matrix and, hence, have complexity $\mathcal{O}(i^3)$, where i is the number of iterations. Fortunately, thanks to our annealing schedule, the number of unique points in our Gaussian process grows sub-linearly with the number of iterations. This slow growth makes it possible to adopt kernel specification techniques, as proposed by [65], to drastically reduce the computational cost without suffering any loss in accuracy.

Finally, in all our experiments, we set $\alpha = 4$, $k = 100$, $m = \frac{B}{k}$, where B is the number of burn-in samples. In our experience, the algorithm is robust with respect to these settings and we used the same set of parameters throughout our experiments with the exception of Γ. Γ is easy to set, since one can choose the bound to be large enough to contain all reasonable ϵ and L, while allowing the adaptive algorithm enough time to explore. Alternatively, one could gauge the hardness of the sampling problem at hand and set more reasonable bounds. In general, harder sampling problems require a smaller ϵ and a larger L. We follow this second strategy throughout our experiments and found that most sensible bounds led to performance similar to the ones reported.

6.3 Analysis of Convergence

The proof of ergodicity of the adaptive HMC algorithm capitalizes on existing results for Langevin diffusions and adaptive MCMC on compact state spaces. The method of proof is based on the standard Lyapunov stability functions, also known as drift or potential functions.

We assume that our target distribution is compactly supported on $\mathcal{M}$. In practice, for target distributions that are not compactly supported, we could set $\mathcal{M}$ large enough to contain most of the mass of our target distribution. The sampler is restricted to $\mathcal{M}$ by following this standard approach of rejecting all proposals that fall outside $\mathcal{M}$.

Let $\{P_\gamma\}_{\gamma\in\Gamma}$ be a collection of Markov chain kernels, each admitting π as the stationary distribution. That is, for each value of $\gamma = (\epsilon, L)$, we have one such kernel. Moreover, let P_γ^n denote the n-step Markov kernel. Our proof requires the following classical definitions:

Definition 6. (**Small set**) *A subset of the state space $C \subseteq \mathcal{X}$ is small if there exists $n_0 \in \mathbb{N}^+$, $\xi > 0$ and a probability measure $\nu(.)$ such that $P^{n_0}(x, \cdot) \geq \xi\nu(\cdot)\ \forall x \in C$.*

Definition 7. (**Drift condition**) *A Markov chain satisfies the drift condition if for a small set C, there exist constants $0 < \lambda < 1$ and $b < \infty$, and a function $V : \mathcal{X} \to [1, \infty]$ such that $\forall x \in \mathcal{X}$*

$$\int_{\mathcal{X}} P(x, dy)V(y) \leq \lambda V(x) + b\mathbf{1}_C(x).$$

Having defined the necessary concepts, we now move on to show the ergodicity of our adapted approach.

Proposition 1. *Suppose that P_γ, when restricted to a compact set $\mathcal{M}$, admits the stationary distribution π for all $\gamma \in \Gamma$. If π is continuous, positive and bounded on $\mathcal{M}$, and $|\Gamma|$ is finite, then the adaptive HMC sampler is ergodic.*

Proof. To show that adaptive HMC converges on a compact set, we first show that $\mathcal{M}$ is a small set.

The transition kernel of the random time HMC algorithm can be written as $P_\gamma(x, .) = \sum_{l=1}^{L} \frac{1}{L} Q_{l,\epsilon}(x, .)$ where $Q_{l,\epsilon}(x, .)$ is the transition kernel of an HMC sampler that takes l leapfrog steps with parameter ϵ. In particular $Q_{1,\epsilon}(x, .)$ is the transition kernel of Metropolis adjusted Langevin algorithm (MALA). As π is bounded, and the proposal distribution of MALA is positive every where, we have that $Q_{1,\epsilon}$ is μ^{Leb}-irreducible. By a slight modification of Theorem 2-2 in [181], for Markov chains defined by MALA, and any compact set

C with positive Lebesgue measure (i.e. $\mu^{Leb}(C) > 0$) there exists $\xi > 0$ and a probability measure $\nu(\cdot)$ such that $\forall x \in C\ Q^1_{1,\epsilon}(x, .) \geq \xi\nu(.)$. Hence, $\mathcal{M}$ is a small set since

$$P^1_\gamma(x, .) \geq \frac{1}{L} Q^1_{1,\epsilon}(x, .) \geq \frac{1}{L}\xi\nu(.)$$

for any compact set C where $\mu^{Leb}(C) > 0$. The drift condition is trivially satisfied by each HMC sampler when we choose C to be $\mathcal{M}$, and V to be such that $V(x) = 1$ for all x.

Having proved these conditions, we can now appeal to Theorem 15.0.1 of [151] to conclude that $\|P^n_\gamma(x, \cdot) - \pi(\cdot)\| < R_\gamma V(x)\rho^n_\gamma$ for all n and for $0 < \rho_\gamma < 1$. Since $V(X) = 1$ $\forall x$, we have

$$\|P^n_\gamma(x, \cdot) - \pi(\cdot)\| < R_\gamma \rho^n_\gamma.$$

Define $R_{max} = \sup_{\gamma\in\Gamma} R_\gamma$ and $\rho_{max} = \sup_{\gamma\in\Gamma} \rho_\gamma$, then $\forall x \in \mathcal{M}$ and $\forall\gamma \in \Gamma$ we have

$$\|P^n_\gamma(x, \cdot) - \pi(\cdot)\| < R_{max}\rho^n_{max}.$$

We have shown that the kernels $\{P_\gamma(x, \cdot)\}_{\gamma\in\Gamma}$ are simultaneously uniformly ergodic. Also, the adaptive HMC sampler has diminishing adaptation by design. By Theorem 5 of [179], these two conditions imply the claim of our proposition. □

In general two sets of conditions together guarantee ergodicity of an adaptive MCMC algorithm [179, 8]. First, the adaptation has to diminish eventually. The second set of conditions is usually placed on the underlying MCMC samplers. In [179], the samplers are required to be either simultaneously uniformly or geometrically ergodic. [8] weakened the conditions required, still requiring diminishing adaptation, but the requirements on the underlying MCMC samplers were reduced to sub-geometric ergodicity. Since the publication of our paper that this chapter is based on, new results on the ergodicity of HMC emerged (see, e.g., [26, 136]). Proof of convergence for adaptive HMC without the compactness assumption is therefore an interesting future research direction.

6.4 Results

We show the performance of our adaptive algorithm on four widely-used models. We evaluate the performance of the samplers using the effective sample size (ESS) using: $ESS = R\,(1 + 2\sum_k \rho_k)$, where R is the number of posterior samples, and $\sum_k \rho_k$ is the sum of K monotone sample auto-correlations computed using the monotone sequence estimator [77]. We adopt the total number of leapfrog steps used in producing the set of samples as a proxy for computational demand, since the computation is dominated by the gradient evaluation required for each leapfrog step. An efficient sampler will result in the

highest ESS for the least computation, and we will thus report the effective sample size per leapfrog step used (ESS/L), similarly to [95], since this takes into account computational requirements. We compute the ESS/L over all dimensions of the target distribution and report the minimum, median and maximum ESS obtained. While we report all three summary statistics, we focus on the *minimum ESS/L* as the most useful measure, since this allows us to evaluate the efficiency of the most confined coordinate, and is more indicative of ESS jointly over all coordinates rather than, as computed, over every coordinate independently [161, 77].

We compare our adaptive HMC to NUTS, and extend our approach and compare an adaptive version of RMHMC to the standard RMHMC. For NUTS, we tuned the free parameters of its dual averaging algorithm to obtain the best performance, and for RMHMC we use the experimental protocol and code used by [77]. We do this for all experiments in this section. Code to reproduce these results is available online[1].

6.4.1 Bayesian Logistic Regression

We consider a data set $\mathbf{X}$ consisting of N observations and D features or covariates, and a binary label $\mathbf{y}$. Using regression coefficients $\boldsymbol{\beta}$ and bias β_0 the joint distribution for the logistic regression model is:

$$\begin{aligned}\log p(\mathbf{X},\mathbf{y},\boldsymbol{\beta},\beta_0) &\propto \log p(\mathbf{y}|\mathbf{X},\boldsymbol{\beta},\beta_0)+\log p(\boldsymbol{\beta},\beta_0)\\ &= -\sum_i \log\left(1+\exp\left(-y_i(\beta_0+\mathbf{x}_i^\top\boldsymbol{\beta})\right)\right) - \frac{\beta_0^2}{2\sigma^2} - \frac{\boldsymbol{\beta}^\top\boldsymbol{\beta}}{2\sigma^2},\end{aligned} \tag{6.1}$$

where $y_i \in \{-1, 1\}$, and σ^2 is the prior variance of the regression coefficients. We present results on five data sets from the UCI repository. The data sets have varying characteristics with features D ranging from 2 to 24, and the number of observations N from 250 to 1000. For each data set, we generate 5000 samples after a burnin phase of 1000 samples, and repeat this process 10 times using differing starting points. The top row of figure 6.2 compares the performance of our adaptive HMC (AHMC) to NUTS, while the bottom row compares our adaptive RMHMC (ARMHMC) to RMHMC. For this experiment, for AHMC, we set Γ such that $\epsilon \in [0.01, 0.2]$ and $L \in \{0, \cdots, 100\}$, and for ARMHMC, we use $\epsilon \in [0.1, 1]$ and $L \in \{1, \cdots, 12\}$.

The columns of figure 6.2 show box plots of the minimum, median and maximum ESS/L values obtained. We see that the adaptive methods (AHMC and ARMHMC) exhibit good performance. For the minimum ESS/L, AHMC has better (higher) values that NUTS

[1] `https://bitbucket.org/ziyuw/ahmc`

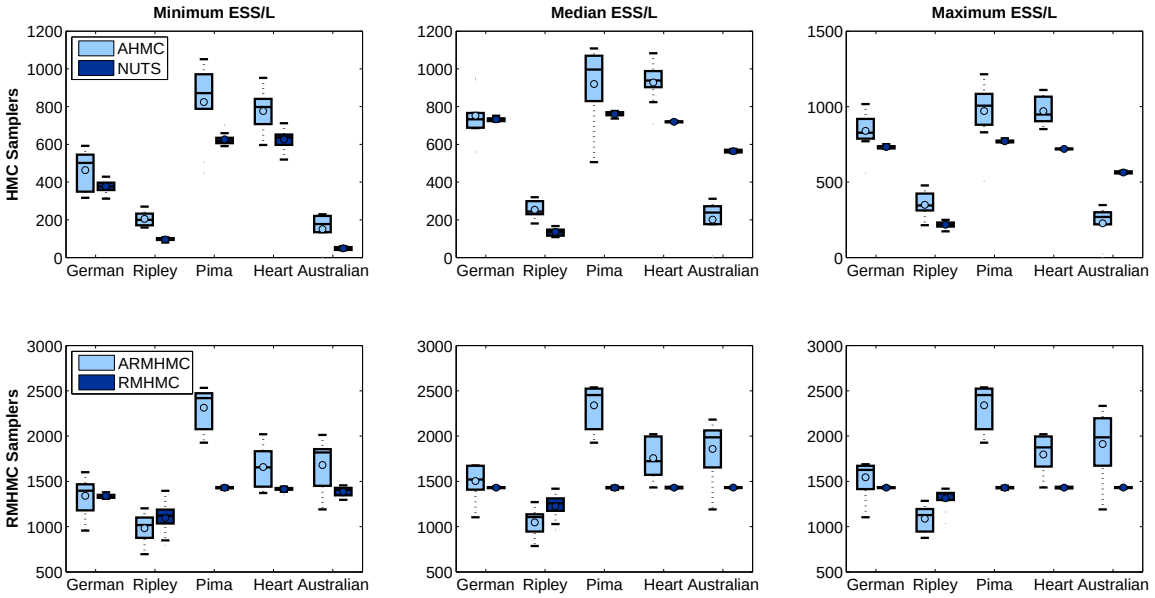


Figure 6.2: Box plots comparing ESS/L for Bayesian logistic regression. Top row: AHMC vs NUTS. Bottom row: ARMHMC vs RMHMC.

for all the data sets, and this behavior is consistent across most other data sets for the other summary statistics. Thus AHMC typically provides better performance and a higher effective number of samples per unit of computation used than NUTS. We also see that the ARMHMC can improve RMHMC and provide better ESS/L on what is already a highly efficient sampler.

6.4.2 Log-Gaussian Cox Point Process

We model a data set $\mathbf{Y} = \{y_{ij}\}$ that consists of counts at locations $(i, j), i, j = 1 \ldots, d$ in a regular spatial grid using a log-Gaussian Cox point process (LGC) [49, 77]. Observations y_{ij} are Poisson distributed and conditionally independent given a latent intensity process $\mathbf{\Lambda} = \{\lambda_{ij}\}$ with means $s\lambda_{ij} = s\exp(x_{ij})$, where $s = \frac{1}{d^2}$. The rates $\mathbf{X} = \{x_{ij}\}$ are obtained from a Gaussian process with mean function $m(x_{ij}) = \mu\mathbf{1}$ and covariance function $\Sigma(x_{ij}, x_{i'j'}) = \sigma^2 \exp\left(-\delta(i, i', j, j')/\beta d\right)$, where $\delta(i, i', j, j') = \sqrt{(i - i')^2 + (j - j')^2}$. The joint probability $\log p(\mathbf{y}, \mathbf{x}|\mu, \sigma, \beta)$ is proportional to:

$$\sum_{i,j} y_{ij}x_{ij} - d\exp(x_{ij}) - \frac{1}{2}(\mathbf{x} - \mu\mathbf{1})^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu\mathbf{1}). \tag{6.2}$$

We generate samples jointly for $\mathbf{x}, \sigma, \mu, \beta$ using a grid of size $d = 64$, using a synthetic data set obtained by drawing from the generative process for this model. We generate 5000 samples after a burnin of 1000 samples. For this model, we use $L \in \{1, \cdots, 500\}$,

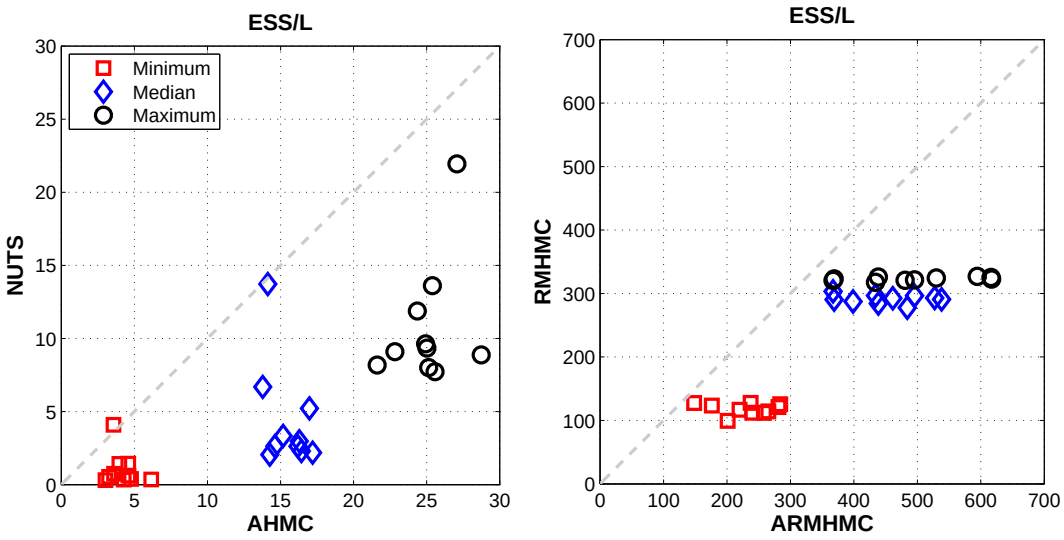


Figure 6.3: Comparing minimum (red), median (blue) and maximum (black) ESS/L for the Log-Gaussian Cox model. Each of the colored glyphs represents one of the 10 chains generated.

$\epsilon \in [0.001, 0.1]$ for AHMC, and use $L \in \{1, \cdots, 60\}$, $\epsilon \in [0.01, 1]$ for ARMHMC. We compare the performance of the adaptive method we presented in terms of ESS per leapfrog step in figure 6.3. We compare AHMC versus NUTS and ARMHMC versus RMHMC, showing the minimum, median and maximum ESS per leapfrog step obtained for 10 chains with dispersed starting points. We see that almost all points lie below the diagonal line, which indicates that the AHMC and ARMHMC have better ESS/L compared to NUTS and RMHMC, respectively. Thus even for high-dimensional models with strong correlations our adaptive method allows for automatic tuning of the sampler and consequently the ability to obtain higher quality samples than with competing methods.

We examine the quality of the posterior distribution obtained for AHMC and NUTS in figure 6.4, by visualizing the latent field and its variance, and comparing this to the true data (which is known for this data set). The top row shows the true latent fields. From the true data observations (shown in top right corner), we see that there are few data points in this region and thus we expect to have a high variance in this region. The average of the samples obtained using AHMC shows that we can accurately obtain samples from the latent field $\mathbf{x}$, and that the samples have a variance matching our expectations. While NUTS is able to also produce good samples of the latent field, the variance of the field is not well captured (bottom right image).

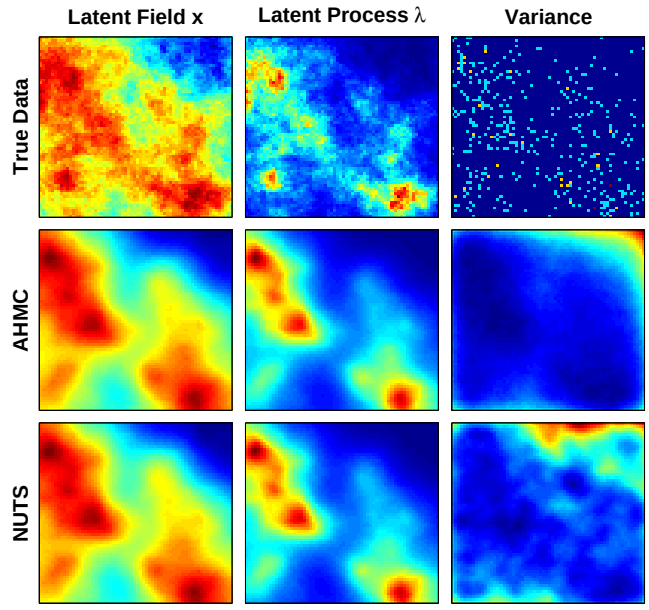


Figure 6.4: Comparing quality of posterior distributions from samples obtained using AHMC and NUTS for the log-Gaussian Cox model. The top-right image shows the locations of the true data.

6.4.3 Stochastic Volatility

We consider a stochastic volatility model described by [119] and [77], in which we consider observations y_t, regularly spaced in time for $t = 1, \ldots, T$. Each y_t is specified using a latent variable x_t, which represents the log-volatility following auto-regressive AR(1) dynamics. The model is specified as:

$$y_t = \epsilon_t \beta \exp\left(0.5 x_t\right), \qquad \epsilon_t \sim \mathcal{N}(0,1) \tag{6.3}$$

$$x_{t+1} = \phi x_t + \eta_{t+1}, \qquad \eta_{t+1} \sim \mathcal{N}(0, \sigma^2) \tag{6.4}$$

$$x_1 \sim \mathcal{N}\left(0, \frac{\sigma^2}{1-\phi^2}\right), \qquad p(\beta) \propto \frac{1}{\beta}. \tag{6.5}$$

For stationarity of the log-volatility, $|\phi| < 1$, and the standard deviation $\sigma > 0$, whose priors we set to $\frac{\phi+1}{2} \sim Beta(20, 1.5)$ and $\sigma^2 \sim$ inv-$\chi^2(10, 0.05)$, respectively. The parameters to be sampled by HMC is thus $\mathbf{\Theta} = \{\mathbf{x}, \beta, \phi, \sigma^2\}$, and the joint probability is:

$$p(\mathbf{y}, \mathbf{\Theta}) = \prod_{t=1}^{T} p(y_t|x_t, \beta) p(x_t|x_{t-1}, \phi, \sigma^2) p(\beta) p(\sigma^2) p(\phi). \tag{6.6}$$

We make use of the transformations $\sigma = \exp(\gamma)$ and $\phi = \tanh(\alpha)$ to ensure that we sample using unconstrained variables; the use of this transformation requires the addition of the Jacobian of the transformation of variables. We generate samples jointly using our

Table 6.1: Comparative results for the stochastic volatility model.

	ESS per Leapfrog		
Sampler	minimum	median	maximum
AHMC	1.3 ± 0.1	6.9 ± 0.7	14.9 ± 1.4
NUTS	0.7 ± 0.3	3.5 ± 1.6	9 ± 2.8

AHMC methods, using training data with $T = 2000$. For our experiments, we use a burnin period of $10,000$ samples and thereafter generate $20,000$ posterior samples. We restrict our box constraint such that $L \in \{1, \cdots, 300\}$, $\epsilon \in [10^{-4}, 10^{-2}]$. We show the results comparing ESS for the two methods in table 6.1. These results again show higher values for ESS per leapfrog step, demonstrating that a better performing sampler can be obtained using AHMC – further demonstrating the advantages of AHMC methods for sampling from complex hierarchical models.

6.5 Results on Bayesian Analysis of Continuous Time Markov Chains

In the previous section, we evaluated A(RM)HMC on standard test cases. In this section, we explore a novel application of AHMC.

Generalized Linear Model for Continuous Time Markov Chains (GLM-CTMC) is a recently developed model for analyzing continuous time, discrete state space time series. Previous applications of GLMs to continuous time Markov chains have been challenging due to the lack of effective MCMC algorithm. The authors of [232] overcame this challenge by applying auxiliary variable construction as well as AHMC. In this section, we evaluate AHMC on sampling from the posterior parameters of GLM-CTMC [232] and demonstrate its effectiveness.

The task we consider is the posterior parameter sampling of an amino acid evolutionary model with a synthetic dataset. For the construction of the model as well as the dataset, please refer to Section 7 of [232] as the descriptions are quite lengthy and the exact setup is orthogonal to our task.

As a baseline, we first perform experiments using a Metropolis-Hastings algorithm with a Normally distributed proposal distribution. Since we observed the performance of this baseline to be highly sensitive to the bandwidth of the proposal, we repeat the experiment with a wide range of bandwidths. We also run an adaptive MCMC algorithm from [180], namely a NMH algorithm based on a joint Normal proposal distribution with an empirical covariance estimated from the MCMC run. The proposal distribution is given at iteration

n by

$$Q_n(\mathbf{x}, \cdot) = \begin{cases} N(\mathbf{x}, (0.1)^2 I_d/d) & \text{for} \quad n \leqslant 2d, \\ (1-\beta)N(\mathbf{x}, (2.38)^2 \Sigma_n/d) + \beta N(\mathbf{x}, (0.1)^2 I_d/d) & \text{for} \quad n > 2d, \end{cases} \tag{6.7}$$

where $\mathbf{x}$ is the estimate in the previous iteration, d is the dimension of the parameters, I_d is an identity matrix of dimension d and Σ_n is the current empirical estimate of the covariance matrix based on the run so far and β is a small positive constant (we take β=0.05, as suggested in [180]). It is known from [177] and [178] that the proposal $N(x, (2.38)^2\Sigma/d)$ is optimal in a particular large-dimensional context. We compared these baseline samplers to AHMC. In this section, for computational efficiency, a hard limit of 3000 adaptation steps is imposed. After this horizon, the parameters ϵ and L are fixed.

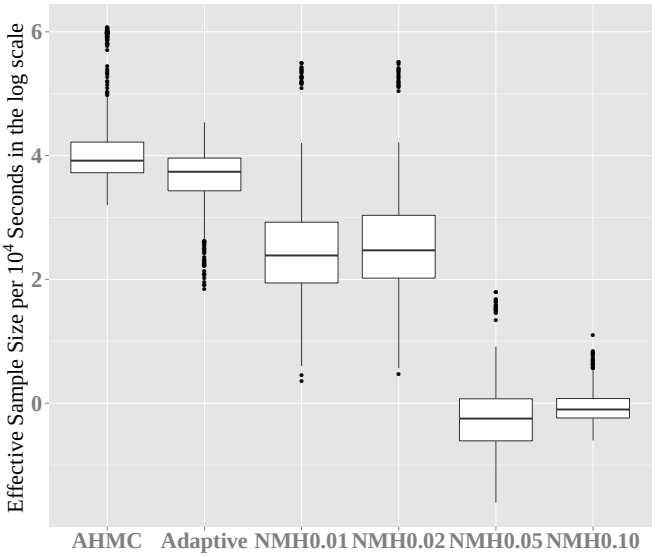


Figure 6.5: Effective sample size per 10^4 seconds in the log scale using AHMC, an adaptive NMH with a joint proposal guided by an estimated covariance matrix and NMH with bandwidths of 0.01, 0.02, 0.05, 0.10, where NMH x stands for a Normal proposal Metropolis Hastings algorithms with proposal bandwidth x.

We quantitatively evaluate the computational efficiency by monitoring the ESS per second to compute which we use the package from [172]. We calculate the ESS per second for each feature weight and report the extrema and quantile ESS per second across the different feature weights. We repeat each experiment four times and average the results. We show the results in Figure 6.5. All results are scaled by 10^4 seconds for ease of presentation. See Table 6.2 for results on a wider range of different bandwidth. Our method achieves a higher efficiency compared to classical NMH kernels with all the bandwidth tested, as well

as compared to the adaptive NMH algorithm. Even when comparing our sampler to the "oracle" NMH baseline, the minimum ESS per second is superior by an order of magnitude, and the quantiles, by a factor of at least two. Our sampler outperforms adaptive NMH by a factor of four in terms of the minimum ESS per second.

Method	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
AHMC	**24.56**	**41.42**	**50.32**	**86.01**	**67.90**	**432.20**
Adaptive NMH	6.32	30.91	40.07	41.84	52.45	93.34
NMH 0.002	0.17	0.53	0.97	1.54	1.53	17.77
NMH 0.005	0.51	1.70	2.47	4.89	3.99	74.39
NMH 0.01	1.44	6.97	10.88	20.25	18.62	244.20
NMH 0.0125	0.71	3.53	5.65	9.86	9.25	117.70
NMH 0.015	0.80	4.01	6.49	11.24	10.78	128.10
NMH 0.018	1.23	6.77	11.05	19.03	18.29	226.30
NMH 0.02	1.60	7.54	11.82	20.98	20.80	245.50
NMH 0.03	1.32	3.97	6.16	11.69	10.28	162.90
NMH 0.04	0.40	1.44	2.18	3.61	3.73	38.49
NMH 0.05	0.20	0.54	0.78	0.97	1.07	6.02
NMH 0.06	0.36	0.81	1.08	1.25	1.44	6.56
NMH 0.07	0.28	0.65	0.83	0.89	1.06	2.47
NMH 0.08	0.31	0.73	0.95	1.04	1.25	3.39
NMH 0.09	0.25	0.53	0.67	0.77	0.93	2.78
NMH 0.10	0.55	0.79	0.90	0.97	1.08	3.03
NMH 0.15	0.20	0.26	0.30	0.36	0.41	0.81
NMH 0.20	0.27	0.31	0.35	0.38	0.43	0.63

Table 6.2: ESS per 10^4 seconds. NMH x stands for a Normal proposal Metropolis Hastings algorithms with proposal bandwidth x.

Next, we investigate the effectiveness of our Bayesian optimization adaptation scheme. Again for the detailed experimental setup, please refer to Section 7 of [232]. We obtain $L = 31$ and $\epsilon = 0.02473$ after adaptation. To assess the quality of these values, we repeat the same experiment several times without adaptation, on a grid of parameters centered at $L = 31$ and $\epsilon = 0.02473$; see Figure 6.6. Each cell in the plot reflects the average over ESS per second across all feature weights, displayed in the log scale. The results show that the performance of the combination of tuning parameters identified by our adaptation scheme is close to optimal. This result is non-trivial as the objective Bayesian optimization optimizes is not the same objective we used in evaluation.

Moreover, we found that the cost of the adaptive Bayesian optimization is negligible in all cases we encountered. While the cost depends on the total number of optimization steps, we found that a small number of steps was sufficient in practice (Figure 6.6), with a cost on the order of 2–3 minutes. By comparison the wall-clock time for running 100,000 MCMC iterations was 2,262 minutes.

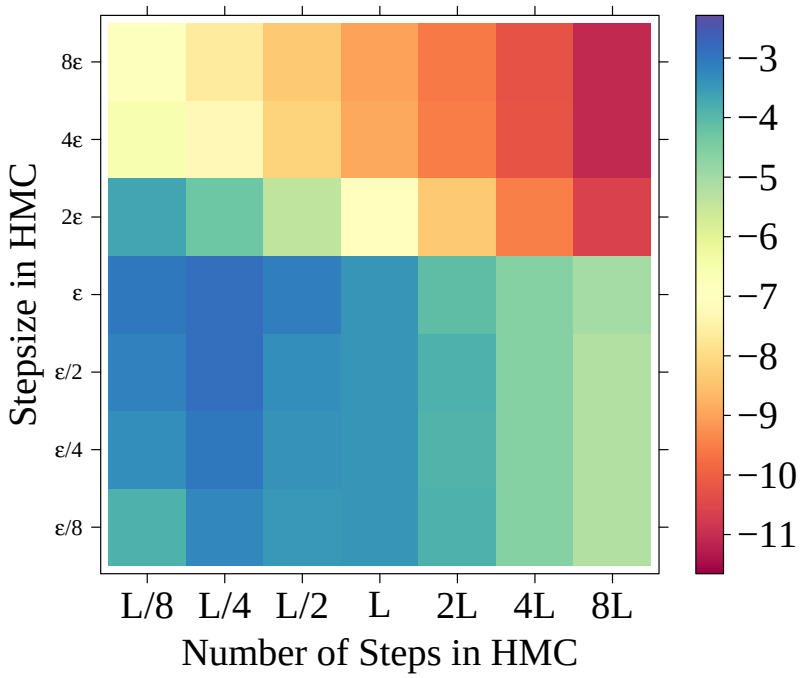


Figure 6.6: ESS per second (displayed in log scale) across all features obtained by running HMC with tuning parameters coming from a grid centered at the values ϵ and L coming from AHMC.

6.6 Conclusion

We addressed the widely-experienced difficulty in tuning Hamiltonian-based Monte Carlo samplers by developing algorithms for infinite adaptation of these Markov chains using Bayesian optimization. The adaptive Hamiltonian Monte Carlo and adaptive Riemann manifold HMC we developed automate the process of finding the best parameters that control the performance of the sampler, removing the need for time-consuming and expert-driven tuning of these samplers. Our experiments show conclusively that over a wide range of models and data sets, the use of adaptive algorithms makes it easy to obtain more efficient samplers, in some cases precluding the need for more complex approaches. Hamiltonian-based Monte Carlo samplers are widely known to be an excellent choice of MCMC method, and we hope that this chapter removes a key obstacle towards the more widespread use of these samplers in practice.

Chapter 7

Conclusion

This work primarily focuses on advancing the field of Bayesian optimization in both theoretical and practical fronts. We attack a few open problems in the field of Bayesian optimization and then turn our attention to improving the practical performance of Bayesian optimization and applying it to adapting (RM)HMC samplers. In this chapter, we summarize key results and contributions of this work (Section 7.1) and outline the direction of future research (Section 7.2).

7.1 Contributions and Results

Bayesian optimization without auxiliary optimization. In Chapter 2, we introduce an approach for doing Gaussian process based Bayesian optimization without having to optimize an auxiliary acquisition function. Traditional approaches of Bayesian optimization all require the optimization of an acquisition function which is often multi-modal and thus NP-hard to solve exactly. To optimize these auxiliary problems, practitioners often depended on heuristics which not only introduces practical difficulties but also theoretical ones. By combining Bayesian optimization with tree partitioning ideas [158], we derive the first Bayesian optimization algorithm that does not optimize an acquisition function. We show empirical results demonstrating the effectiveness of our approach. In addition, we also prove the convergence of our proposed algorithm.

Convergence results with varying hyper-parameters and noise. In this work, we bridge the gap between theory and practice in Bayesian optimization by introducing the first convergence result that allows both varying kernel hyper-parameters and noisy observations. In practice, it is often essential to tune the kernel hyper-parameters of the underlying Gaussian process for Bayesian optimization to be effective (e.g. [202]). Also, a core tenet of Bayesian optimization is to handle noisy observations. Prior to this work, however, there

did not exist any theoretical analysis of Bayesian optimization that can accommodate both varying hyper-parameters and noisy observations which is by far the most common setting in practice. In Chapter 3, we derive a cumulative regret bound in this setting. The theoretical results also provide us with guidelines on how to design hyper-parameter estimation methods. A simple simulation demonstrates the importance of following these guidelines.

High-dimensional Bayesian optimization. Despite the successes of Bayesian optimization, it is restricted to problems of moderate dimension, and several workshops on Bayesian optimization have identified its scaling to high-dimensions as one of the holy grails of the field. In Chapter 4, we introduce a novel random embedding idea to attack this problem. The resulting Random EMbedding Bayesian Optimization (REMBO) algorithm is very simple, has important invariance properties, and applies to domains with both categorical and continuous variables. We present a thorough theoretical analysis of REMBO. Empirical results confirm that REMBO can effectively solve problems with billions of dimensions, provided the intrinsic dimensionality is low. They also show that REMBO achieves state-of-the-art performance in optimizing the 47 discrete parameters of a popular mixed integer linear programming solver.

Portfolio of acquisition functions. The performance of a Bayesian optimization method depends on its exploration strategy, i.e. the choice of acquisition function, and it is not clear a priori which choice will result in superior performance. While portfolio methods [97] provide an effective, principled way of combining a collection of acquisition functions, they are often based on measures of past performance which can be misleading. To address this issue, we introduce (Chapter 5) the Entropy Search Portfolio (ESP): a novel approach to portfolio construction which is motivated by information theoretic considerations. We show that ESP outperforms existing portfolio methods on several real and synthetic problems, including geostatistical datasets and simulated control tasks. We not only show that ESP is able to offer performance as good as the best, but unknown, acquisition function, but surprisingly it often gives better performance. Finally, over a wide range of conditions we find that ESP is robust to the inclusion of poor acquisition functions.

Adaptive Hamiltonian Monte Carlo. In Chapter 6, we address the widely experienced difficulty in tuning Monte Carlo sampler based on simulating Hamiltonian dynamics. We develop an algorithm that allows for the adaptation of Hamiltonian and Riemann manifold Hamiltonian Monte Carlo samplers using Bayesian optimization that allows for infinite adaptation of the parameters of these samplers. We show that the resulting samplers are

ergodic, and that the use of our adaptive algorithms makes it easy to obtain more efficient samplers, in some cases precluding the need for more complex solutions. Hamiltonian-based Monte Carlo samplers are widely known to be an excellent choice of MCMC method, and we aim in this work to remove a key obstacle towards the more widespread use of these samplers in practice.

7.2 Future Work

In this work, we have introduced both practical and theoretical advances to Bayesian optimization. In this section, we envisage the potential ways of further improving Bayesian optimization and its applications as well as ways of adapting the essential components of Bayesian optimization to further the advancement of the field of Reinforcement Learning.

7.2.1 Improvement to Bayesian Optimization and its Applications

Extending BaMSOO to stochastic settings with varying hyper-parameters. In Chapter 2, we introduced BaMSOO which is capable of doing Bayesian optimization without auxiliary acquisition function optimization. BaMSOO is, however, limited to noiseless observations. Although, one can adapt the kernel hyper-parameters in practice, the current proof assumes fixed hyper-parameters. The difficulty lies in the fact that BaMSOO does not modify the nodes of the tree once their UCB and LCB values have been evaluated. By not re-evaluating the bounds, we save computation significantly. In face of noise and varying kernel hyper-parameters, however, these values may become stale. One potential direction is to allow the bounds of some nodes to be evaluated more than once. This is similar to the approach taken in [213]. In taking this approach, one has to be careful to control the total number of node evaluations so as to limit the computational expenses. Techniques and Lemmas introduced in Chapter 3 can be recycled to prove the convergence of the new algorithm.

Further advances in adaptive HMC. In Chapter 6, we introduced the AHMC algorithm which automatically adapts the number of leapfrog steps as well as the step size. There are two directions in which we can take our AHMC algorithm further. First, we would like to adapt the mass matrix of HMC. The mass matrix, which is the component of HMC that holds the most tunable parameters, is kept fixed in AHMC. As recent advances in optimization suggests (e.g. [63, 120]), adapting just the diagonal entries of the second order statistics by stochastic approximation in an optimizer can bring about tremendous gains.

The same idea, we believe, translates to sampling. In adapting HMC, however, one has to take extra care to adapt all three components simultaneously: the steps size, number of leapfrog steps, and (diagonal entries of) the mass matrix. In addition to better adaptation, we would also like to understand the convergence properties of adaptive HMC samplers better. Recently, a number of publications concerning the convergence properties of HMC appeared [26, 41, 136]. These results could solve the key problems in establishing the convergence of adaptive HMC.

Efficient multi-task Bayesian optimization. The ability to adapt our experience to new tasks is an important hallmark of an human. Bayesian optimization should be able to utilize data from a variety of tasks to better optimize the task of interest. Multi-task Bayesian optimization attacks precisely this problem. The previous Multi-task approach [208], however, relies on entropy search as the acquisition function. Then entropy search criterion, although conceptual simple, is really expensive to even approximately compute. Recent approximations like PES [92] dramatically reduce computation required, but imposes restrictions on the types of kernels one can use. Instead of using entropy search, we can rely on traditional acquisition functions like EI. To illustrate our idea, consider the following simple setup with two tasks: a main task of interest, and a cheap proxy of the main task. In the start of an iteration, we optimize EI while restricting ourselves to the main task. We call the optimizer $\mathbf{x}$. Instead of immediately evaluating $\mathbf{x}$ on the main task, we try to find a set of parameters, over either the main task or the proxy, that reduces the entropy of $f(\mathbf{x})$ the most per unit of expected computation time. That is we want to reduce the uncertainty of $f(\mathbf{x})$ using the least amount of computation. When the proxy task is cheap to evaluate, the aforementioned method would take advantage of it and greatly reduce the total computation time.

7.2.2 Statistical Modeling in Reinforcement Learning

Statistical models like Gaussian processes play a big role in Bayesian optimization since they not only capture the mean but also model uncertainties. It is the uncertainty estimates that allow us to trade off exploration and exploitation efficiently to achieve great sample efficiency in Bayesian optimization. Statistical models, however, are much more underexplored in the field of Reinforcement Learning (RL).

Although seemingly different, RL and Bayesian optimization share many similarities. Specifically, both methods try to solve sequential decision making problems. Whereas RL assumes many states in which the agent takes actions, Bayesian optimization assumes

only one. Without having to model the transition between states, statistical models used in Bayesian optimization focus on modeling the mean and uncertainty of the actions.

Although statistical modeling is much harder in RL as it demands the understanding of transitions between states, we can still expect to benefit from it. Indeed, the method that achieves the best data efficiency on the control problem of Cartpole is PILCO [56] which uses Gaussian processes to model the state transitions. Theoretical as well as practical results show, when the statistical models are powerful enough, Thompson sampling is one of the best algorithms for doing exploration in RL [165, 164]. Last but not least, recent efforts on modeling the densities of states resulted in great increase in performance of RL algorithms when it comes to hard exploration problems [18]. These results suggest, perhaps one of the best route in achieving better data efficiency and exploration in RL, involves building better statistical models.

One particular way of achieving better statistical modeling is through hierarchical abstractions. Imagine the task of traveling from Oxford to Edinburgh. One plan that can accomplish the task would be to take the train from Oxford to Paddington station and then the tube to London St Pancras station. Edinburgh would then be a simple train ride away from St Pancras station. Instead of working on this abstract level, a robot that uses Thompson sampling would try to model the uncertainties of rewards in relation to the torques to be put on its motors. Although a perfect statistical model could potentially enable the robot to solve the task, building such a statistical model would nonetheless require a tremendous amount of data. Assuming the robot can accomplish each individual part of the journey from London to Edinburgh, however, its statistical model would only require the knowledge of the public transportation network of the UK and the robot would be able to plan like a human. This story sound simple, but it does illustrate the point that our statistical models has to be applied at the right level of abstraction. Some recent work has successfully applied hierarchical temporal abstractions in the area of continuous control [89]. Much more awaits in this exciting direction of research.

Bibliography

[1] P. Adams, R and O. Stegle. Gaussian process product models for nonparametric nonstationarity. In *ICML*, pages 1–8, 2008.

[2] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *ICML*, 2013.

[3] E. B. Anderes and M. L. Stein. Estimating deformations of isotropic Gaussian random fields on the plane. *The Annals of Statistics*, 36(2):719–741, 2008.

[4] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43, 2003.

[5] C. Andrieu and C. Robert. Controlled MCMC for optimal sampling. Technical Report 0125, Cahiers de Mathematiques du Ceremade, Universite Paris-Dauphine, 2001.

[6] Christophe Andrieu, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[7] JA. M. Assael, Z. Wang, and N. de Freitas. Heteroscedastic treed Bayesian optimisation. *arXiv preprint arXiv:1410.7172*, 2014.

[8] Y. Atchadé and G. Fort. Limit theorems for some adaptive MCMC algorithms with subgeometric kernels. *Bernoulli*, 16(1):116–154, 2010.

[9] C. Audet, J. Jr, Dennis, D. W. Moore, A. Booker, and P. D. Frank. Surrogate-model-based method for constrained optimization. In *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2000.

[10] J. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *Conference on Learning Theory*, 2010.

[11] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.

[12] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Symposium on Foundations of Computer Science*, pages 322–331, 1995.

[13] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[14] J. Azimi, A. Jalali, and X.Z. Fern. Hybrid batch bayesian optimization. In *ICML*, 2012.

[15] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In *ICML*, 2013.

[16] R. Bardenet and B. Kégl. Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm. In *ICML*, pages 55–62, 2010.

[17] T Bartz-Beielstein, C Lasarczyk, and M Preuss. Sequential parameter optimization. In *Proc. CEC-05*, 2005.

[18] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[19] R. Benassi, J. Bect, and E. Vazquez. Robust Gaussian process-based global optimization using a fully Bayesian expected improvement criterion. In *Learning and Intelligent Optimization*, pages 176–190. Springer, 2011.

[20] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, pages 2546–2554, 2011.

[21] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[22] J. Bergstra, B. Komer, C. Eliasmith, and D. Warde-Farley. Preliminary evaluation of hyperopt algorithms on HPOLib. *International Conference on Machine Learning AutoML Workshop*, 2014.

[23] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search. *CoRR*, abs/1209.5111, 2012.

[24] M. Berkelaar, K. Eikland, and P. Notebaert. *lpsolve : Open source (Mixed-Integer) Linear Programming system*, 2016. http://lpsolve.sourceforge.net/.

[25] A. Beskos, N. S. Pillai, G. O. Roberts, J. M. Sanz-Serna, and A. M. Stuart. Optimal tuning of the hybrid Monte-Carlo algorithm. *Preprint arXiv:1001.4460*, 2010.

[26] M. J Betancourt, S. Byrne, S. Livingstone, and M. Girolami. The geometric foundations of hamiltonian monte carlo. *arXiv preprint arXiv:1410.5110*, 2014.

[27] S. Bochner. *Lectures on Fourier integrals*. Princeton University Press, 1959.

[28] E. V. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-task Gaussian process prediction. In *NIPS*, 2008.

[29] L. Bornn, G. Shaddick, and J. V. Zidek. Modeling nonstationary processes through dimension expansion. *Journal of the American Statistical Society*, 107(497), 2012.

[30] P. Boyle. *Gaussian Processes for Regression and Optimisation*. PhD thesis, Victoria University of Wellington, Wellington, New Zealand, 2007.

[31] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[32] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

[33] E. Brochu, T. Brochu, and N. de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *ACM SIGGRAPH / Eurographics SCA*, pages 103–112, 2010.

[34] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report UBC-2009-23 and arXiv:1012.2599v1, 2009.

[35] E. Brochu, N. de Freitas, and A. Ghosh. Active preference learning with discrete choice data. In *NIPS*, pages 409–416, 2007.

[36] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[37] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. 2009.

[38] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

[39] A. D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

[40] D. Busby. Hierarchical adaptive experimental design for Gaussian process emulators. *Reliability Engineering and System Safety*, 94(7):1183–1193, July 2009.

[41] S. Byrne. Geometry and ergodicity of hamiltonian monte carlo. 2015.

[42] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian processes for regression. *arXiv preprint arXiv:1402.5876*, 2014.

[43] A. Carpentier and R. Munos. Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In *AIStats*, pages 190–198, 2012.

[44] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, 2006.

[45] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[46] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *NIPS*, pages 2249–2257, 2011.

[47] B. Chen, R.M. Castro, and A. Krause. Joint optimization and variable selection of high-dimensional Gaussian processes. In *ICML*, 2012.

[48] L. Chen, Z. Qin, and J. S. Liu. Exploring Hybrid Monte Carlo in Bayesian Computation. *Sigma*, 2:2–5, 2001.

[49] O. F. Christensen, G. O. Roberts, and J. S. Rosenthal. Scaling limits for the transient phase of local Metropolis–Hastings algorithms. *Journal of the Royal Statistical Society: Series B*, 67(2):253–268, 2005.

[50] I. Clark and W. V. Harper. *Practical Geostatistics 2000: Case Studies*. Ecosse North America, 2008.

[51] S. Clark. *Parallel Machine Learning Algorithms In Bioinformatics And Global Optimization*. PhD thesis, Cornell University, 2012.

[52] E. Contal, V. Perchet, and N. Vayatis. Gaussian process optimization with mutual information. In *ICML*, 2013.

[53] P.A. Coquelin and R. Munos. Bandit algorithms for tree search. In *UAI*, pages 67–74, 2007.

[54] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, (7):81–227, 2011.

[55] N. de Freitas, A. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *ICML*, 2012.

[56] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[57] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. Learning where to attend with deep architectures for image tracking. *Neural Computation*, 24(8):2151–2184, 2012.

[58] T. Desautels, A. Krause, and J. Burdick. Parallelizing exploration-exploitation tradeoffs with Gaussian process bandit optimization. *Journal of Machine Learning Research*, 2014.

[59] J. Djolonga, A. Krause, and V. Cevher. High dimensional Gaussian process bandits. In *NIPS*, pages 1025–1033, 2013.

[60] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, July 2015.

[61] T. Domhan, T. Springenberg, and F. Hutter. Extrapolating learning curves of deep neural networks. In *International Conference on Machine Learning AutoML Workshop*, 2014.

[62] S. Duane, A. D. Kennedy, B. J. Pendleton, and D Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[63] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[64] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *NIPS Workshop on Bayesian Optimization in Theory and Practice*.

[65] Y.. Engel. Algorithms and representations for reinforcement learning. *Doktorarbeit, The Hebrew University of Jerusalem*, 2005.

[66] M. Feurer, T. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *AAAI*, 2015.

[67] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, pages 3212–3220, 2012.

[68] V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck. Multi-bandit best arm identification. In *Advances in Neural Information Processing Systems*, pages 2222–2230, 2011.

[69] J. R. Gardner, M. J. Kusner, Z. Xu, K. Q. Weinberger, and J. P. Cunningham. Bayesian optimization with inequality constraints. In *ICML*, pages 937–945, 2014.

[70] R. Garnett, M. A. Osborne, and P. Hennig. Active learning of linear embeddings for Gaussian processes. In *UAI*, 2014.

[71] R. Garnett, M. A. Osborne, S. Reece, A. Rogers, and S. J. Roberts. Sequential Bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 53(9):1430–1446, 2010.

[72] R. Garnett, M. A. Osborne, and S. J. Roberts. Bayesian optimization for sensor set selection. In *ACM/IEEE IPSN*, pages 209–219. ACM, 2010.

[73] M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. In *UAI*, 2014.

[74] S. Ghosal and A. Roy. Posterior consistency of Gaussian process prior for nonparametric binary regression. *The Annals of Statistics*, 34:2413–2429, 2006.

[75] D. Ginsbourger and R. Le Riche. Dealing with asynchronicity in parallel Gaussian process based global optimization. http://hal.archives-ouvertes.fr/hal-00507632, 2010.

[76] D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

[77] M. Girolami and B. Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 73(2):123–214, 2011.

[78] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

[79] C. P. Gomes, W.J. van Hoeve, and A. Sabharwal. Connections in networks: A hybrid approach. In *CPAIOR*, volume 5015, pages 303–307, 2008.

[80] P. Goovaerts. *Geostatistics for natural resources evaluation*. Oxford University Press, 1997.

[81] R. B. Gramacy, G. A. Gray, S. Le Digabel, H. K.H. Lee, P. Ranjan, G. Wells, and S. M. Wild. Modeling an augmented Lagrangian for improved blackbox constrained optimization. *arXiv preprint arXiv:1403.4890*, 2014.

[82] R. B. Gramacy, H. K. H. Lee, and W. G. Macready. Parameter space exploration with Gaussian process trees. In *ICML*, pages 45–52, 2004.

[83] R. B. Gramacy and Herbert K.H. Lee. Optimization under unknown constraints. *arXiv preprint arXiv:1004.4027*, 2010.

[84] S. Grunewalder, J.Y. Audibert, M. Opper, and J. Shawe-Taylor. Regret bounds for Gaussian process bandit problems. In *AIStats*, pages 273–280, 2010.

[85] F. Hamze, Z. Wang, and N. de Freitas. Self-avoiding random dynamics on integer complex systems. *ACM Transactions on Modelling and Computer Simulation*, 23(1):9:1–9:25, 2013.

[86] F. Hamze, Z. Wang, and N. de Freitas. Self-avoiding random dynamics on integer complex systems. *ACM Transactions on Modeling and Computer Simulation*, 23(1):9:1–9:25, 2013.

[87] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[88] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[89] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[90] P. Hennig and C.J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[91] J. M. Hernández-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani. Predictive entropy search for Bayesian optimization with unknown constraints. In *ICML*, 2015.

[92] J. M Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *NIPS*. 2014.

[93] D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. *Bayesian Statistics*, 6, 1998.

[94] G. E. Hinton and R. Salakhutdinov. Using deep belief nets to learn covariance kernels for Gaussian processes. In *NIPS*, pages 1249–1256, 2008.

[95] M. D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Preprint arXiv:1111.4246*, 2011.

[96] M. W. Hoffman, H. Kueck, N. de Freitas, and A. Doucet. New inference strategies for solving Markov decision processes using reversible jump MCMC. In *UAI*, pages 223–231, 2009.

[97] M.W. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for Bayesian optimization. In *UAI*, pages 327–336, 2011.

[98] M.W. Hoffman, B. Shahriari, and N. de Freitas. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *AIStats*, pages 365–374, 2014.

[99] H. H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, 2012.

[100] D. Hsu, S. M. Kakade, and T. Zhang. A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17(52):1–6, 2012.

[101] D. Huang, T.T. Allen, W.I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential Kriging meta-models. *J. of Global Optimization*, 34(3):441–466, 2006.

[102] F. Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, Vancouver, Canada, 2009.

[103] F. Hutter, H. Hoos, and K. Leyton-Brown. Identifying key algorithm parameters and instance features using forward selection. In *LION*, 2013. To appear; retrieved from www.cs.ubc.ca/~hutter/papers/2013LION_KeyParameters.pdf.

[104] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *ICML*, 2014.

[105] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In *CPAIOR*, pages 186–202, 2010.

[106] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, pages 507–523, 2011.

[107] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, pages 507–523, 2011.

[108] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Parallel algorithm configuration. In *LION*, pages 55–70, 2012.

[109] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[110] H. Ishwaran. Applications of hybrid Monte Carlo to Bayesian generalized linear models: Quasicomplete separation and neural networks. *Journal of Computational and Graphical Statistics*, 8(4):779–799, 1999.

[111] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[112] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21(4):345–383, 2001.

[113] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global optimization*, 13(4):455–492, 1998.

[114] E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pages 592–600, 2012.

[115] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, volume 7568 of *Lecture Notes in Computer Science*, pages 199–213. Springer Berlin Heidelberg, 2012.

[116] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez. Bayesian optimization with exponential convergence. In *Advances in Neural Information Processing Systems*, pages 2809–2817, 2015.

[117] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic Gaussian process regression. In *ICML*, pages 393–400, 2007.

[118] J. Kim, T. Ohta, Y. Tateisi, and J. ichi Tsujii. GENIA corpus - a semantically annotated corpus for bio-textmining. In *ISMB (Supplement of Bioinformatics)*, pages 180–182, 2003.

[119] S. Kim, N. Shephard, and S. Chib. Stochastic volatility: likelihood inference and comparison with ARCH models. *The Review of Economic Studies*, 65(3):361–393, 1998.

[120] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[121] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. 2006.

[122] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery*, 18(1):140–181, 2009.

[123] A. Krause and C. S. Ong. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

[124] D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of Chemical, Metallurgical, and Mining Society of South Africa*, 1951.

[125] H. Kueck, N. de Freitas, and A. Doucet. SMC samplers for Bayesian optimal nonlinear design. In *IEEE Nonlinear Statistical Signal Processing Workshop*, pages 99–102, 2006.

[126] H. Kueck, M. Hoffman, A. Doucet, and N. de Freitas. Inference and learning for active sensing, experimental design and control. In H. Araujo, A. Mendonca, A. Pinho, and M. Torres, editors, *Pattern Recognition and Image Analysis*, volume 5524, pages 1–10. Springer Berlin Heidelberg, 2009.

[127] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering*, 86(1):97–106, 1964.

[128] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[129] M. Lázaro-Gredilla and A. R. Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *Neural Networks, IEEE Transactions on*, 21(8):1345–1351, 2010.

[130] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[131] Q. V Le, A. J Smola, and S. Canu. Heteroscedastic Gaussian process regression. In *ICML*, pages 489–496, 2005.

[132] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *Principles and Practice of Constraint Programming*, pages 556–572. Springer, 2002.

[133] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *World Wide Web*, pages 661–670, 2010.

[134] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

[135] D. V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

[136] S. Livingstone, M. Betancourt, S. Byrne, and M. Girolami. On the geometric ergodicity of Hamiltonian Monte Carlo. *arXiv preprint arXiv:1601.08057*, 2016.

[137] D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Canada, 2008.

[138] D. Lizotte, R. Greiner, and D. Schuurmans. An experimental methodology for response surface optimization methods. *J. of Global Optimization*, pages 1–38, 2011.

[139] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with Gaussian process regression. In *IJCAI*, pages 944–949, 2007.

[140] M Locatelli. Bayesian algorithms for one-dimensional global optimization. *J. Global Optimization*, 1997.

[141] M. Lzaro-gredilla and M. K. Titsias. Variational heteroscedastic Gaussian process regression. In *ICML*, pages 841–848. ACM, 2011.

[142] O. Madani, D. Lizotte, and R. Greiner. Active model selection. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

[143] N. Mahendran, Z. Wang, F. Hamze, and N. de Freitas. Adaptive MCMC with Bayesian optimization. In *AIStats*, pages 751–760, 2012.

[144] R. Marchant and F. Ramos. Bayesian optimization for intelligent environmental monitoring. In *IROS*, pages 2242–2249, 2012.

[145] O. Maron and A. W Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. *Robotics Institute*, page 263, 1993.

[146] R. Martinez–Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103, 2009.

[147] J. Martinez, J. J. Little, and N. de Freitas. Bayesian optimization with an empirical hardness model for approximate nearest neighbour search. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[148] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A Castellanos. Active policy learning for robot planning and exploration under uncertainty. *RSS*, 2007.

[149] G Matheron. The theory of regionalized variables and its applications. *Cahier du Centre de Morphologie Mathematique, Ecoles des Mines*, 1971.

[150] B. C. May, N. Korda, A. Lee, and D. S. Leslie. Optimistic Bayesian sampling in contextual bandit problems. Technical Report 11:01, Statistics Group, School of Mathematics, University of Bristol, 2011.

[151] Sean P. Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, 1993.

[152] Y. Miao, Z. Wang, and P. Blunsom. Bayesian optimisation for machine translation. *arXiv preprint arXiv:1412.7180*, 2014.

[153] V. Mnih, C. Szepesvári, and J. Audibert. Empirical Bernstein stopping. In *ICML*, pages 672–679. ACM, 2008.

[154] J. Močkus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *J. of Global Optimization*, 4(4):347–365, 1994.

[155] J. Močkus, A. Močkus, and L. Močkus. *Bayesian approach for randomization of heuristic algorithms of discrete programming*. American Math. Society, 1999.

[156] J. Močkus, V. Tiesis, and A. Žilinskas. The application of bayesian methods for seeking the extremum. In L Dixon and G Szego, editors, *Toward Global Optimization*, volume 2. Elsevier, 1978.

[157] S. Mohamed, K. Heller, and Z. Ghahramani. Bayesian exponential family PCA. In *Advances in Neural Information Processing Systems*, pages 1089–1096. 2008.

[158] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *NIPS*, pages 783–791, 2011.

[159] R. Munos. From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning. Technical Report hal-00747575, INRIA Lille, 2014.

[160] R. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[161] R. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

[162] R. Neal and J. Zhang. High dimensional classification with Bayesian neural networks and Dirichlet diffusion trees. In *Feature Extraction*, pages 265–296. Springer, 2006.

[163] J.A. Nelder and R.W.M. Wedderburn. Generalized linear models. 135(3):370–384, 1972.

[164] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances In Neural Information Processing Systems*, pages 4026–4034, 2016.

[165] Ian Osband, Dan Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

[166] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimization. In *LION*, 2009.

[167] C. Paciorek and M. Schervish. Nonstationary covariance functions for gaussian process regression. In *NIPS*, volume 16, pages 273–280, 2004.

[168] A. Parameswaran, H. Garcia-Molina, and A. Rajaraman. Towards the web of concepts: Extracting concepts from large datasets. *Proceedings of the VLDB Endowment*, 3(1-2):566–577, 2010.

[169] C. Pasarica and A. Gelman. Adaptively scaling the Metropolis algorithm using expected squared jumped distance. *Statistica Sinica*, 20(1):343, 2010.

[170] V. Picheny and D. Ginsbourger. A nonstationary space-time Gaussian process model for partially converged simulations. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):57–78, 2013.

[171] J. C. Pinheiro and D. M. Bates. Unconstrained parametrizations for variance-covariance matrices. *Statistics and Computing*, 6(3):289–296, 1996.

[172] M. Plummer, N. Best, K. Cowles, and K. Vines. Coda: Convergence diagnosis and output analysis for mcmc. *R News*, 6(1):7–11, 2006.

[173] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[174] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

[175] M. Ranzato and G. Hinton. Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *CVPR*, pages 2551–2558, 2010.

[176] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[177] G. O Roberts, A. Gelman, and W. R Gilks. Weak convergence and optimal scaling of random walk metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.

[178] G. O Roberts and J. S Rosenthal. Optimal scaling for various metropolis-hastings algorithms. *Statistical Science*, 16(4):351–367, 2001.

[179] G. O. Roberts and J. S. Rosenthal. Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. *Journal of applied probability*, 44(2):458–475, 2007.

[180] G. O Roberts and J. S. Rosenthal. Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.

[181] G. O Roberts and R. L Tweedie. Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. *Biometrika*, 83(1):95–110, 1996.

[182] M. Rudelson and R. Vershynin. Non-asymptotic theory of random matrices: Extreme singular values. In *Int. Congress of Mathematicians*, pages 1576–1599, 2010.

[183] D. Russo and B. Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[184] Jasper S., K. Swersky, R. S. Zemel, and R. P. Adams. Input warping for Bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014.

[185] J Sacks, W J Welch, T J Welch, and H P Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[186] P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, 1992.

[187] A. Sankar, D.A. Spielman, and S.H. Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *Arxiv preprint cs/0310022*, 2003.

[188] T. J. Santner, B. Williams, and W. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

[189] M. J. Sasena. *Flexibility and Efficiency Enhancement for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[190] A. M. Schmidt and A. O'Hagan. Bayesian inference for nonstationary spatial covariance structures via spatial deformations. *Journal of the Royal Statistical Society, Series B*, 65:743–758, 2003.

[191] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 1997.

[192] M. Schonlau, W. J. Welch, and D. R. Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, 34:11–25, 1998.

[193] Steven L Scott. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[194] M. Seeger, Y. Teh, and M. I. Jordan. Semiparametric latent factor models. In *AIStats*, 2005.

[195] M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318, 2003.

[196] A. Shah, A. G. Wilson, and Z. Ghahramani. Student-t processes as alternatives to Gaussian processes. In *AIStats*, pages 877–885, 2014.

[197] B. Shahriari, K. Swersky, Z. Wang, R. P Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 2016.

[198] B. Shahriari, Z. Wang, M. W. Hoffman, A. Bouchard-Côté, and N. de Freitas. An entropy search portfolio. In *NIPS workshop on Bayesian Optimization*, 2014.

[199] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304, 2011.

[200] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2005.

[201] E. Snelson, C. E. Rasmussen, and Z. Ghahramani. Warped Gaussian processes. In *NIPS*, 2003.

[202] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, pages 2951–2959, 2012.

[203] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, Prabhat, and R. Adams. Scalable Bayesian optimization using deep neural networks. In *ICML*, 2015.

[204] Jasper Snoek. *Bayesian Optimization and Semiparametric Models with Applications to Assistive Technology*. PhD thesis, University of Toronto, Toronto, Canada, 2013.

[205] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, pages 1015–1022, 2010.

[206] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer, 2008.

[207] K. Swersky, D. Duvenaud, J. Snoek, F. Hutter, and M. A. Osborne. Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011*, 2014.

[208] K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *NIPS*, pages 2004–2012, 2013.

[209] K. Swersky, J. Snoek, and R. P. Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

[210] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[211] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *KDD*, pages 847–855, 2013.

[212] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

[213] M. Valko, A. Carpentier, and R. Munos. Stochastic simultaneous optimistic optimization. In *ICML*, 2013.

[214] M. Vallati, C. Fawcett, A. E. Gerevini, H. H. Hoos, and A. Saetti. Generating fast domain-optimized planners by automatically configuring a generic parameterised planner. In *ICAPS-PAL*, 2011.

[215] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-N recommender systems via Gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 225–232. ACM, 2014.

[216] E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *J. of Statistical Planning and Inference*, 140:3088–3095, 2010.

[217] E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, 140(11):3088–3095, 2010.

[218] M. Vihola. Grapham: Graphical models with adaptive random walk Metropolis algorithms. *Computational Statistics and Data Analysis*, 54(1):49 – 54, 2010.

[219] J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *J. of Global Optimization*, 44(4):509–534, 2009.

[220] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. Sample efficient actor-critic with experience replay. *ICLR*, 2017.

[221] Z. Wang and N. de Freitas. Predictive adaptation of hybrid Monte Carlo with Bayesian parametric bandits. In *NIPS Deep Learning and Unsupervised Feature Learning Workshop*, 2011.

[222] Z. Wang and N. de Freitas. Bayesian multiscale optimistic optimization. In *AIStats*, 2014.

[223] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 2016.

[224] Z. Wang, S. Mohamed, and N. de Freitas. Adaptive Hamiltonian and Riemann manifold Monte Carlo samplers. In *ICML*, 2013.

[225] Z. Wang, S. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, 2016.

[226] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. de Freitas. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, pages 1778–1784, 2013.

[227] E.R. Westervelt and J.W. Grizzle. *Feedback Control of Dynamic Bipedal Robot Locomotion*. Control and Automation Series. CRC PressINC, 2007.

[228] B. J. Williams, T. J. Santner, and W. I. Notz. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, 10:1133–1152, 2000.

[229] D. Yogatama and G. Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *AIStats*, pages 1077–1085, 2014.

[230] D. Yogatama and N. A. Smith. Bayesian optimization of text representations. *arXiv preprint arXiv:1503.00693*, 2015.

[231] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee. Improving object detection with deep convolutional networks via Bayesian optimization and structured prediction. In *CVPR*, 2015.

[232] T. Zhao, Z. Wang, A. Cumberworth, J. Gsponer, N. de Freitas, and A. Bouchard-Côté. Bayesian analysis of continuous time Markov chains with application to phylogenetic modelling. *Bayesian Analysis*, 2016.

[233] A. Žilinskas and J. Žilinskas. Global optimization based on a statistical model and simplical partitioning. *Computers and Mathematics with Applications*, 44:957–967, 2002.

Appendix A

Proofs for BaMSOO

We begin by introducing some notation. Let $\mathbf{x}_h^*$ denote the optimal node at level h. That is the cell of $\mathbf{x}_h^*$ contains the optimizer $\mathbf{x}^*$. Also let f^+ and $\mathbf{x}^+$ represent the best function value observed thus far and the associated node respectively.

A.1 Technical Lemmas

Lemma 2 (Lemma 5 of [55])**.** *Given a set of points $\mathbf{x}_{1:T} := \{\mathbf{x}_1, \dots, \mathbf{x}_T\} \in \mathcal{D}$ and a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ with kernel κ the following bounds hold:*

1. *Any $f \in \mathcal{H}$ is Lipschitz continuous with constant $\|f\|_{\mathcal{H}} L$, where $\|\cdot\|_{\mathcal{H}}$ is the Hilbert space norm and L satisfies the following:*

$$L^2 \leq \sup_{\mathbf{x} \in \mathcal{D}} \partial_{\mathbf{x}} \partial_{\mathbf{x}'} \kappa(x, x')|_{\mathbf{x}=\mathbf{x}'}$$

and for $\kappa(\mathbf{x}, \mathbf{x}') = \widetilde{\kappa}(\mathbf{x} - \mathbf{x}')$ we have

$$L^2 \leq \partial_{\mathbf{x}}^2 \widetilde{\kappa}(\mathbf{x})|_{x=0}.$$

2. *The projection operator $P_{1:T}$ on the subspace $\operatorname{span}_{t=1:T}\{\kappa(x_t, \cdot)\} \subseteq \mathcal{H}$ is given by*

$$P_{1:T} f := \mathbf{k}^\top(\cdot) \mathbf{K}^{-1} \langle \mathbf{k}(\cdot), f \rangle$$

where $\mathbf{k}(\cdot) = \mathbf{k}_{1:T}(\cdot) := [\kappa(\mathbf{x}_1, \cdot) \cdots \kappa(\mathbf{x}_T, \cdot)]^\top$ and $\mathbf{K} := [\kappa(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1:T}$; moreover, we have that

$$\langle \mathbf{k}(\cdot), f \rangle := \begin{bmatrix} \langle \kappa(\mathbf{x}_1, \cdot), f \rangle \\ \vdots \\ \langle \kappa(\mathbf{x}_T, \cdot), f \rangle \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_T) \end{bmatrix}.$$

Here $P_{1:T} P_{1:T} = P_{1:T}$ and $\|P_{1:T}\| \leq 1$ and $\|\mathbf{1} - P_{1:T}\| \leq 1$.

3. *Given tuples* $(\mathbf{x}_i, f_i)$ *with* $f_i = f(\mathbf{x}_i)$*, the minimum norm interpolation* $\bar{f}$ *with* $\bar{f}(\mathbf{x}_i) = f(\mathbf{x}_i)$ *is given by* $\bar{f} = P_{1:T} f$*. Consequently its residual* $g := (\mathbf{1} - P_{1:T})f$ *satisfies* $g(\mathbf{x}_i) = 0$ *for all* $\mathbf{x}_i \in \mathbf{x}_{1:T}$*.*

Lemma 3 (Lemma 6 of [55])**.** *Under the assumptions of Lemma 2 it follows that*

$$|f(\mathbf{x}) - P_{1:T} f(\mathbf{x})| \leq \|f\|_{\mathcal{H}} \sigma_T(\mathbf{x}),$$

where $\sigma_T^2(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x}) - \mathbf{k}_{1:T}^\top(\mathbf{x}) \mathbf{K}^{-1} \mathbf{k}_{1:T}(\mathbf{x})$ *and this bound is tight. Moreover,* $\sigma_T^2(\mathbf{x})$ *is the posterior predictive variance of a Gaussian process with the same kernel.*

Lemma 4 (Adapted from Proposition 1 of [55])**.** *Let* $\kappa : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ *be a kernel that is twice differentiable along the diagonal* $\{(\mathbf{x}, \mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^D\}$*, with* L *defined as in Lemma 2.1, and* f *be an element of the RKHS with kernel* κ*. If* f *is evaluated at point* $\mathbf{x}$*, then for any other point* $\mathbf{y}$ *we have* $\sigma_T(\mathbf{y}) \leq L\|\mathbf{x} - \mathbf{y}\|$*.*

Proof. Let $\mathcal{H}$ be the RKHS corresponding to κ and $f \in \mathcal{H}$ an arbitrary element with $g := (\mathbf{1} - P_{1:T})f$; the residual defined in lemma 2.3. Since $g \in \mathcal{H}$, we have by Lemma 2.1, g is Lipschitz. Thus we have that for any point $\mathbf{y}$:

$$|g(\mathbf{y})| \leq L\|g\|_{\mathcal{H}}\|\mathbf{y} - \mathbf{x}\| \leq L\|f\|_{\mathcal{H}}\|\mathbf{y} - \mathbf{x}\|, \tag{A.1}$$

where the second inequality is guaranteed by Lemma 2.2. On the other hand, by Lemma 3, we know that for all $\mathbf{y}$ we have the following tight bound:

$$|g(\mathbf{y})| \leq \|f\|_{\mathcal{H}} \sigma_T(\mathbf{y}) \tag{A.2}$$

Now, given the fact that both inequalities (A.1) and (A.2) are bounding the same quantity and that the latter is a tight estimate, we necessarily have that:

$$\|f\|_{\mathcal{H}} \sigma_T(\mathbf{y}) \leq L\|f\|_{\mathcal{H}}\|\mathbf{y} - \mathbf{x}\|.$$

Canceling $\|f\|_{\mathcal{H}}$ gives us the result. □

Lemma 5 (Adapted from Lemma 5.1 of [205])**.** *Let* f *be a sample from a GP. Consider* $\eta \in (0, 1)$ *and set* $B_T = 2\log(\pi_T/\eta)$ *where* $\sum_{i=1}^{\infty} \pi_T^{-1} = 1$*,* $\pi_T > 0$*. Then,*

$$|f(\mathbf{x}_T) - \mu_T(\mathbf{x}_T)| \leq B_T^{\frac{1}{2}} \sigma_T(\mathbf{x}_T) \; \forall T \geq 1$$

holds with probability at least $1 - \eta$*.*

Proof. For $\mathbf{x}_T$ we have that $f(\mathbf{x}) \sim \mathcal{N}(\mu_T(\mathbf{x}_T), \sigma_T(\mathbf{x}_T))$ since f is a sample from the GP. Now, if $r \sim \mathcal{N}(0,1)$, then

$$\begin{aligned}\mathbb{P}(r > c) &= e^{-c^2/2}(2\pi)^{-1/2}\int e^{-(r-c)^2/2-c(r-c)}dr \\ &< e^{-c^2/2}\mathbb{P}(r>0) = \frac{1}{2}e^{-c^2/2}.\end{aligned}$$

Thus we have that

$$\mathbb{P}\left(f(\mathbf{x}) - \mu_T(\mathbf{x}) > B_T^{1/2}\sigma_T(\mathbf{x})\right) = \mathbb{P}(r > B_T^{1/2}) < \frac{1}{2}e^{-B_T/2}.$$

By symmetry and the union bound, we have that $\mathbb{P}\left(|f(\mathbf{x}) - \mu_T(\mathbf{x})| > B_T^{1/2}\sigma_T(\mathbf{x})\right) < e^{-B_T/2}$. By applying the union bound again, we derive

$$\mathbb{P}\left(|f(\mathbf{x}) - \mu_T(\mathbf{x})| > B_T^{1/2}\sigma_T(\mathbf{x})\ \forall T \geq 1\right) < \sum_{T=1}^{\infty} e^{-B_T/2}.$$

By substituting $B_T = 2\log(\pi_T/\eta)$, we obtain the result. As in [205], we can set $\pi_T = \pi^2 T^2/6$. □

Since each node's UCB and LCB are only evaluated at most once, we give the following shorthands in notation. Let $N(\mathbf{x})$ be the number of evaluations of confidence bounds by the time the UCB of $\mathbf{x}$ is evaluated (line 12 of Algorithm 4) and let $T(\mathbf{x}) = |\mathcal{D}_t|$ be the time the UCB of $\mathbf{x}$ is evaluated. Define $\mathcal{U}(\mathbf{x}) = \mathcal{U}_{N(\mathbf{x})}(\mathbf{x}|\mathcal{D}_{T(\mathbf{x})}) = \mu(\mathbf{x}|\mathcal{D}_{T(\mathbf{x})}) + B_{N(\mathbf{x})}\sigma(\mathbf{x}|\mathcal{D}_{T(\mathbf{x})})$ and $\mathcal{L}(\mathbf{x}) = \mathcal{L}_{N(\mathbf{x})}(\mathbf{x}|\mathcal{D}_{T(\mathbf{x})}) = \mu(\mathbf{x}|\mathcal{D}_{T(\mathbf{x})}) - B_{N(\mathbf{x})}\sigma(\mathbf{x}|\mathcal{D}_{T(\mathbf{x})})$.

Lemma 6. *Consider $\mathcal{B}(\mathbf{x}^*, \rho)$ and $\gamma \in (0,1)$ as in Assumptions 2 and 3. Suppose $\mathcal{L}(\mathbf{x}_h^*) \leq f(\mathbf{x}_h^*) \leq \mathcal{U}(\mathbf{x}_h^*)$. If $\mathbf{x}_h^* \in \mathcal{B}(\mathbf{x}^*, \rho)$ and $\delta(h) < \epsilon_0$ then there exists a constant $\bar{c}$ such that $\mathcal{L}(\mathbf{x}_h^*) \geq f^* - \bar{c}B_{N(\mathbf{x}_h^*)}\gamma^{\frac{h}{2}}$.*

Proof. If $\mathbf{x}_h^*$ is not evaluated then $f(\mathbf{x}^+) \geq \mathcal{U}_T(\mathbf{x}_h^*) \geq f^* - \delta(h) \geq f^* - \epsilon_0$ which implies that $\mathbf{x}^+ \in \mathcal{B}(\mathbf{x}^*, \rho)$. Therefore, $f^* - c_2\|\mathbf{x}^+ - \mathbf{x}^*\|^2 \geq f(\mathbf{x}^+) \geq \mathcal{U}_T(\mathbf{x}_h^*) \geq f^* - \delta(h)$ which in turn implies that $\|\mathbf{x}^+ - \mathbf{x}^*\| \leq \sqrt{\frac{\delta(h)}{c_2}}$. Similarly $f^* - c_2\|\mathbf{x}_h^* - \mathbf{x}^*\|^2 \geq f(\mathbf{x}_h^*) \geq f^* - \delta(h)$. Therefore $\|\mathbf{x}_h^* - \mathbf{x}^*\| \leq \sqrt{\frac{\delta(h)}{c_2}}$. By the triangle inequality, we have

$$\|\mathbf{x}^+ - \mathbf{x}_h^*\| \leq \|\mathbf{x}^+ - \mathbf{x}^*\| + \|\mathbf{x}_h^* - \mathbf{x}^*\| \leq 2\sqrt{\frac{\delta(h)}{c_2}}.$$

By Lemma 4, we have that $\sigma_{T(\mathbf{x}_h^*)}(\mathbf{x}_h^*) \leq 2L\sqrt{\frac{\delta(h)}{c_2}}$. By the definition of $\mathcal{L}_T$, we can argue that

$$\begin{aligned}\mathcal{L}(\mathbf{x}_h^*) &\geq \mathcal{U}(\mathbf{x}_h^*) - 4B_{N(\mathbf{x}_h^*)}L\sqrt{\frac{\delta(h)}{c_2}} \\ &\geq f^* - \delta(h) - 4B_{N(\mathbf{x}_h^*)}L\sqrt{\frac{\delta(h)}{c_2}} \\ &= f^* - c\gamma^h - 4B_{N(\mathbf{x}_h^*)}L\sqrt{\frac{c\gamma^h}{c_2}}.\end{aligned}$$

Note that since $\gamma \in (0,1)$, $\gamma < \gamma^{1/2}$. Assume that $B_1 = b$. Let $\bar{c} = c/b + 4L\sqrt{\frac{c}{c_2}}$. Since $B_N > B_1 \; \forall N > 1$, we have the statement.

If $\mathbf{x}_h^*$ is evaluated then the statement is trivially true. □

Definition 8. *Let* $\bar{\gamma} := \gamma^{\frac{1}{2}}$, $\bar{\delta}_h := \bar{c}B_{N(\mathbf{x}_h^*)}\bar{\gamma}^h$, *and* $I_h^\epsilon = \{(h,i) : f(\mathbf{x}_{h,i}) + \epsilon \geq f^*\}$.

Lemma 7. *Assume that* $h_{\max} = n^\epsilon$. *For a node* $\mathbf{x}_{h,i}$ *at level* h, $B_{N(\mathbf{x}_{h,i})} = \mathcal{O}(\sqrt{h})$.

Proof. Assume that there are n_i nodes expanded at the end of iteration i of the outer loop (the while loop). In the $i+1^{th}$ iteration of the outer loop, there can be at most $h_{\max}(n_i)$ additional expansions added. Thus the total number of expansions at the end of iteration i is at most $n_{i-1} + h_{\max}(n_{i-1})$. We can prove by induction that $n_i \leq i^{\frac{1}{1-\epsilon}}$. Since any node at level h would be expanded after at most 2^h iterations, at the time of expansion of any node at level h, we have that $n < (2^h)^{\frac{1}{1-\epsilon}} = 2^{\frac{h}{1-\epsilon}}$ where n is the total number of expansions. Thus, there would be at most $2 \times 2^{\frac{h}{1-\epsilon}}$ evaluations. Hence,

$$B_{N(\mathbf{x}_{h,i})} \leq \sqrt{2\log(\pi^2 2^{\frac{2h}{1-\epsilon}+2}/6\eta)} \leq \sqrt{2\log(2^{\frac{2h}{1-\epsilon}+2}) + 2\log(\pi^2/6\eta)} = \mathcal{O}(\sqrt{h}).$$

□

Lemma 8. *After a finite number* (n_0) *of node expansions, an optimal node* $\mathbf{x}_{h_0}^* \in \mathcal{B}(\mathbf{x}^*, \rho)$ *is expanded such that* $\bar{c}B_{N(\mathbf{x}_{h_0}^*)}\bar{\gamma}_0^h \leq \epsilon_0$. *Also* $\forall h > h_0$, *we have that* $\bar{c}B_{N(\mathbf{x}_h^*)}\bar{\gamma}^h \leq \epsilon_0$ *and* $\mathbf{x}_h^* \in \mathcal{B}(\mathbf{x}^*, \rho)$.

Proof. Since it is clear that BaMSOO would expand every node after a finite number of node expansions, we only have to show that there exists an h_0 that satisfies the conditions. By Lemma 7, we have that $\forall h \; B_{N(\mathbf{x}_h^*)} = \mathcal{O}(\sqrt{h})$. Since $\bar{\gamma} < 1$, there exists an h_0 such that $\bar{c}B_{N(\mathbf{x}_h^*)}\bar{\gamma}^h \leq \epsilon_0 \; \forall h > h_0$. Since $f(\mathbf{x}_h^*) > f^* - \delta(h) > f^* - \bar{c}B_{N(\mathbf{x}_h^*)}\bar{\gamma}^h \geq f^* - \epsilon_0$, we have by Assumption 2 that, $\mathbf{x}_h^* \in \mathcal{B}(\mathbf{x}^*, \rho)$. □

Lemma 9. $\sum_{h=0}^{H} |I_h^{\bar{\delta}(H)}| \leq C \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{(D/4-D/\alpha)H}$ *for some constant* C *for all* $H > h_0$.

Proof. By Lemma 8, we know that $\bar{\delta}(H) = \bar{c} B_{N(\mathbf{x}_H^*)} \bar{\gamma}^H < \epsilon_0$ if $H > h_0$. Therefore, by Assumption 2, we have that $\chi_{\bar{\delta}(H)} = \{\mathbf{x} \in \chi : f(\mathbf{x}) \geq f^* - \bar{\delta}(H)\} \subseteq \mathcal{B}(x^*, \rho)$. Again by Assumption 2, we have that

$$f^* - \bar{\delta}(H) \leq f(\mathbf{x}) \leq f^* - c_2 \|\mathbf{x} - \mathbf{x}^*\|_2^2 \; \forall \mathbf{x} \in \chi_{\bar{\delta}(H)}.$$

Thus $\chi_{\bar{\delta}(H)} \subseteq \mathcal{B}\left(x^*, \sqrt{\frac{\bar{\delta}(H)}{c_2}}\right) = \mathcal{B}\left(x^*, \sqrt{\frac{\bar{c} B_{N(\mathbf{x}_H^*)} \gamma^{H/2}}{c_2}}\right)$.

Since each cell (h, i) contains a ℓ-ball of radius $\nu\delta(h)$ centered at $\mathbf{x}_{h,i}$ we have that each cell contains a ball $\mathcal{B}(\mathbf{x}_{h,i}, (\nu\delta(h))^{1/\alpha}) = \mathcal{B}(\mathbf{x}_{h,i}, (\frac{\nu c}{c_1})^{1/\alpha} \gamma^{h/\alpha})$. By the argument of volume, we have that $|I_h^{\bar{\delta}(H)}| \leq C_1 \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{HD/4 - hD/\alpha}$ for some constant C_1. Finally,

$$\begin{aligned}
\sum_{h=0}^{H} |I_h^{\bar{\delta}(H)}| &\leq C_1 \sum_{h=0}^{H} \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{HD/4 - hD/\alpha} \\
&= C_1 \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{HD/4} \sum_{h=0}^{H} \gamma^{-hD/\alpha} \\
&= C_1 \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{HD/4} \sum_{h=0}^{H} \left(\gamma^{D/\alpha}\right)^{h-H} \\
&\leq C_1 \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{HD/4} \sum_{h=0}^{\infty} \left(\gamma^{D/\alpha}\right)^{h-H} \\
&= C_1 \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{HD/4} \frac{\gamma^{-DH/\alpha}}{1 - \gamma^{D/\alpha}} \\
&= \frac{C_1}{1 - \gamma^{D/\alpha}} \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{HD/4 - DH/\alpha} \\
&= \frac{C_1}{1 - \gamma^{D/\alpha}} \left(B_{N(\mathbf{x}_H^*)}\right)^{D/2} \gamma^{(D/4 - D/\alpha)H}.
\end{aligned}$$

Setting $C = \frac{C_1}{1-\gamma^{D/\alpha}}$ gives us the desired result. □

Lemma 10. *Suppose* $\mathcal{L}(\mathbf{x}_h^*) \leq f(\mathbf{x}_h^*) \leq \mathcal{U}(\mathbf{x}_h^*)$. *If* $\mathbf{x}_h^*$ *is not evaluated (that is* $\mathcal{U}(\mathbf{x}_h^*) < f^+$*) then* f^+ *is* $\delta(h)$*-optimal.*

Proof. $f^+ > \mathcal{U}(\mathbf{x}_h^*) \geq f(\mathbf{x}_h^*) > f^* - \delta(h)$. □

A.2 Main Results

A.2.1 Simple Regret

Let h_n^* be the deepest level of an expanded optimal node with n node expansions. This following lemma is adapted from Lemma 2 of [158].

Lemma 11. *Suppose $\mathcal{L}(\mathbf{x}) \leq f(\mathbf{x}) \leq \mathcal{U}(\mathbf{x})$ for all $\mathbf{x}$ whose confidence region are evaluated. Whenever $h \leq h_{\max}(n)$ and $n \geq Ch_{\max}(n)\sum_{i=h_0}^{h}\left(B_{N(\mathbf{x}_i^*)}\right)^{D/2}\gamma^{(D/4-D/\alpha)i} + n_0$ for some constant C, we have $h_n^* \geq h$.*

Proof. We prove the statement by induction. By Lemma 8, we have that after n_0 node expansions, a node $\mathbf{x}_{h_0}^* \in \mathcal{B}(\mathbf{x}^*, \rho)$ is expanded. Also $\forall h > h_0$, we have that $\bar{c}B_{N(\mathbf{x}_h^*)}\bar{\gamma}^h \leq \epsilon_0$ and $\mathbf{x}_h^* \in \mathcal{B}(\mathbf{x}^*, \rho)$. For $h = h_0$, the statement is trivially satisfied. Thus assume that the statement is true for h. Let n be such that $n \geq Ch_{\max}(n)\sum_{i=h_0}^{h+1}\left(B_{N(\mathbf{x}_i^*)}\right)^{D/2}\gamma^{(D/4-D/\alpha)i} + n_0$. By the inductive hypothesis we have that $h_n^* \geq h$. As long as the optimal node at level $h+1$ is not expanded, all nodes expanded at the level are $\bar{\delta}(h+1)$-optimal by Lemma 6. By Lemma 9, we know that after $Ch_{\max}(n)\left(B_{N(\mathbf{x}_{h+1}^*)}\right)^{D/2}\gamma^{(D/4-D/\alpha)(h+1)}$ node expansions, the optimal node at level $h+1$ will be expanded since there are at most $\sum_{i=0}^{h+1}\left|I_i^{\bar{\delta}(h+1)}\right|$ $\bar{\delta}(h+1)$-optimal nodes at or beneath level $h+1$. Thus $h_n^* \geq h+1$. □

Theorem 8. *Suppose $\mathcal{L}(\mathbf{x}) \leq f(\mathbf{x}) \leq \mathcal{U}(\mathbf{x})$ for all $\mathbf{x}$ whose confidence region is evaluated. Let us write $h(n)$ to be the smallest integer $h \geq h_0$ such that*

$$Ch_{\max}(n)\sum_{i=h_0}^{h}\left(B_{N(\mathbf{x}_i^*)}\right)^{D/2}\gamma^{(D/4-D/\alpha)i} + n_0 \geq n.$$

Then the loss is bounded as

$$r_n \leq \delta(\min\{h(n), h_{\max}(n)+1\})$$

and $h_n^* \geq \min\{h(n)-1, h_{\max}(n)\}$.

Proof. From Lemma 9, and the definition of $h(n)$ we have that

$$Ch_{\max}(n)\sum_{i=h_0}^{h(n)-1}\left(B_{N(\mathbf{x}_i^*)}\right)^{D/2}\gamma^{(D/4-D/\alpha)i} + n_0 < n.$$

By Lemma 11, we have that $h_n^* \geq h(n)-1$ if $h(n)-1 \leq h_{\max}(n)$ and $h_n^* \geq h_{\max}(n)$ otherwise. Therefore $h_n^* \geq \min\{h(n)-1, h_{\max}(n)\}$.

By Lemma 10, we know that if $\mathbf{x}_{h_n^*+1}^*$ is not evaluated then f^+ is $\delta(h_n^*+1)$-optimal. If $\mathbf{x}_{h_n^*+1}^*$ is evaluated, then $f\left(\mathbf{x}_{h_n^*+1}^*\right)$ is $\delta(h_n^*+1)$-optimal. Thus $r_n \leq \delta(\min\{h(n), h_{\max}(n)+1\})$. □

Proof of Corollary 1. Suppose $\mathcal{L}(\mathbf{x}) \leq f(\mathbf{x}) \leq \mathcal{U}(\mathbf{x})$ for all $\mathbf{x}$ whose confidence region is evaluated. By Lemma 5, we know that this holds with probability at least $1 - \eta$.

By the definition of $h(n)$ we have that

$$\begin{aligned} n &\leq Ch_{\max}(n) \sum_{i=h_0}^{h(n)} \left(B_{N\left(\mathbf{x}_i^*\right)}\right)^{D/2} \gamma^{(D/4-D/\alpha)i} + n_0 \\ &\leq Ch_{\max}(n) \left(B_{N\left(\mathbf{x}_{h(n)}^*\right)}\right)^{D/2} \sum_{i=h_0}^{h(n)} \gamma^{-di} + n_0 \\ &\leq Ch_{\max}(n) \left(B_{N\left(\mathbf{x}_{h(n)}^*\right)}\right)^{D/2} \gamma^{-dh_0} \frac{\gamma^{-dh(n)} - 1}{\gamma^{-d} - 1} + n_0 \end{aligned} \tag{A.3}$$

If $h(n) \leq h_{\max}(n) + 1$, then by Theorem 8, we have that $h_n^* \geq h(n) - 1$. After n expansions, the optimal node $\mathbf{x}_{h(n)-1}^*$ has been expanded which suggests that its children's confidence bounds have been evaluated. Hence, $N\left(\mathbf{x}_{h(n)}^*\right) < 2n$ since there have only been n expansions. Therefore,

$$(A.3) \leq Kn^{\epsilon} \left(B_{2n}\right)^{D/2} \gamma^{-dh(n)}$$

for some constant K which implies that

$$\gamma^{h(n)} \leq K^{1/d} B_{2n}^{\frac{2\alpha}{4-\alpha}} n^{-\frac{1-\epsilon}{d}} = K^{1/d} \left[2\log(4\pi^2 n^2/6\eta)\right]^{\frac{\alpha}{4-\alpha}} n^{-\frac{1-\epsilon}{d}}.$$

By Theorem 8, we have that

$$r_n \leq c \min\left\{K^{1/d} \left[2\log(4\pi^2 n^2/6\eta)\right]^{\frac{\alpha}{4-\alpha}} n^{-\frac{1-\epsilon}{d}}, \gamma^{(n+1)^{\epsilon}}\right\} = \mathcal{O}\left(n^{-\frac{1-\epsilon}{d}} \log^{\frac{\alpha}{4-\alpha}}(n^2/\eta)\right).$$

□

Appendix B

Proofs for Hyper Parameter Optimization

B.1 Concentration

Lemma 12. *If ϵ_T is σ-sub-Gaussian, then $\mathbb{P}(|\epsilon_T| \geq a) \leq 2\exp\left(-\frac{a^2}{2\sigma^2}\right)$ $\forall a > 0$.*

Proof. By Markov's inequality, we can see that $\forall \rho > 0$

$$\mathbb{P}\left(\epsilon_T \geq a\right) = \mathbb{P}\left[\exp(\rho\epsilon_T) \geq \exp(\rho a)\right] \leq \frac{\mathbb{E}\left[\exp(\rho\epsilon_T)\right]}{\exp(\rho a)} \leq \exp\left(\frac{\sigma^2\rho^2}{2} - \rho a\right).$$

By taking $\rho = \frac{a}{\sigma^2}$, we have that $\mathbb{P}(\epsilon_T \geq a) \leq \exp\left(-\frac{a^2}{2\sigma^2}\right)$. By symmetry, we have that $\mathbb{P}(|\epsilon_T| \geq a) \leq 2\exp\left(-\frac{a^2}{2\sigma^2}\right)$. □

Lemma 13. *Let ϵ_t be independently σ-sub-Gaussian with $t \in \{1, \cdots, T\}$. Then, $\sum_{t=1}^{T} \lambda_t \epsilon_t$ is $(\|\boldsymbol{\lambda}\|\sigma)$-sub-Gaussian.*

Proof. For all $\rho \in \mathbb{R}$, we have

$$\begin{aligned}\mathbb{E}\left[\exp\left(\rho\sum_{t=1}^{T}\lambda_t\epsilon_t\right)\right] &= \mathbb{E}\left[\prod_{t=1}^{T}\exp\left(\rho\lambda_t\epsilon_t\right)\right] = \prod_{t=1}^{T}\mathbb{E}\left[\exp\left(\rho\lambda_t\epsilon_t\right)\right] \\ &\leq \prod_{t=1}^{T}\exp\left(\frac{\rho^2\lambda_t^2\sigma^2}{2}\right) = \exp\left(\frac{\rho^2\sigma^2\sum_{t=1}^{T}\lambda_t^2}{2}\right).\end{aligned}$$

□

To shorten the notation, in the remainder of this section, we will use $\mathbf{f}_T$ to denote the vector $(f_1, \ldots, f_T) = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_T))$ and, similarly, we use $\boldsymbol{\epsilon}_T$ in place of $\boldsymbol{\epsilon}_{1:T}$ and $\mathbf{y}_T$ in place of $\mathbf{y}_{1:T}$.

Lemma 14. $\mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T$ *is* $\left(\|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}\right)$*-sub-Gaussian.*

Proof. Using Lemma 13, with $\boldsymbol{\lambda} = \mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}$, we notice that we only need to bound $\boldsymbol{\lambda}^T\boldsymbol{\lambda}$. Proceeding,

$$\begin{aligned}
\boldsymbol{\lambda}^T\boldsymbol{\lambda} &= \operatorname{tr}(\mathbf{f}_T^T(K_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-2}\mathbf{f}_T) && \text{(B.1)}\\
&\leq \frac{1}{\sigma^2}\mathbf{f}_T^T(K_T^{\boldsymbol{\theta}})^{-1}\mathbf{f}_T && \text{(B.2)}\\
&\leq \frac{1}{\sigma^2}\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})} && \text{(B.3)}
\end{aligned}$$

The first inequality follows by choosing a constant $C_1 = \frac{1}{\sigma}$ so that the quadratic term $(K_T^{\boldsymbol{\theta}})^2_{i,j}$ upper-bounds the linear term $C_1^2(K_T^{\boldsymbol{\theta}})_{i,j}$. The last inequality holds because of the fact that $\mathbf{f}_T^T(K_T^{\boldsymbol{\theta}})^{-1}\mathbf{f}_T$ is the minimum value for the optimization problem:

$$\begin{aligned}
\min_{g\in\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})} \quad & \|g\|^2_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}\\
\text{s.t.} \quad & g(\mathbf{x}_t) = f_t \text{ for } t = 1,\ldots,T
\end{aligned}$$

for which f satisfies the constraint. That is, the function $g(\cdot)$ that agrees with $f(\mathbf{x}_t)$ has minimum norm $\mathbf{f}_T^T(K_T^{\boldsymbol{\theta}})^{-1}\mathbf{f}_T$. Hence, any other function $f(\cdot)$ that agrees with $f(\mathbf{x}_t)$ must have equal or larger norm. □

Lemma 15. $\mathbb{P}\left(\sigma^{-2}\|\boldsymbol{\epsilon}_T\|^2 - \boldsymbol{\epsilon}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T > 2\gamma_T^{\boldsymbol{\theta}} + 2\sqrt{2\gamma_T^{\boldsymbol{\theta}}\eta} + 2\sigma\eta\right) \leq e^{-\eta}$ *for any* $\eta > 0$.

Proof. First, by rearrangement we have that:

$$\sigma^{-2}\|\boldsymbol{\epsilon}_T\|^2 - \boldsymbol{\epsilon}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T = \boldsymbol{\epsilon}_T^T\left[\sigma^{-2}\mathbf{I} - (\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\right]\boldsymbol{\epsilon}_T = \quad \boldsymbol{\epsilon}_T^T\mathbf{Q}^T\boldsymbol{\Sigma}\mathbf{Q}\boldsymbol{\epsilon}_T$$

In the above equation $\mathbf{Q}^T\boldsymbol{\Sigma}\mathbf{Q}$ is the eigenvalue decomposition of the matrix

$$\boldsymbol{\Lambda} := \left[\sigma^{-2}\mathbf{I} - (\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\right]$$

where $\mathbf{Q}$ is an orthonormal matrix and $\boldsymbol{\Sigma}$ is a diagonal matrix.

The diagonal entries of Λ are such that $\Sigma_{i,i} = \frac{\Delta_i}{\sigma^2(\Delta_i+\sigma^2)}$ where Δ_i is the i^{th} eigenvalue of $\mathbf{K}_T^{\boldsymbol{\theta}}$. We know that $\operatorname{tr}(\boldsymbol{\Lambda}) = \operatorname{tr}(\boldsymbol{\Sigma}) = \sum_{i=1}^T \frac{\Delta_i}{\sigma^2(\Delta_i+\sigma^2)}$ and $\operatorname{tr}(\boldsymbol{\Lambda}^2) = \operatorname{tr}(\boldsymbol{\Sigma}^2) = \sum_{i=1}^T \left(\frac{\Delta_i}{\sigma^2(\Delta_i+\sigma^2)}\right)^2$. It is easy to see that $\operatorname{tr}(\boldsymbol{\Lambda}) \leq \sigma^{-2}\sum_{i=1}^T \log(1+\sigma^{-2}\Delta_i)$ since $\frac{\Delta_i}{\sigma^2(\Delta_i+\sigma^2)} \leq \sigma^{-2}\log(1+\sigma^{-2}\Delta_i)$ for all $1 \leq i \leq T$. Also $\operatorname{tr}(\boldsymbol{\Lambda}^2) \leq \sigma^{-4}\sum_{i=1}^T \log(1+\sigma^{-2}\Delta_i)$ since $\frac{\Delta_i}{\sigma^2(\Delta_i+\sigma^2)} < \sigma^{-2}$ for all $1 \leq i \leq T$. Finally, $\|\boldsymbol{\Lambda}\|_2 = \max_{1\leq i\leq T}\sqrt{\frac{\Delta_i}{\sigma^2(\Delta_i+\sigma^2)}} \leq \sigma^{-1}$ again because of the fact that $\frac{\Delta_i}{\sigma^2(\Delta_i+\sigma^2)} \leq \sigma^{-2}$.

Using the definition of maximum information gain γ_T^θ for Gaussians, we have the following three facts:

$$\begin{aligned} \operatorname{tr}(\mathbf{\Lambda}) &\leq 2\sigma^{-2}\gamma_T^{\boldsymbol{\theta}} && \text{(B.4)} \\ \operatorname{tr}(\mathbf{\Lambda}^2) &\leq 2\sigma^{-4}\gamma_T^{\boldsymbol{\theta}} && \text{(B.5)} \\ \|\mathbf{\Lambda}\|_2 &\leq \sigma^{-1}. && \text{(B.6)} \end{aligned}$$

By Theorem 2.1 of [100], we have that

$$\begin{aligned} & \mathbb{P}\left(\boldsymbol{\epsilon}_T^T\mathbf{\Lambda}\boldsymbol{\epsilon}_T > 2\gamma_T^{\boldsymbol{\theta}} + 2\sqrt{2\gamma_T^{\boldsymbol{\theta}}\eta} + 2\sigma\eta\right) \\ = \; & \mathbb{P}\left(\boldsymbol{\epsilon}_T^T\mathbf{\Lambda}\boldsymbol{\epsilon}_T > \sigma^2(2\sigma^{-2}\gamma_T^{\boldsymbol{\theta}} + 2\sqrt{2\sigma^{-4}\gamma_T^{\boldsymbol{\theta}}\eta} + 2\sigma^{-1}\eta)\right) \\ \leq \; & \mathbb{P}\left(\boldsymbol{\epsilon}_T^T\mathbf{\Lambda}\boldsymbol{\epsilon}_T > \operatorname{tr}(\mathbf{\Lambda}) + 2\sqrt{\operatorname{tr}(\mathbf{\Lambda}^2)\eta} + 2\|\mathbf{\Lambda}\|\eta\right) && \text{(B.7)} \\ \leq \; & e^{-\eta} && \text{(B.8)} \end{aligned}$$

which concludes the proof. □

Proposition 2. *Let* $(\varphi_T^{\boldsymbol{\theta}})^2 = \|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}^2 + \sqrt{8\gamma_{T-1}^{\boldsymbol{\theta}}\log(\frac{T^2\pi^2}{3\delta})} + \sqrt{2\log(\frac{2T^2\pi^2}{3\delta})}\|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})} + 2\gamma_{T-1}^{\boldsymbol{\theta}} + 2\sigma\log(\frac{T^2\pi^2}{3\delta})$. *Then* $\mathbb{P}\left(\|\mu_T(\cdot;\boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}_T^{\boldsymbol{\theta}}} \leq \varphi_{T+1}^{\boldsymbol{\theta}}\right) \geq 1 - \frac{6\delta}{\pi^2(T+1)^2}$.

Proof. Let $\|f\|_{K_T^{\boldsymbol{\theta}}}$ denote the RKHS norm of $f(\cdot)$ associated with the posterior covariance $\mathcal{K}_T^{\boldsymbol{\theta}}$ of the GP (equation (3.4)). From Lemma 7.2 of [205], we have

$$\|\mu_t(\cdot;\boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}_T^{\boldsymbol{\theta}}}^2 = \|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}^2 - \mathbf{y}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}_T + \sigma^{-2}\|\boldsymbol{\epsilon}_T\|^2. \quad \text{(B.9)}$$

This expression, with $\mathbf{y}_T = \mathbf{f}_T + \boldsymbol{\epsilon}_T$, can be easily bounded

$$\begin{aligned} \|\mu_T(\cdot;\boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}_T^{\boldsymbol{\theta}}}^2 &= \|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}^2 - \mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\mathbf{f}_T - 2\mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T \\ &\quad -\boldsymbol{\epsilon}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T + \sigma^{-2}\|\boldsymbol{\epsilon}_T\|^2 \\ &\leq \|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}^2 - 2\mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T \\ &\quad -\boldsymbol{\epsilon}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T + \sigma^{-2}\|\boldsymbol{\epsilon}_T\|^2. \end{aligned}$$

Next, we prove that $2\mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T$ and $\sigma^{-2}\|\boldsymbol{\epsilon}_T\|^2 - \boldsymbol{\epsilon}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T$ are bounded with high probability.

By Lemma 14 we know that $\mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T$ is $\left(\|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}\right)$-sub-Gaussian. Hence, we can apply the concentration result of Lemma 12 to this variable, as follows:

$$\begin{aligned}
&\mathbb{P}\left[|2\mathbf{f}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T| \geq \sqrt{2\log(\frac{4(T+1)^2\pi^2}{6\delta})}\|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}\right] \\
\leq\ & 2\exp\left(-\frac{2\log(\frac{4(T+1)^2\pi^2}{6\delta})\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}}{2\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}}\right) \\
=\ & \frac{6\delta}{2\pi^2(T+1)^2}. \qquad \text{(B.10)}
\end{aligned}$$

By Lemma 15, with the choice $\eta = \log(\frac{2(T+1)^2\pi^2}{6\delta})$, we obtain

$$\mathbb{P}\left(\frac{\|\boldsymbol{\epsilon}_T\|^2}{\sigma^2} - \boldsymbol{\epsilon}_T^T(\mathbf{K}_T^{\boldsymbol{\theta}} + \sigma^2\mathbf{I})^{-1}\boldsymbol{\epsilon}_T > 2\gamma_T^{\boldsymbol{\theta}} + \sqrt{8\gamma_T^{\boldsymbol{\theta}}\eta} + 2\sigma\eta\right) \leq \frac{6\delta}{2\pi^2(T+1)^2}.$$

Finally, we can use a union bound to combine these two results, yielding

$$\mathbb{P}\left[\|\mu_T(\cdot;\boldsymbol{\theta}) - f(\cdot)\|_{\mathcal{K}_T^{\boldsymbol{\theta}}} \geq \varphi_T^{\boldsymbol{\theta}}\right] \leq \frac{6\delta}{\pi^2(T+1)^2}.$$

□

B.2 Supporting lemmas

Lemma 16 (Lemma 5.3 of [205])**.** *The information gain for the points selected can be expressed in terms of the predictive variances. That is*

$$\mathcal{I}_{\boldsymbol{\theta}}(\mathbf{y}_T;\mathbf{f}_T) = \frac{1}{2}\sum_{t=1}^{T}\log\left(1+\sigma^{-2}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta})\right)$$

Lemma 17. *If $\boldsymbol{\theta}' \leq \boldsymbol{\theta}$, then $\gamma_T^{\boldsymbol{\theta}} \leq \gamma_T^{\boldsymbol{\theta}'}$.*

Proof. By definition there exist a set $\mathcal{A}$ such that $\gamma_T^{\boldsymbol{\theta}} = \mathcal{I}_{\boldsymbol{\theta}}(\mathbf{y}_{\mathcal{A}};\mathbf{f}_{\mathcal{A}})$. Hence, using Lemma 16,

$$\begin{aligned}
\gamma_T^{\boldsymbol{\theta}} &= \mathcal{I}_{\boldsymbol{\theta}}(\mathbf{y}_{\mathcal{A}};\mathbf{f}_{\mathcal{A}}) \\
&= \frac{1}{2}\sum_{\mathbf{x}_t\in\mathcal{A}}\log\left(1+\sigma^{-2}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta})\right) \\
&\leq \frac{1}{2}\sum_{\mathbf{x}_t\in A}\log\left(1+\sigma^{-2}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}')\right) \\
&\leq \max_{\mathcal{B}\subset\mathcal{X}:|\mathcal{B}|=T}\left[\frac{1}{2}\sum_{\mathbf{x}_t\in\mathcal{B}}\log\left(1+\sigma^{-2}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}')\right)\right] \\
&= \gamma_T^{\boldsymbol{\theta}'}.
\end{aligned}$$

□

Lemma 18 (Based on Lemma 5.4 of [205]). $\sum_{t=1}^{T} \sigma_{t-1}^2(\mathbf{x};\boldsymbol{\theta}) \leq \frac{2}{\log(1+\sigma^2)}\gamma_T^{\boldsymbol{\theta}}$.

Proof. Since $s^2 \leq \frac{1}{\sigma^2 \log(1+\sigma^2)} \log(1+s^2)$, we have that by Lemma 16

$$\begin{aligned}\sum_{t=1}^{T} \sigma_{t-1}^2(\mathbf{x};\boldsymbol{\theta}) &\leq \sum_{t=1}^{T} \frac{\sigma^2}{\sigma^2 \log(1+\sigma^2)} \log(1+\sigma^{-2}\sigma_{t-1}^2(\mathbf{x};\boldsymbol{\theta})) \\ &= \frac{2}{\log(1+\sigma^2)} \mathcal{I}_{\boldsymbol{\theta}}(\mathbf{y}_T;\mathbf{f}_T) \\ &\leq \frac{2}{\log(1+\sigma^2)} \gamma_T^{\boldsymbol{\theta}}. \end{aligned} \tag{B.11}$$

□

Lemma 19 (Lemma 4 of [39]). *If $f \in \mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})$, then $f \in \mathcal{H}_{\boldsymbol{\theta}'}(\mathcal{X})$ for all $0 < \boldsymbol{\theta}' \leq \boldsymbol{\theta}$ and*

$$\|f\|_{\mathcal{H}_{\boldsymbol{\theta}'}(\mathcal{X})}^2 \leq \left(\prod_{i=1}^{d} \frac{\theta_i}{\theta_i'}\right) \|f\|_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})}^2.$$

B.3 Properties of the expected improvement acquisition function

Lemma 20 (Based on Lemma 8 of [39]). *Let $\varphi_t^{\boldsymbol{\theta}}$ be as defined in Proposition 2 and $\nu > 0$. Assume that $|\mu_{t-1}(\mathbf{x},\boldsymbol{\theta}) - f(\mathbf{x})| \leq \varphi_t^{\boldsymbol{\theta}}\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})$. For $\mathbf{x} \in \mathcal{X}$, $t \in \mathbb{N}$, set $\mu_{\boldsymbol{\theta}}^+ = \max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x},\boldsymbol{\theta})$, and $I_t^{\boldsymbol{\theta}}(\mathbf{x}) = \max\{0, f(\mathbf{x}) - \mu_{\boldsymbol{\theta}}^+\}$. Then for*

$$\tau(z) := z\Phi(z) + \phi(z),$$

we have that

$$\max\left(I_t^{\boldsymbol{\theta}}(\mathbf{x}) - \varphi_t^{\boldsymbol{\theta}}\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta}), \frac{\tau(-\varphi_t^{\boldsymbol{\theta}}/\nu)}{\tau(\varphi_t^{\boldsymbol{\theta}}/\nu)} I_t^{\boldsymbol{\theta}}(\mathbf{x})\right) \leq \alpha_{\boldsymbol{\theta}}^{EI}(\mathbf{x}|\mathcal{D}_{t-1}) \leq I_t^{\boldsymbol{\theta}}(\mathbf{x}) + (\varphi_t^{\boldsymbol{\theta}} + \nu)\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})$$

Proof. If $\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta}) = 0$, then $\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1}) = I_t^{\boldsymbol{\theta}}(\mathbf{x})$ which makes the result trivial. Thus for the remainder of the proof, we assume that $\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta}) > 0$. Set $q = \frac{f(\mathbf{x})-\mu_{\boldsymbol{\theta}}^+}{\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})}$, and $u = \frac{\mu_{t-1}(\mathbf{x},\boldsymbol{\theta})-\mu_{\boldsymbol{\theta}}^+}{\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})}$. Then we have that

$$\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1}) = \nu\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})\tau\left(\frac{u}{\nu}\right).$$

By the assumption, we have that $|u-q| < \varphi_t^{\boldsymbol{\theta}}$. As $\tau'(z) = \Phi(z) \in [0,1]$, τ is non-decreasing and $\tau(z) \leq 1 + z$ for $z > 0$. Hence,

$$\begin{aligned}
\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1}) &\leq \nu\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})\tau\left(\frac{\max\{0,q\}+\varphi_t^{\boldsymbol{\theta}}}{\nu}\right) \\
&\leq \nu\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})\left(\frac{\max\{0,q\}+\varphi_t^{\boldsymbol{\theta}}}{\nu}+1\right) \\
&= I_t^{\boldsymbol{\theta}}(\mathbf{x}) + \left(\varphi_t^{\boldsymbol{\theta}}+\nu\right)\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})
\end{aligned}$$

If $I_t^{\boldsymbol{\theta}}(\mathbf{x}) = 0$, then the lower bound is trivial as $\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1})$ is non-negative. Thus suppose $I_t^{\boldsymbol{\theta}}(\mathbf{x}) > 0$. Since $\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1}) \geq 0$, and $\tau(z) \geq 0$ for all z, and $\tau(z) = z + \tau(-z) \geq z$. Therefore,

$$\begin{aligned}
\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1}) &\geq \nu\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})\tau\left(\frac{q-\varphi_t^{\boldsymbol{\theta}}}{\nu}\right) \\
&\geq \nu\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})\left(\frac{q-\varphi_t^{\boldsymbol{\theta}}}{\nu}\right) \\
&\geq I_t^{\boldsymbol{\theta}}(\mathbf{x}) - \varphi_t^{\boldsymbol{\theta}}\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta}). \qquad \text{(B.12)}
\end{aligned}$$

Also, as τ is increasing,

$$\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1}) \geq \nu\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta})\tau\left(\frac{-\varphi_t^{\boldsymbol{\theta}}}{\nu}\right). \qquad \text{(B.13)}$$

Combining (B.12) and (B.13), we get

$$\alpha_{\boldsymbol{\theta}}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1}) \geq \frac{\nu\tau(-\varphi_t^{\boldsymbol{\theta}}/\nu)}{\varphi_t^{\boldsymbol{\theta}}+\nu\tau(-\varphi_t^{\boldsymbol{\theta}}/\nu)} I_t^{\boldsymbol{\theta}}(\mathbf{x}) = \frac{\tau(-\varphi_t^{\boldsymbol{\theta}}/\nu)}{\tau(\varphi_t^{\boldsymbol{\theta}}/\nu)} I_t^{\boldsymbol{\theta}}(\mathbf{x})$$

which concludes the proof. □

Lemma 21. $\left|\mu_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) - \mu_{\boldsymbol{\theta}_t}^+\right| \leq \sqrt{\log(t-1+\sigma^2) - \log(\sigma^2)}\nu\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)$

Proof. For convenience, define $\mathbf{x}_t^+ = \arg\max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)$. Recall that

$$\mu_{\boldsymbol{\theta}_t}^+ = \max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x};\boldsymbol{\theta}_t).$$

Therefore, by the fact that $\alpha_{\boldsymbol{\theta}_t}^{\mathrm{EI}}(\mathbf{x}_t|\mathcal{D}_{t-1}) = \max_{\mathbf{x}\in\mathcal{X}} \alpha_{\boldsymbol{\theta}_t}^{\mathrm{EI}}(\mathbf{x}|\mathcal{D}_{t-1})$, we have

$$\begin{aligned}
&\nu\sigma_{t-1}(\mathbf{x}_t^+;\boldsymbol{\theta}_t)\tau(0) = \alpha_{\boldsymbol{\theta}_t}^{\mathrm{EI}}(\mathbf{x}_t^+|\mathcal{D}_{t-1}) \\
&\leq \alpha_{\boldsymbol{\theta}_t}^{\mathrm{EI}}(\mathbf{x}_t|\mathcal{D}_{t-1}) = \nu\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)\tau\left(\frac{\mu_{t-1}(\mathbf{x}_t,\boldsymbol{\theta}_t)-\mu_{\boldsymbol{\theta}_t}^+}{\nu\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)}\right), \qquad \text{(B.14)}
\end{aligned}$$

where τ is defined as in Lemma 20. We know that $\tau(0) = \frac{1}{\sqrt{2\pi}}$. Thus, equation (B.14) can be re-written as

$$\frac{\sigma_{t-1}(\mathbf{x}_t^+; \boldsymbol{\theta}_t)}{\sqrt{2\pi}} \leq \sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)\tau\left(\frac{\mu_{t-1}(\mathbf{x}_t, \boldsymbol{\theta}_t) - \mu_{\boldsymbol{\theta}_t}^+}{\nu\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)}\right). \tag{B.15}$$

By the definition of $\mu_{\boldsymbol{\theta}_t}^+$ we know that $\frac{\mu_{t-1}(\mathbf{x}_t, \boldsymbol{\theta}_t) - \mu_{\boldsymbol{\theta}_t}^+}{\nu\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)} \leq 0$. Therefore

$$\begin{aligned}\tau\left(\frac{\mu_{t-1}(\mathbf{x}_t, \boldsymbol{\theta}_t) - \mu_{\boldsymbol{\theta}_t}^+}{\nu\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)}\right) &\leq \phi\left(\frac{\mu_{t-1}(\mathbf{x}_t, \boldsymbol{\theta}_t) - \mu_{\boldsymbol{\theta}_t}^+}{\nu\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)}\right) \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\mu_{t-1}(\mathbf{x}_t, \boldsymbol{\theta}_t) - \mu_{\boldsymbol{\theta}_t}^+}{\nu\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)}\right)^2\right).\end{aligned} \tag{B.16}$$

Combining equations (B.15) and (B.16), we have

$$\left|\mu_t(\mathbf{x}_t; \boldsymbol{\theta}_t) - \mu_{\boldsymbol{\theta}_t}^+\right| \leq \sqrt{2\log\left(\frac{\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)}{\sigma_{t-1}(\mathbf{x}_t^+; \boldsymbol{\theta}_t)}\right)}\nu\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t).$$

Since $\log(\sigma_{t-1}(\mathbf{x}_t; \boldsymbol{\theta}_t)) \leq 0$, it remains to show that $-\log(\sigma_{t-1}^2(\mathbf{x}_t^+; \boldsymbol{\theta}_t)) \leq \log(t - 1 + \sigma^2) - \log\sigma^2$. To show this, it suffices to show that $\sigma_{t-1}^2(\mathbf{x}_t^+; \boldsymbol{\theta}_t) \geq \sigma^2/(t - 1 + \sigma^2)$. To see this, first note that $\sigma_{t-1}^2(\mathbf{x}_t^+; \boldsymbol{\theta}_t)$ is minimized if $\mathbf{x}_i = \mathbf{x}_t^+ \; \forall i \leq t - 1$. That is

$$\sigma_{t-1}^2(\mathbf{x}_t^+; \boldsymbol{\theta}_t) \geq 1 - \mathbf{1}^T(\mathbf{J} + \sigma^2 I)^{-1}\mathbf{1}$$

where $\mathbf{J}$ is a matrix of all ones. Notice that $\mathbf{J}$ is of rank 1. Let $\mathbf{Q}\boldsymbol{\Sigma}\mathbf{Q}^T$ be the eigen-decomposition of $\mathbf{J}$ such that $\Sigma_{1,1} = \lambda$ where λ is the only eigenvalue of $\mathbf{J}$. Since $\mathbf{1}$ is an eigenvector of $\mathbf{J}$, we know that $\lambda = \|\mathbf{1}\|_2^2$ and $\mathbf{Q}^T\mathbf{1} = [\|\mathbf{1}\|, 0, \cdots, 0]^T$. Because $(\mathbf{J} + \sigma^2\mathbf{I})^{-1} = \mathbf{Q}(\boldsymbol{\Sigma} + \sigma^2\mathbf{I})^{-1}\mathbf{Q}^T$, we have that $\mathbf{1}^T(\mathbf{J} + \sigma^2\mathbf{I})^{-1}\mathbf{1} = \frac{\|\mathbf{1}\|_2^2}{\|\mathbf{1}\|_2^2 + \sigma^2}$. Therefore

$$\sigma_{t-1}^2(\mathbf{x}_t^+; \boldsymbol{\theta}_t) \geq 1 - \frac{\|\mathbf{1}\|_2^2}{\|\mathbf{1}\|_2^2 + \sigma^2} = \frac{\sigma^2}{\|\mathbf{1}\|_2^2 + \sigma^2}$$

which concludes the proof since $\|\mathbf{1}\|_2^2 = t - 1$. □

B.4 Proof of main result

Proof of Theorem (1). We will need the following definitions $\mu_{\boldsymbol{\theta}_t}^+ = \max_{\mathbf{x}\in\mathcal{X}}\mu_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t)$ and $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}}\alpha_{\boldsymbol{\theta}_t}^{\text{EI}}(\mathbf{x}|\mathcal{D}_{t-1})$. By the Cauchy-Schwarz inequality,

$$\begin{aligned}|\mu_{t-1}(\mathbf{x}, \boldsymbol{\theta}_t) - f(\mathbf{x})| &\leq \left(\mathcal{K}_{t-1}^{\boldsymbol{\theta}_t}(\mathbf{x}, \mathbf{x})\right)^{1/2}\|\mu_{t-1}(\cdot; \boldsymbol{\theta}_t) - f(\cdot)\|_{\mathcal{K}_{t-1}^{\boldsymbol{\theta}_t}} \\ &= \sigma_{t-1}(\mathbf{x}; \boldsymbol{\theta}_t)\|\mu_{t-1}(\cdot; \boldsymbol{\theta}_t) - f(\cdot)\|_{\mathcal{K}_{t-1}^{\boldsymbol{\theta}_t}}.\end{aligned}$$

By Proposition 2 and the union bound, we know that $\|\mu_{t-1}(\cdot;\boldsymbol{\theta}_t) - f(\cdot)\|_{\mathcal{K}_{t-1}^{\boldsymbol{\theta}_t}} \leq \varphi_t^{\boldsymbol{\theta}_t}$ for all $t \geq 1$ holds with probability at least $1 - \sum_{t=1}^{\infty} \frac{6\delta}{\pi^2 t^2} = 1 - \delta$. Thus for the remainder of the proof, let us assume that $|\mu_{t-1}(\mathbf{x}, \boldsymbol{\theta}_t) - f(\mathbf{x})| \leq \varphi_t^{\boldsymbol{\theta}_t} \sigma_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)\ \forall t \in \mathbb{N}, \mathbf{x} \in \mathcal{X}$.

The regret at round t is

$$\begin{aligned} r_t &= f(\mathbf{x}^*) - f(\mathbf{x}_t) \\ &= (f(\mathbf{x}^*) - \mu^+_{\boldsymbol{\theta}_t}) - (f(\mathbf{x}_t) - \mu^+_{\boldsymbol{\theta}_t}) \\ &\leq I_t^{\boldsymbol{\theta}_t}(\mathbf{x}^*) + \left[(\mu^+_{\boldsymbol{\theta}_t} - \mu_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)) + \varphi_t^{\boldsymbol{\theta}_t}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)\right]. \end{aligned} \tag{B.17}$$

By Lemma 20, which defines the improvement as $I_t^{\boldsymbol{\theta}}(\mathbf{x}) = \max\{0, f(\mathbf{x}) - \mu^+_{\boldsymbol{\theta}}\}$, we know that $I_t^{\boldsymbol{\theta}_t}(\mathbf{x}^*) \leq \frac{\tau(\varphi_{t-1}^{\boldsymbol{\theta}_t}/\nu_t^{\boldsymbol{\theta}_t})}{\tau(-\varphi_{t-1}^{\boldsymbol{\theta}_t}/\nu_t^{\boldsymbol{\theta}_t})}\alpha^{\mathrm{EI}}_{\boldsymbol{\theta}_t}(\mathbf{x}^*|\mathcal{D}_{t-1})$. By the assumption on $\left(\nu_t^{\boldsymbol{\theta}_t}\right)^2$, there exists a constant C_3 such that $\left(\frac{\tau(\varphi_t^{\boldsymbol{\theta}_t}/\nu_t^{\boldsymbol{\theta}_t})}{\tau(-\varphi_t^{\boldsymbol{\theta}_t}/\nu_t^{\boldsymbol{\theta}_t})}\right) \leq C_3$. By Lemma 21, we also have that $(\mu^+_{\boldsymbol{\theta}_t} - \mu_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)) \leq \sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\nu_t^{\boldsymbol{\theta}_t}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t)$.

$$\begin{aligned} r_t &\leq C_3\alpha^{\mathrm{EI}}_{\boldsymbol{\theta}_t}(\mathbf{x}^*|\mathcal{D}_{t-1}) + \left(\sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\nu_t^{\boldsymbol{\theta}_t} + \varphi_t^{\boldsymbol{\theta}_t}\right)\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) \\ &\leq C_3\alpha^{\mathrm{EI}}_{\boldsymbol{\theta}_t}(\mathbf{x}_t|\mathcal{D}_{t-1}) + \left(\sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\nu_t^{\boldsymbol{\theta}_t} + \varphi_t^{\boldsymbol{\theta}_t}\right)\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) \\ &\leq C_3\left(I_t^{\boldsymbol{\theta}_t}(\mathbf{x}_t) + (\varphi_t^{\boldsymbol{\theta}_t} + \nu_t^{\boldsymbol{\theta}_t})\sigma_{t-1}(\mathbf{x};\boldsymbol{\theta}_t)\right) + \left(\sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\nu_t^{\boldsymbol{\theta}_t} + \varphi_t^{\boldsymbol{\theta}_t}\right)\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) \\ &\leq C_3(\mu_{t-1}(\mathbf{x}_t,\boldsymbol{\theta}_t) + \varphi_t^{\boldsymbol{\theta}_t}\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) - \mu^+_{\boldsymbol{\theta}_t})^+ \\ &\quad + \left((C_3+1)\varphi_t^{\boldsymbol{\theta}_t} + \left(C_3 + \sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\right)\nu_t^{\boldsymbol{\theta}_t}\right)\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) \\ &\leq \left((2C_3+1)\varphi_t^{\boldsymbol{\theta}_t} + \left(C_3 + \sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\right)\nu_t^{\boldsymbol{\theta}_t}\right)\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t). \end{aligned} \tag{B.18}$$

Define $(\varphi_t^L)^2 := C_2\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})} + 2\gamma_{t-1}^{\boldsymbol{\theta}^L} + \sqrt{8}\log^{1/2}(2t^2\pi^2/3\delta)\left(\sqrt{C_2}\|f\|_{\mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})} + \sqrt{\gamma_{t-1}^{\boldsymbol{\theta}^L}}\right) + 2\sigma\log(2t^2\pi^2/3\delta)$. By Lemma 19, we know that $\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}}(\mathcal{X})} \leq C_2\|f\|^2_{\mathcal{H}_{\boldsymbol{\theta}^U}(\mathcal{X})}$. Also, by Lemma 17, $\gamma_t^{\boldsymbol{\theta}_t} \leq \gamma_t^{\boldsymbol{\theta}^L}$. Therefore, $(\varphi_t^{\boldsymbol{\theta}_t})^2 \leq \left(\varphi_t^L\right)^2$.

$$\begin{aligned} r_t &\leq \left((2C_3+1)\varphi_t^L + \left(C_3 + \sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\right)\nu_t^{\boldsymbol{\theta}^L}\right)\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}_t) \\ &\leq \left((2C_3+1)\varphi_t^L + \left(C_3 + \sqrt{\log\left(\frac{t+\sigma^2}{\sigma^2}\right)}\right)\nu_t^{\boldsymbol{\theta}^L}\right)\sigma_{t-1}(\mathbf{x}_t;\boldsymbol{\theta}^L). \end{aligned} \tag{B.19}$$

To simplify notation, let t_0 be such that $\log\left(\frac{t_0}{\sigma^2}\right) > 1$ and WLOG assume $C_3 > 1$. Then

$$\begin{aligned}
\sum_{t=t_0}^{T} r_t^2 &\leq \sum_{t=t_0}^{T} (2C_3+1)^2 \left(\varphi_t^L + \sqrt{\log\left(\frac{t}{\sigma^2}\right)}\nu_t^{\boldsymbol{\theta}^L}\right)^2 \sigma_{t-1}^2(\mathbf{x};\boldsymbol{\theta}^L) \\
&\leq \sum_{t=t_0}^{T} 2(2C_3+1)^2 \left((\varphi_t^L)^2 + \log\left(\frac{t}{\sigma^2}\right)(\nu_t^{\boldsymbol{\theta}^L})^2\right) \sigma_{t-1}^2(\mathbf{x};\boldsymbol{\theta}^L) \\
&\leq 2(2C_3+1)^2 \left((\varphi_T^L)^2 + \log\left(\frac{T}{\sigma^2}\right)(\nu_T^{\boldsymbol{\theta}^L})^2\right) \sum_{t=t_0}^{T} \sigma_{t-1}^2(\mathbf{x};\boldsymbol{\theta}^L). \qquad \text{(B.20)}
\end{aligned}$$

By Lemma 18, we know that $\sum_{t=t_0}^{T} \sigma_{t-1}^2(\mathbf{x};\boldsymbol{\theta}^L) \leq \frac{2}{\log(1+\sigma^2)}\gamma_T^{\boldsymbol{\theta}^L}$. Finally, applying the Cauchy-Schwarz inequality yields $R_T^2 \leq T\sum_{t=1}^{T} r_t^2$ thus concluding the proof. □

Appendix C

Regret Bound for REMBO

Before embarking on the proof of Theorem 7, we introduce some definitions and state a few preliminary results, which we quote from [39] to facilitate the reading of this exposition.

We denote the Fourier transform of any function $\phi(\mathbf{x})$ as $\widehat{\phi}(\boldsymbol{\xi}) = \int_{\mathbb{R}^d} e^{-2\pi i \mathbf{x}^T \boldsymbol{\xi}} \phi(\mathbf{x}) d\mathbf{x}$. In this section we consider kernels of the form

$$k_{\boldsymbol{\Delta}}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = K_{\boldsymbol{\Delta}}(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}) = K(\boldsymbol{\Delta}^{-\frac{1}{2}}(\mathbf{x}^{(1)} - \mathbf{x}^{(2)}))$$

where $\boldsymbol{\Delta}$ is a positive definite matrix and K has Fourier transform $\widehat{K}$ such that $\widehat{K}$ is isotropic and radially non-increasing. Notice that both the squared exponential kernel and the skew squared exponential kernel introduced in Definition 4 of the main text are represented in the form above. The popular kernels from the Matérn class can also be represented in this form. In general, the results in this section would follow for any kernel that satisfies the four assumptions detailed by [39].

Lemma 22 (Lemma 1 of [39]). *$\mathcal{H}(\mathbb{R}^d)$ is the space of real continuous functions $f \in L^2(\mathbb{R}^d)$ whose norm*

$$\|f\|^2_{\mathcal{H}(\mathbb{R}^d)} := \int \frac{|\widehat{f}(\boldsymbol{\xi})|^2}{\widehat{K}(\boldsymbol{\xi})} d\boldsymbol{\xi}$$

is finite, taking $0/0 = 0$.

Lemma 23 (Lemma 2 of [39]). *Given a set $\mathcal{S} \subseteq \mathbb{R}^d$, $\mathcal{H}(\mathcal{S})$ is the space of functions $f = g|\mathcal{S}$ (f is g restricted to $\mathcal{S}$) for some $g \in \mathcal{H}(\mathbb{R}^d)$, with norm*

$$\|f\|_{\mathcal{H}(\mathcal{S})} := \inf_{g|\mathcal{S}=f} \|g\|_{\mathcal{H}(\mathbb{R}^d)},$$

and there is a unique g minimizing this expression.

Lemma 24 (Lemma 4 of [39], extended to our setting). *Let $\mathcal{S} \subseteq \mathbb{R}^d$ and $\boldsymbol{\Delta}$ and $\boldsymbol{\Delta}'$ be two symmetric positive definite matrices. Let $\lambda_{\max}(\boldsymbol{\Delta}')$, $\lambda_{\min}(\boldsymbol{\Delta})$ be the largest and the smallest*

eigenvalues of $\mathbf{\Delta}'$, $\mathbf{\Delta}$ *respectively such that* $\lambda_{\max}(\mathbf{\Delta}') \leq \lambda_{\min}(\mathbf{\Delta})$. *Then* $f \in \mathcal{H}_{k_{\mathbf{\Delta}}}(\mathcal{S})$ *implies* $f \in \mathcal{H}_{k_{\mathbf{\Delta}'}}(\mathcal{S})$ *and also*

$$\|f\|_{\mathcal{H}_{k_{\mathbf{\Delta}'}}(\mathcal{S})} \leq \left(\frac{|\mathbf{\Delta}|}{|\mathbf{\Delta}'|}\right)^{\frac{1}{2}} \|f\|_{\mathcal{H}_{k_{\mathbf{\Delta}}}(\mathcal{S})}$$

where $|\mathbf{\Delta}|$ *is the determinant of* $\mathbf{\Delta}$.

Proof. Since $\mathbf{\Delta}$ and $\mathbf{\Delta}'$ are positive definite we can write $\mathbf{\Delta} = Q^T \Sigma Q$ and $\mathbf{\Delta}' = Q'^T \Sigma' Q'$ where Q and Q' are orthonormal. Let $C = \left(\frac{|\mathbf{\Delta}'|}{|\mathbf{\Delta}|}\right)^{\frac{1}{2}}$. Since $\lambda_{\max}(\mathbf{\Delta}') \leq \lambda_{\min}(\mathbf{\Delta})$, we know that

$$\|\Sigma'^{\frac{1}{2}} Q' \boldsymbol{\xi}\|^2 = \boldsymbol{\xi}^T \mathbf{\Delta}' \boldsymbol{\xi} \leq \boldsymbol{\xi}^T \mathbf{\Delta} \boldsymbol{\xi} = \|\Sigma^{\frac{1}{2}} Q \boldsymbol{\xi}\|^2.$$

As $\widehat{K}$ is isotropic and radially non-increasing, we have that

$$\widehat{K_{\mathbf{\Delta}'}}(\boldsymbol{\xi}) = |\mathbf{\Delta}'|^{\frac{1}{2}} \widehat{K}(\Sigma'^{\frac{1}{2}} Q' \boldsymbol{\xi}) \geq |\mathbf{\Delta}'|^{\frac{1}{2}} \widehat{K}(\Sigma^{\frac{1}{2}} Q \boldsymbol{\xi}) = |\mathbf{\Delta}'|^{\frac{1}{2}} |\mathbf{\Delta}|^{-\frac{1}{2}} \widehat{K_{\mathbf{\Delta}}}(\boldsymbol{\xi}) = C \widehat{K_{\mathbf{\Delta}}}(\boldsymbol{\xi})$$

where the first and the second last equality follows from the following property of Fourier transforms: $\widehat{h}(\boldsymbol{\xi}) = \frac{1}{|M|} \widehat{f}(M^{-T} \boldsymbol{\xi})$ if $h(\mathbf{x}) = f(M\mathbf{x})$ given a non-singular matrix M. Given $f \in \mathcal{H}_{k_{\mathbf{\Delta}}}(\mathcal{S})$, let $g \in \mathcal{H}_{k_{\mathbf{\Delta}}}(\mathbb{R}^d)$ be its minimum norm extension, as in Lemma 23. By the definition of RKHS norm in Lemma 22,

$$\|f\|^2_{\mathcal{H}_{k_{\mathbf{\Delta}'}}(\mathcal{S})} = \|g\|^2_{\mathcal{H}_{k_{\mathbf{\Delta}'}}(\mathbb{R}^d)} = \int \frac{|\widehat{g}|^2}{\widehat{K_{\mathbf{\Delta}'}}} \leq \int \frac{|\widehat{g}|^2}{C\widehat{K_{\mathbf{\Delta}}}} = C^{-1} \|f\|^2_{\mathcal{H}_{k_{\mathbf{\Delta}}}(\mathcal{S})}.$$

Since C is finite, by Lemma 22 we have that $f \in \mathcal{H}_{k_{\mathbf{\Delta}}}(\mathcal{S})$ implies $f \in \mathcal{H}_{k_{\mathbf{\Delta}'}}(\mathcal{S})$. □

Definition 9. *Given a map* $\pi : \mathcal{S} \to \mathcal{T}$ *between any two sets* $\mathcal{S}$ *and* $\mathcal{T}$, *and any map* $f : \underbrace{\mathcal{T} \times \cdots \times \mathcal{T}}_{n\text{-times}} \to \mathbb{R}$, *with* $n \geq 1$, *we define the* pull-back *of* f *under* π *as follows:*

$$\pi^* f(s_1, \ldots, s_n) := f\left(\pi(s_1), \ldots, \pi(s_n)\right).$$

That is, one evaluates the pull-back $\pi^* f$ *on points in* $\mathcal{S}$ *by first "pushing them forward" onto* $\mathcal{T}$ *and then using* f *to get a number.*

If the map π *is given by a matrix* $\mathbf{A}$, *we will use the notation* $\mathbf{A}^* f$ *for the pull-back of* f *under the linear map induced by* $\mathbf{A}$. *Moreover, given a matrix* $\mathbf{A}$ *and a set* $\mathcal{S}$ *in its target space, we will denote by* $\mathbf{A}^{-1}(\mathcal{S})$ *the set of all points that are mapped into* $\mathcal{S}$ *by* $\mathbf{A}$.

Proposition 3 (Theorem 2 by [39], paraphrased for our particular setting)**.** *Given a squared exponential kernel* k_ℓ *on a compact subset* $\mathcal{Y} \subset \mathbb{R}^d$ *and a function* $f \in \mathcal{H}_\ell(\mathcal{Y})$, *then applying Expected Improvement to* f *results in simple regret that diminishes according to* $\mathcal{O}\left(t^{-\frac{1}{d}}\right)$, *with the constants worsening as the norm* $\|f\|_{\mathcal{H}_\ell(\mathcal{Y})}$ *increases.*

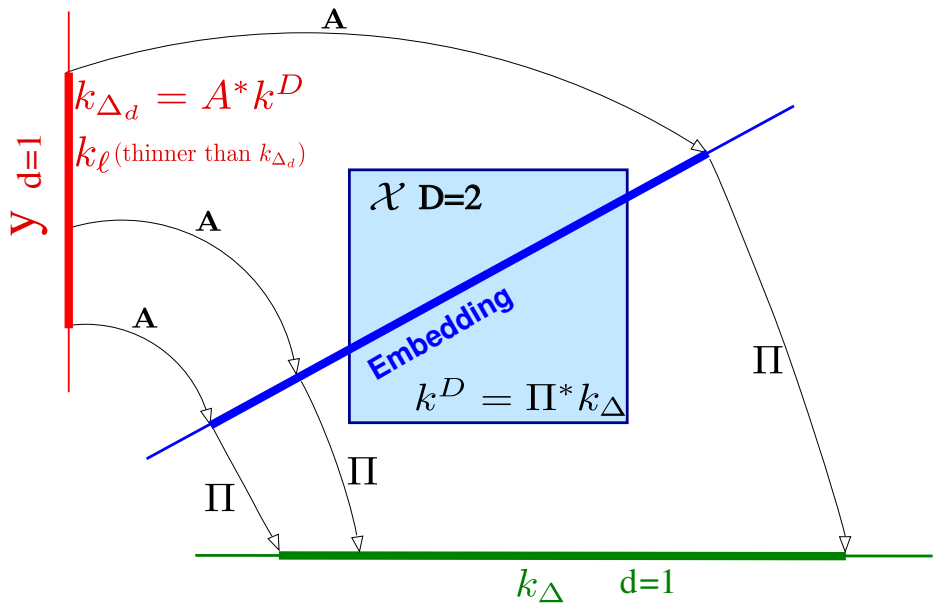


Figure C.1: A illustration of the different kernels defined in this proof. In this figure, the red line represents the one-dimensional space $\mathcal{Y}$ over which we optimize. The blue region is the original high-dimensional space. The green line represents the effective subspace $\mathcal{T}$ projected to a one-dimensional space through $\mathbf{\Pi}$. The true kernel $K_{\mathbf{\Delta}}$ is defined on the green region. The true kernel on the high-dimensional space is then naturally $k^D = \mathbf{\Pi}^* k_{\mathbf{\Delta}}$ and the true kernel on $\mathcal{Y}$ is $K_{\mathbf{\Delta}_d} = \mathbf{A}^* k^D$. To optimize without the knowledge of $K_{\mathbf{\Delta}_d}$, we use a kernel k_ℓ which with high probability is thinner than $K_{\mathbf{\Delta}_d}$ in all directions thus preserving convergence properties.

Proof of Theorem 7. The proof of this result is structured into two parts. In the first part of the proof, we give an analytic expression for the true kernel $k_{\mathbf{\Delta}_d}$ in the low dimensional space over which we optimize. In the second part of the proof, we show that this kernel is well-behaved with high probability (specifically the maximum and minimum eigenvalues of $\mathbf{\Delta}_d$ are bounded above and below by constants depending on the probability of failure) and apply Proposition 3 to acquire the convergence rate.

Let $\mathbf{\Phi} \in \mathbb{R}^{D \times d}$ be a matrix, whose columns form an orthonormal basis for $\mathcal{T}$. Let $\mathbf{\Pi} = \mathbf{\Phi}^T \mathbf{\Phi} \mathbf{\Phi}^T : \mathcal{X} \to \mathbb{R}^d$. Note that $\mathbf{\Pi}$ is composed of an orthogonal projection from $\mathcal{X}$ to $\mathcal{T}$ and a bijective map from $\mathcal{T}$ to $\mathbb{R}^d$. We will also denote the corresponding matrix by $\mathbf{\Pi}$.

Recall from the theorem statement that $\overline{f|_{\mathcal{T}}}$ is assumed to be an element of the RKHS $\mathcal{H}_{k_{\mathbf{\Delta}}}$, and that we have $f = \mathbf{\Pi}^* \overline{f|_{\mathcal{T}}}$, i.e. f is obtained from "stretching $f|_{\mathcal{T}}$ open" along the orthogonal subspace of $\mathcal{T}$. We can also define the kernel over $\mathbb{R}^D$ by $k^D := \mathbf{\Pi}^* k_{\mathbf{\Delta}}$.

Now, given the embedding $\mathbb{R}^d \hookrightarrow \mathbb{R}^D$ defined by the matrix $\mathbf{A}$, the pull-back function $\mathbf{A}^* f$ is an element of the RKHS $\mathcal{H}_{\mathbf{A}^* k^D}$: Henceforth, we will use the notation

$$k_{\mathbf{\Delta}_d} := \mathbf{A}^* k^D = \mathbf{A}^* \mathbf{\Pi}^* k_{\mathbf{\Delta}} = (\mathbf{\Pi A})^* k_{\mathbf{\Delta}}.$$

In more detail, for $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathbb{R}^d$

$$k_{\mathbf{\Delta}_d}(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = K\left((\mathbf{y}^{(1)} - \mathbf{y}^{(2)})^\top \mathbf{A}^\top \mathbf{\Pi}^\top \mathbf{\Delta}^{-1} \mathbf{\Pi} \mathbf{A} (\mathbf{y}^{(1)} - \mathbf{y}^{(2)})\right).$$

For a pictorial illustration of the different kernels defined in this proof, please refer to Figure C.1.

Since $\mathbf{\Pi}$ is an orthogonal projection matrix, it has an SVD decomposition $\mathbf{\Pi} = \mathbf{USV}$ consisting of an orthogonal $d \times d$ matrix $\mathbf{U}$, an orthogonal $D \times D$ matrix $\mathbf{V}$ and a $d \times D$ matrix $\mathbf{S}$ that has the following form:

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{bmatrix}.$$

Now, given a fixed orthogonal matrix $\mathbf{O} \in \mathbb{R}^{D \times D}$ and a random Gaussian vector $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{D \times D})$, due to the rotational symmetry of the normal distribution, the vector $\mathbf{Ov}$ is also a sample from $\mathcal{N}(\mathbf{0}, \mathbf{I}_{D \times D})$. Therefore, given a random Gaussian matrix Γ, $\mathbf{O}\Gamma$ is also a random Gaussian matrix with the same distribution of entries. Moreover, given $\mathbf{S}$ as above, $\mathbf{S}\Gamma$ is a $d \times D$ random Gaussian matrix, since multiplying any matrix by $\mathbf{S}$ on the left simply extracts the first d rows of the matrix.

Given this, if we fix an orthogonal decomposition $\mathbf{\Delta}^{-1} = \mathbf{P}^\top \mathbf{D}^{-1} \mathbf{P}$, where $\mathbf{P}$ is orthogonal and $\mathbf{D}$ is a diagonal matrix with the eigenvalues of $\mathbf{\Delta}$ along the diagonal, we can conclude that

$$\mathbf{G} := \mathbf{P\Pi A} = \mathbf{PUSVA}$$

is a random Gaussian matrix, and so the matrix $\mathbf{\Delta}_d = \mathbf{A}^\top \mathbf{\Pi}^\top \mathbf{\Delta}^{-1} \mathbf{\Pi A}$ can be decomposed into random Gaussian and diagonal matrices as follows:

$$\mathbf{\Delta}_d^{-1} = \mathbf{G}^\top \mathbf{D}^{-1} \mathbf{G}.$$

Since random Gaussian matrices, as argued in the proof of Theorem 3, have full rank almost surely, $\mathbf{\Delta}_d^{-1}$ is of full rank and is positive definite.

In the remainder of this proof, we replace $k_{\mathbf{\Delta}_d}$ with a kernel k_ℓ that is "thinner" than $k_{\mathbf{\Delta}_d}$ and so $\mathbf{A}^* f$ is also an element of the RKHS of k_ℓ. By showing that this is true, REMBO (which uses k_ℓ) has enough approximation power. Moreover, the statement of Proposition 3 applies.

Let $s_{\min}(\mathbf{M})$ and $s_{\max}(\mathbf{M})$ denote the smallest and the largest singular values of a matrix $\mathbf{M}$. With this notation in hand, we point out the following two facts about concentration of singular values:

I. Since for any pair of matrices $\mathbf{A}$ and $\mathbf{B}$, we have $s_{\max}(\mathbf{AB}) \leq s_{\max}(\mathbf{A})s_{\max}(\mathbf{B})$, we get

$$\frac{1}{\lambda_{\min}(\boldsymbol{\Delta}_d)} = \lambda_{\max}(\boldsymbol{\Delta}_d^{-1}) \leq s_{\max}(\mathbf{G})^2 s_{\max}(\mathbf{D}^{-1}) \leq \frac{s_{\max}(\mathbf{G})^2}{r^2}$$

and since $\mathbf{G}$ is a random matrix with Gaussian entries, we have (cf. Equation 2.3 by [182])

$$P\left(s_{\max}(\mathbf{G}) < 2\sqrt{d} + t\right) \leq 1 - 2e^{-t^2/2},$$

and so with probability $1 - \frac{\epsilon}{2}$, we have

$$s_{\max}(\mathbf{G}) < 2\sqrt{d} + \sqrt{2\ln\frac{4}{\epsilon}}.$$

Therefore, with probability $1 - \frac{\epsilon}{2}$, we have

$$\lambda_{\min}(\boldsymbol{\Delta}_d) > \left(\frac{r}{2\sqrt{d} + \sqrt{2\ln\frac{4}{\epsilon}}}\right)^2. \tag{C.1}$$

Henceforth, we will use the notation

$$\ell = \ell(\epsilon) := \frac{r}{2\sqrt{d} + \sqrt{2\ln\frac{4}{\epsilon}}}$$

II. On the other hand, we have

$$\frac{1}{\lambda_{\max}(\boldsymbol{\Delta}_d)} = \lambda_{\min}(\boldsymbol{\Delta}_d^{-1}) \geq s_{\min}(\mathbf{G})^2 s_{\min}(\mathbf{D}^{-1}) \geq \frac{s_{\min}(\mathbf{G})^2}{R^2}$$

together with the following probabilistic bound on $s_{\min}(\mathbf{G})$ (cf. Equation 3.2 by [182]):

$$P\left(s_{\min}(\mathbf{G}) > \frac{\delta}{\sqrt{d}}\right) > 1 - \delta.$$

So, with probability $1 - \frac{\epsilon}{2}$, we have

$$s_{\min}(\mathbf{G}) > \frac{\epsilon}{2\sqrt{d}},$$

and so

$$\lambda_{\max}(\boldsymbol{\Delta}_d) < \frac{4dR^2}{\epsilon^2} \tag{C.2}$$

holds with probability $1 - \frac{\epsilon}{2}$.

In what follows, we will use the notation:

$$U = U(\epsilon) := \frac{2R\sqrt{d}}{\epsilon}$$

Now, with these estimates in hand, we have that by Lemma 23 and Lemma 24 the following bound holds with probability $1-\epsilon$:

$$\|\mathbf{A}^* f\|_{\mathcal{H}_\ell(\mathbf{A}^{-1}(\mathcal{X}))} \leq \|\mathbf{A}^* f\|_{\mathcal{H}_\ell(\mathbb{R}^d)} \leq \left(\frac{U(\epsilon)}{\ell(\epsilon)}\right)^{\frac{d}{2}} \|\mathbf{A}^* f\|_{\mathcal{H}_{\boldsymbol{\Delta}_d}(\mathbb{R}^d)} \tag{C.3}$$

Since the transformation $\boldsymbol{\Pi}\mathbf{A}$ is invertible, we have that the map $(\boldsymbol{\Pi}\mathbf{A})^* : \mathcal{H}_{\boldsymbol{\Delta}}(\mathbb{R}^d) \to \mathcal{H}_{\boldsymbol{\Delta}_d}(\mathbb{R}^d)$ (recall that $k_{\boldsymbol{\Delta}_d} = k_{(\boldsymbol{\Pi}\mathbf{A})^*\boldsymbol{\Delta}}$) that sends $g \in H_{\boldsymbol{\Delta}}$ to $(\boldsymbol{\Pi}\mathbf{A})^* g$ is an isomorphism of Hilbert spaces and so

$$\left\|\overline{f|_{\mathcal{T}}}\right\|_{\mathcal{H}_{\boldsymbol{\Delta}}(\mathbb{R}^d)} = \|\mathbf{A}^* f\|_{\mathcal{H}_{\boldsymbol{\Delta}_d}(\mathbb{R}^d)} \tag{C.4}$$

since we have $\mathbf{A}^* f = \mathbf{A}^* \left(\boldsymbol{\Pi}^* \overline{f|_{\mathcal{T}}}\right) = (\boldsymbol{\Pi}\mathbf{A})^* \overline{f|_{\mathcal{T}}}$.

By combining C.3 and C.4, we have that

$$\|\mathbf{A}^* f\|_{\mathcal{H}_\ell(\mathbf{A}^{-1}(\mathcal{X}))} \leq \left(\frac{U(\epsilon)}{\ell(\epsilon)}\right)^{\frac{d}{2}} \left\|\overline{f|_{\mathcal{T}}}\right\|_{\mathcal{H}_{\boldsymbol{\Delta}}(\mathbb{R}^d)}.$$

Now that we know that the $\mathcal{H}_\ell(\mathbb{R}^d)$ norm of $\mathbf{A}^* f$ is finite, we can apply the Expected Improvement algorithm to it on the set $\mathbf{A}^{-1}(\mathcal{X})$ with kernel k_ℓ, instead of the unknown kernel $k_{\boldsymbol{\Delta}_d}$, and then Proposition 3 tells us that the simple regret would be in $\mathcal{O}\left(t^{-\frac{1}{d}}\right)$. □