General-purpose Information-theoretical Bayesian Optimisation

A thesis by acronyms

Henry B. Moss

Submitted for the degree of Doctor of Philosophy at Lancaster University.

February 2021

STOR-i
excellence with impact

Abstract

Bayesian optimisation (BO) is an increasingly popular strategy for optimising functions with substantial query costs. By sequentially focusing evaluation resources into promising areas of the search space, BO is able to find reasonable solutions within heavily restricted evaluation budgets. Consequently, BO has become the *de facto* approach for fine-tuning the hyper-parameters of machine learning models and has had numerous successful applications in industry and across the experimental sciences.

This thesis seeks to increase the scope of information-theoretic BO, a popular class of search strategies that regularly achieves state-of-the-art optimisation. Unfortunately, current information-theoretic BO routines require sophisticated approximation schemes that incur substantially large computational overheads and are, therefore, applicable only to optimisation problems defined over low-dimensional and Euclidean search spaces. This thesis proposes information-theoretic approximations that extend the Max-value Entropy Search of Wang and Jegelka (2017) to a much wider class of optimisation tasks, including noisy, batch and multi-fidelity optimisation across both Euclidean and highly-structured discrete spaces. To comprehensively test our proposed search strategies, we construct novel frameworks for performing BO over the highly-structured string spaces that arise in synthetic gene design and molecular search problems, as well as for objective functions with controllable observation noise. Finally, we demonstrate the real-world applicability of BO as part of a sophisticated machine learning pipeline for fine-tuning multi-speaker text-to-speech models .

Acknowledgements

This PhD is the accumulation of work with many collaborators. First and foremost, I wish to thank my supervisors David Leslie and Paul Rayson for their unyielding guidance and support. I'm particularly thankful for David's effortless ability to rescue me from mathematical 'rabbit holes' and, perhaps more importantly, for allowing me the freedom to venture down these holes in the first place. David and I have shared passions outside of work in cycling and playing jazz. I am grateful for his route suggestions, rides in Bristol, Strava kudos and opportunities to play together in the 'Northern All-Stars Big Band'. Paul shared with me his passion with natural language processing, for which I will be forever grateful. This fascinating and challenging application of machine learning formed the initial foundations of this thesis.

Outside of my core supervisory team, I was privileged to have many other excellent mentors. I would like to thank Javier González for his support, enthusiasm and perceptive comments across our many collaborations. I am also thankful for the opportunities he provided at Amazon, as well as introducing me to the one and only Roberto Barra-Chicote. I believe it unlikely that I will ever meet anyone with such a great passion for science than Roberto, from whom I learnt a great deal about research within industry. I am particularly grateful to Daniel Beck, whose expertise was crucial in forming the most interesting part of this thesis, the creation of which required a trip to Melbourne (and a hasty retreat from a pandemic). This visit was enabled by Daniel's generosity but it was his openness and friendliness that gave me the confidence to fly around the world in the first place.

My PhD was cultivated in the stimulating research environment provided by the EPSRC-funded STOR-i doctoral training centre. I am grateful to Jon Tawn, Idris

Eckley, Kevin Glazebrook, Jen Bull, Kim Wilson, Wendy Shimmin and Nicola Sargent for their work in building and maintaining STOR-i, as well as all the personal support they provided over the past three years.

None of the above could have occurred without the support of Lin McIntosh, who introduced my 16-yr old self to the power and beauty of mathematics. Her tireless work was crucial in building my love for science and she taught me many of the skills upon which this thesis relies.

Finally, I wish to acknowledge the support of my family during all my academic pursuits. Thank you to Katie Miles, for all our adventures and your endless support and kindness. Thank you also to my parents and grandparents for the love and encouragement you have provided from my first year at school all the way up to this final twentieth year.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

Chapter 3 has been published as Moss H. B., Leslie D. S. & Rayson P., MUMBO: MUlti-task Max-value Bayesian Optimisation, *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020.

Chapter 4 has been published as Moss H. B., Beck D., Leslie D. S., Gonzalez J. & Rayson P., Bayesian Optimisation over String spaces, *Advances in Neural Information Processing Systems*, 2020.

Chapter 5 has been submitted to *The Journal of Machine Learning Research*.

An early version of chapter 6 was presented at the Workshop on Real World Experimental Design and Active Learning during *The International Conference on Machine Learning*, 2020.

Chapter 7 has been published as Moss H. B., Aggarwal V., Prateek N., Gonzalez J. & Barra-Chicote R., BOFFIN TTS: Few-shot Speaker Adaptation by Bayesian Optimisation, *The International Conference on Acoustics, Speech and Signal Processing*, 2020.

All coauthors acted in supervisory roles, helping fine-tune ideas and the final manuscripts. I conceived the core ideas, formulated mathematical derivations, and built all code contributions.

The word count for this thesis is 41,000.

Henry Moss

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Countless problems across machine learning, operational research, science and engineering can be framed as optimisation tasks. Sometimes these problems have properties that can be exploited to yield efficient optimisation, for example convex objective functions permit gradient-based methods and some polynomial objective functions can be tackled by mathematical programming. However, many objective functions do not have such clear properties, with only weak prior knowledge available about their structure. Moreover, as these optimisation tasks are plagued by substantial evaluation costs, most standard optimisation routines are unsuitable as they require many evaluations. Function evaluation costs can be monetary, for example the significant compute required to fine-tune deep-learning models (Yu and Zhu, 2020), supply-chain simulators (Pasupathy and Henderson, 2011) and climate models (Hourdin et al., 2017), or evaluating the objective function could require resource and labour-consuming lab tests, for example when designing molecular structures (MacLeod et al., 2020), gene sequences (Yu et al., 2013) or aerodynamic profiles (Daniels et al., 2018). This broad class of so-called “black-box" optimisation problem, typically characterised by expensive and noisy evaluations, a lack of accessible gradients and high non-convexity, is the focus of this thesis.

A popular solution to high-cost "black-box" optimisation tasks has arisen in Bayesian optimisation (Mockus et al., 1978). As an extension of response surface methods (Hill and Hunter, 1966), Bayesian optimisation uses cheap probabilistic

surrogate models to predict the value of the objective function at previously unobserved locations. Heuristic search strategies can then be defined to explicitly control the balance of exploitation and exploration in subsequent evaluations, typically focusing evaluation resources into areas of the search space with either promising predictions or where there is high uncertainty.

A particularly intuitive and empirically effective class of search strategies are those based on information theory (Cover and Thomas, 2012), a powerful framework from the interface of statistics and theoretical computer science that provides a meaningful measurement of uncertainty. Information-theoretic arguments are particularly well suited to Bayesian optimisation (Hennig and Schuler, 2012), as they provide a clear measure of the utility (the information gained) of making a particular evaluation.

Unfortunately, the application of information-theoretic search strategies in Bayesian optimisation has been plagued by computational issues, with most existing information-theoretic search strategies requiring sophisticated and expensive approximation schemes for even the most simple optimisation tasks. Moreover, information-theoretic strategies proposed for popular extended Bayesian optimisation frameworks, for example those exploiting parallel computing resources or low-fidelity evaluations, incur even larger computational overheads. Therefore, information-theoretic search is currently suitable only for optimisation problems where function query costs are sufficiently large to overshadow very significant optimisation overheads. Another important practical consideration, is that many of the approximations employed in information-theoretic search rely on exploiting properties specific to continuous and fixed-dimensional search spaces, a further serious limitation on their applicability. Consequently, information-theoretic search strategies have yet to be applied for optimisation over non-Euclidean spaces, as demanded by many of the high-cost optimisation tasks mentioned above.

Motivated by the empirical success of information-theory within the few settings where it is currently feasible, the goal of this thesis is to provide new information-theoretic search strategies suitable for a much wider class of Bayesian optimisation problems. We achieve this by proposing a series of novel information-theoretic approximation strategies that are simpler, cheaper and require fewer assumptions on the

properties of the search space than current techniques. We demonstrate, across a wide range of Bayesian optimisation frameworks including noisy, batch, multi-fidelity and string optimisation, and across an even wider range of problems, from hyper-parameter tuning, molecular search, synthetic gene design, reinforcement learning and simulation optimisation, that our computationally light-weight yet high-performing search strategies improve upon the current state-of-the-art in Bayesian optimisation.

1.1 Thesis Structure

In Chapter 2 we will introduce Bayesian optimisation and give an overview of popular extensions. In particular, we will focus on multi-fidelity Bayesian optimisation, batch Bayesian optimisation, and Bayesian optimisation for structured spaces, as these problems motivate the work contained in the remainder of the thesis. Our main contributions are presented as Chapters 3 to 7, each of which has either been published or is currently in submission as a standalone paper. We have included these papers in their published form, with the only major change being the inclusion of preface that extends each abstract to summarise how each contribution fits into the wider narrative of the thesis. We now summarise the main contributions of each of the core chapters of the thesis.

- In Chapter 3, we present MUMBO, the first computationally light-weight information-theoretic approach for multi-task and multi-fidelity Bayesian optimisation. Although outperformed by the subsequent work proposed in Chapter 5, MUMBO provides a comprehensive summary of information-theoretical multi-fidelity optimisation and helps prepare the reader for the more sophisticated approximation strategies that appear later in the thesis.

  The work in this chapter appeared as: Moss H. B., Leslie D. S. & Rayson P., MUMBO: MUlti-task Max-value Bayesian Optimisation, *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020.

- Chapter 4 describes BOSS, a BO framework for high-cost string design problems. Our approach builds a powerful surrogate model based on string kernels and employs genetic algorithms to explore search spaces of strings that follow syntactic constraints. Although BOSS does not use information-theoretic techniques, it forms a challenging non-Euclidean test-case for the information-theoretic approach of Chapter 5.

  The work in this chapter appeared as: Moss H. B., Beck D., Leslie D. S., Gonzalez J. & Rayson P., Bayesian Optimisation over String spaces, *The Conference on Neural Information Processing Systems*, 2020.

- Chapter 5 presents the primary contribution of the thesis through General-purpose Information-Based Bayesian OptisatioN (GIBBON), an information-theoretic search strategy supporting a range of popular Bayesian optimisation problems, including noisy, multi-fidelity and batch optimisations in both continuous and highly-structured spaces. Our principled derivation of GIBBON also provides the first explicit connection between information-theoretic Bayesian optimisation and probabilistic repulsion models. We investigate GIBBON's efficacy and generality across a range of tasks including the multi-fidelity problems introduced in Chapter 3, as well as using GIBBON to provide a batch extension of the BOSS framework of Chapter 4.

  The work in this chapter is in submission for *The Journal of Machine Learning Research*.

- When optimising functions with stochastic evaluations, such as parameter tuning and simulation optimisation, it is common to instead average of a fixed set of noisy realisations of the objective function, for example when using K-fold cross validation (Kohavi, 1995) or sample average approximations (Kleywegt et al., 2002). However, disregarding the true objective function in this manner finds a high-precision optimum of the wrong function. Chapter 6 considers this problem and proposes BOSH, a Bayesian optimisation framework that maintains a growing pool of realisations as the optimisation progresses. BOSH forms a

challenging batch and multi-task Bayesian optimisation problem that further tests the efficacy and generality of our GIBBON search strategy.

A condensed version of the work in this chapter was presented at the Workshop on Real World Experimental Design and Active Learning during *The International Conference on Machine Learning*, 2020.

- Our final chapter considers a practical application of Bayesian optimisation within Amazon Alexa's text-to-speech system. Chapter 7 was completed during an internship at Amazon Research and serves to demonstrate the effectiveness and applicability of Bayesian optimisation in the real-world.

  The work in this chapter appeared as: Moss H. B., Aggarwal V., Prateek N., Gonzalez J. & Barra-Chicote R., BOFFIN TTS: Few-shot Speaker Adaptation by Bayesian Optimisation, *The International Conference on Acoustics, Speech and Signal Processing*, 2020.

- Chapter 8 concludes the thesis, summarising the primary contributions and outlining areas of potential future research.

Chapter 2

Background

2.1 Bayesian Optimisation

A popular solution for the optimisation of high-cost "black-box" functions has arisen in Bayesian optimisation (BO) (Mockus et al., 1978). Formally, BO seeks to find the maximiser

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, g(\mathbf{x}), \tag{2.1.1}$$

over a d-dimensional [1] search space $\mathcal{X} \in \mathbb{R}^d$ whilst incurring as few evaluations of the expensive objective function g as possible. By sequentially deciding where to make each evaluation as the optimisation progresses, BO can direct resources into evaluating promising areas of the search space, thus providing highly efficient optimisation. More precisely, BO's decisions are governed by two components - a surrogate model and an acquisition function (as discussed in depth in Sections 2.1.1 and 2.1.2). For now, we give a high-level overview of BO for a generic choice of surrogate model and acquisition function.

Suppose that we wish to evaluate the objective function for the $n + 1^{th}$ time. By fitting a surrogate model to the n previously collected objective function location-evaluation tuples $D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ (where y_i is the potentially noisy evaluation of

[1] For simplicity, we first focus on fixed-dimensional search spaces in this introduction. Discrete and highly structured search spaces are introduced at the end of this Section and in Chapter 5.

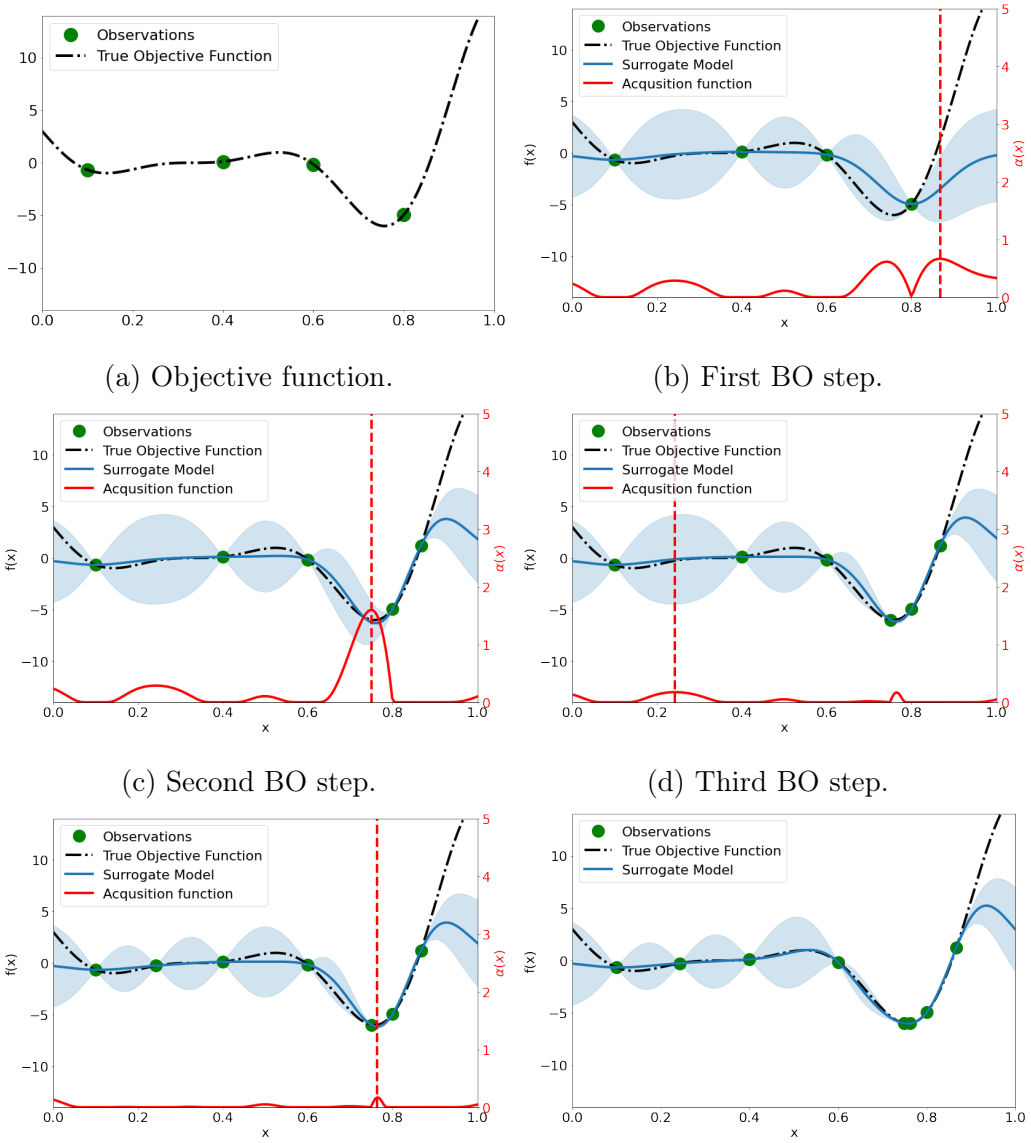
the objective function at location $\mathbf{x}_i$), we can build a probabilistic model for the objective function that summarises our current belief about which areas of the search space maximise our objective function. An acquisition function $\alpha_n : \mathcal{X} \to \mathbb{R}$ then uses the surrogate model to predict the utility of evaluating at a particular location in the search space, producing large values at promising locations. We choose our next objective function evaluation to be the maximiser of this acquisition function, i.e selecting

$$\mathbf{x}_{n+1} = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, \alpha_n(\mathbf{x}).$$

After evaluating g at $\mathbf{x}_{n+1}$, we refit our surrogate model to include the new evaluation and repeat the whole process until the optimisation budget is exhausted. Figure 2.1.1 illustrates a simple BO loop over four iterations, demonstrating fast convergence to the true minima. Panel 2.1.1f shows the final allocation of points from this BO loop, confirming that evaluation resources have been spent effectively.

BO's ability to find good solutions for "black-box" optimisation problems within heavily restricted evaluation budgets has lead to its use across a broad range of settings. One particularly popular application of BO is for tuning machine learning hyper-parameters (Swersky et al., 2013), for example in computer vision (Bergstra et al., 2013), natural language processing (Wang et al., 2015), text-to-speech (Moss et al., 2020a) and reinforcement learning (Chen et al., 2018b). Moreover, BO has also been used to solve optimisation problems from fields as widespread as gene design (González et al., 2014; Tanaka and Iwata, 2018; Moss et al., 2020b), molecular search (Gómez-Bombarelli et al., 2018; Griffiths and Hernández-Lobato, 2020), simulation optimisation (Kleijnen, 2009), and the design of physical science experiments (Frazier and Wang, 2016).

We now introduce the key parts of the standard BO framework, before turning to more sophisticated extended frameworks that are the focus of this thesis.



(a) Objective function.

(b) First BO step.

(c) Second BO step.

(d) Third BO step.

(e) Fourth BO step.

(f) Final sate.

Figure 2.1.1: A demonstrative BO loop. We wish to efficiently minimise the single-dimensional Forrester function starting from an initialisation of four randomly selected objective function evaluations (the green points). Our probabilistic surrogate model provides predictions for the objective function across the whole search space, yielding a predictive mean (the dark blue curve) and variance (the light blue regions). Each BO step evaluates the objective function at the maxima (the dashed vertical red line) of the acquisition function (the red curve).

2.1.1 Gaussian Process Surrogate Models

Although many probabilistic models have been used as BO surrogates, including random forests (Hutter et al., 2011) and neural networks (Snoek et al., 2015), the most popular choice by far is the Gaussian Process (Rasmussen, 2004a, GP). GPs are popular for BO for two key reasons. Firstly, unlike other non-parametric probabilistic models, GPs are hard to over-fit to the small data-sets common in BO as they require the specification of only a handful of model parameters rather than the many thousands required for Bayesian deep learning approaches. Secondly, GPs provide well-calibrated uncertainty estimates which, through convenient analytical expressions, can be accessed at low-cost by our acquisition functions. Regardless of the exact choice of surrogate model, the key attribute required by BO is that the surrogate model provides an accessible Gaussian predictive distributions for g across the whole search space, i.e it supplies functions $\mu_n : \mathcal{X} \to \mathbb{R}$ and $\sigma_n(\mathbf{x}) : \mathcal{X} \to \mathbb{R}^+$ such that $g(\mathbf{x})|D_n \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x}))$. In addition, some acquisition functions (for example, the information-theoretic approaches presented in this thesis) also require the predictive co-variances between sets of objective function evaluations. We now show how such a posterior predictive distribution is provided by GPs.

Loosely speaking, GPs specify a prior over functions, with each sample draw from the GP corresponding to a particular function over the search space $\mathcal{X}$. The smoothness of these sample functions is controlled by a choice of kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ as chosen when specifying the GP. Now, after defining the $n \times n$ Gram matrix $\mathbf{K}_n = [k(\mathbf{x}_i, \mathbf{x}_j)]_{(\mathbf{x}_i, \mathbf{x}_j) \in D_n}$, as well as a vector of observations $\mathbf{y} = [y_i]_{i=1,..,n}$, we can write down the generative model assumed by our GP as

$$\begin{aligned} \mathbf{g} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K}_n) \\ \mathbf{y} &\sim \mathcal{N}(\mathbf{g}, \sigma^2 \mathbf{I}), \end{aligned} \tag{2.1.2}$$

where I is the identity matrix, $\mathbf{0}$ is a vector of zeros, and $\mathbf{g}$ denotes the true values of our objective function (before contamination by observation noise). Observation noise is typically modelled to be Gaussian with homoscedastic variance σ^2. Although alternative noise models have been considered, for example hetroscedastic (Kersting

et al., 2007) or Student-t (Vanhatalo et al., 2009), these alternatives add substantially to the cost of fitting the GP and typically require more data than available in BO applications. In the GP formulation (2.1.2), we have assumed a zero mean for the latent process $g(\mathbf{x})$, however, this can be replaced with an appropriate deterministic function or statistical model (Rasmussen, 2004a) when additional knowledge about the objective function's structure is available.

Crucially for BO, the posterior predictive distribution of a GP is Gaussian with a closed-form mean and variance. In particular, standard conditioning and marginalisation properties of Gaussian distributions (see Rasmussen (2004a)) provide a posterior predictive distribution for the objective function at a new location $\mathbf{x}^*$ as $g(\mathbf{x}^*)|D_n \sim \mathcal{N}(\mu_n(\mathbf{x}^*), \sigma_n^2(\mathbf{x}^*))$, where

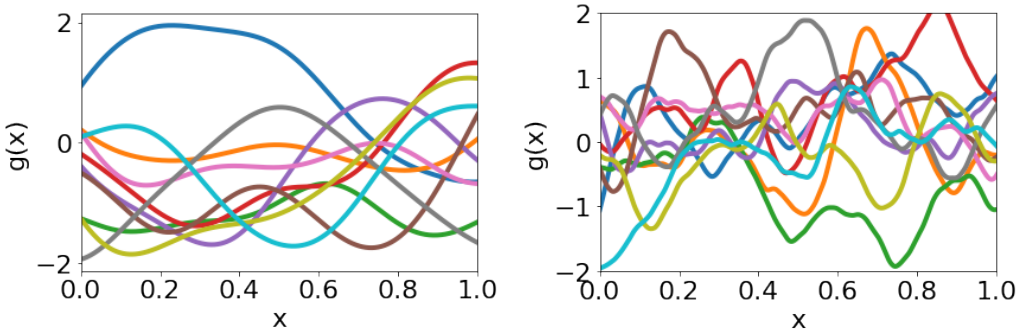
$$\begin{aligned}\mu_n(\mathbf{x}^*) =& \mathbf{k}_n(\mathbf{x}^*)^T(\mathbf{K}_n + \sigma^2 I)^{-1}\mathbf{y}_n \\ \sigma_n^2(\mathbf{x}^*) =& k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_n(\mathbf{x}^*)^T(\mathbf{K}_n + \sigma^2 I)^{-1}\mathbf{k}_n(\mathbf{x}^*)\end{aligned}$$

for $\mathbf{k}_n(\mathbf{x}^*) = [k(\mathbf{x}_i, \mathbf{x}^*)]_{\mathbf{x}_i \in D_n}$. The inversion of the $n \times n$ matrix $\mathbf{K}_n + \sigma^2\mathbf{I}$ is the major contribution to the cost of fitting a GP and requires an $O(n^3)$ computation. However, after this one-off cost, accessing the Gaussian posterior predictive distribution requires only a cheaper $O(n^2)$ calculation.

In the generative model (2.1.2) we see that the choice of kernel function specifies the assumed co-variance structure of the GP. The most popular choices of kernels for BO are the Radial Basis Function (RBF) kernel and Matérn-$\frac{5}{2}$ kernels which measure the similarity between two inputs $\mathbf{x}$ and $\mathbf{x}'$ as

$$\begin{aligned}k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') =& \alpha \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\ell^2}\right) \\ k_{Mat}(\mathbf{x}, \mathbf{x}') =& \alpha\left(1 + \frac{\sqrt{5}||\mathbf{x} - \mathbf{x}'||}{\ell} + \frac{5||\mathbf{x} - \mathbf{x}'||^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}||\mathbf{x} - \mathbf{x}'||}{\ell}\right).\end{aligned}$$

The choice of kernel function imposes strong priors on the type of functions that can be modelled by the GP. For example, the RBF kernel produces functions that are infinitely differentiable, whereas the Matérn-$\frac{5}{2}$'s sample paths are only twice differentiable (see Figure 2.1.2). Both theses kernels have two parameters (α and a



(a) RBF kernel.

(b) Matérn-$\frac{5}{2}$ kernel.

Figure 2.1.2: 10 sample functions drawn from GPs with RBF and Matérn-$\frac{5}{2}$ kernels.

length-scale ℓ) which, as well the observation noise variance σ^2, must be estimated from the data as they control the scale and variability of the function draws. These parameters are typically either chosen to maximise the GP's marginal likelihood or are sampled as part of a fully Bayesian treatment where the kernel parameters are allocated their own priors, the later being more expensive but sometimes yielding more stable models when data is scarce (Snoek et al., 2012). For search spaces with multiple dimensions, it is common to learn a length-scale for each dimension, allowing the learning of the relative importance of each dimension — a process known as automatic relevance determination (Mackay., 1995).

2.1.2 Acquisition functions

Acquisition functions use the predictive distribution of our surrogate model to predict the utility of making a new evaluation. Various heuristic strategies have been developed to form BO acquisition functions, including Probability of Improvement (Jones et al., 1998, PI), Expected Improvement (Jones et al., 1998, EI), Knowledge Gradient (Frazier et al., 2008, KG), and Upper-Confidence Bound (Srinivas et al., 2009, UCB). More recently, a new class of acquisition functions has been proposed based on information theory, including Entropy Search (Hennig and Schuler, 2012, ES), Predictive Entropy Search (Hernández-Lobato et al., 2014, PES) and Max-value Entropy Search (Wang and Jegelka, 2017, MES). Figure 2.1.3 presents a range of acquisition functions when

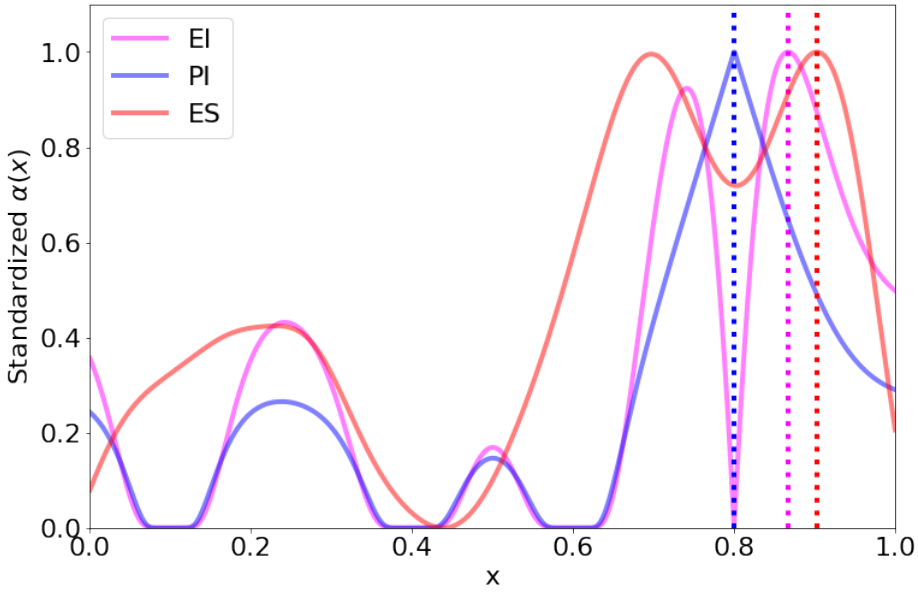


Figure 2.1.3: Different acquisition functions recommend evaluating different points (as denoted by vertical lines). For clarity, all acquisition functions are standardised to lie in $[0, 1]$.

used to perform the first BO step of Figure 2.1.1.

An important practical consideration for acquisition functions is that the efficacy of BO depends crucially on our ability to quickly and cheaply optimise our acquisition function across the search space. This maximisation sub-task, as required for each individual BO step, is henceforth referred to as the inner-loop maximisation. Clearly this inner-loop must incur an order-of-magnitudes lower computational cost than the maximisation of the original objective function for BO to be a feasible optimisation strategy. With this cost in mind, acquisition functions are typically defined to be cheap to query and, when considering continuous search spaces, to have accessible gradients for permitting gradient-based inner-loop maximisation. Of the three acquisition functions presented in Figure 2.1.3, the inner-loop optimisation took less than 0.1 seconds for EI and PI, but over 15 seconds for ES (a difference that grows as we increase the search space dimensions). Although information-theoretic search strategies often yield highly efficient BO (in terms of the number of BO steps required to find high-performing solutions), the substantial computational overhead limits their application to BO tasks

with search spaces of low dimensions, or with very large objective function query costs that can absorb a significant BO overhead. Reducing the cost and improving the applicability of information-theoretic acquisition functions is the primary focus of this thesis.

We now investigate the simple PI and EI acquisition functions to gain intuition about the properties desirable for acquisition functions, before diving deeper into information-theoretic search strategies.

Probability of Improvement (Jones et al., 1998)

The simplest BO acquisition function is PI, where we seek to evaluate g at the location that most likely to yield an improvement over the current best observed value $y_n^* = \mathbf{y}_n$, i.e. we measure the utility of an evaluation via the utility function

$$u^{\mathrm{PI}}(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad g(\mathbf{x}) \geq y_n^*, \\ 0 & \text{otherwise} \end{cases} .$$

Of course, we do not yet know the value of $g(\mathbf{x})$ which must be estimated using our surrogate model. Therefore, the PI acquisition function is defined as the expected utility

$$\begin{aligned} \alpha_n^{\mathrm{PI}}(\mathbf{x}) =& \mathbb{E}\left[u^{\mathrm{PI}}(\mathbf{x}) | D_n\right] \\ =& \mathbb{P}\left(g(\mathbf{x}) \geq y_n^* | D_n\right) \\ =& \Phi\left(-\gamma_{y_n^*}(\mathbf{x})\right), \end{aligned}$$

where $\gamma_y(\mathbf{x}) = \frac{y - \mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}$ and Φ is the Gaussian cumulative density function.

Expected Improvement (Jones et al., 1998)

Although simple to implement, PI can lead to ineffective BO as it is often a very greedy search strategy, repeatedly querying points very close together rather than fully exploring the search space (see Figure 2.1.3). To combat this, we turn to the EI acquisition function, which, rather than caring only if there will be an improvement,

considers the size of potential improvement through the utility function

$$u^{\mathrm{EI}}(\mathbf{x}) = \begin{cases} g(\mathbf{x}) - y_n^* & \text{if} \quad g(\mathbf{x}) \geq y_n^*, \\ 0 & \text{otherwise} \end{cases} .$$

The EI acquisition function is then given by the expected utility

$$\begin{aligned} \alpha_n^{\mathrm{EI}}(\mathbf{x}) =& \mathbb{E}\left[u^{\mathrm{EI}}(\mathbf{x})|D_n\right] \\ =& \left(\mu_n(\mathbf{x}) - y_n^*\right) \Phi(-\gamma_{y_n^*}(\mathbf{x})) + \sigma_n(\mathbf{x})\phi(\gamma_{y_n^*}(\mathbf{x})), \end{aligned} \tag{2.1.3}$$

where ϕ is the Gaussian probability density function.

The analytical form of EI (2.1.3) yields an intuitive decomposition that helps explain why EI can often provide effective BO. In particular, its first term grows with $\mu_n(\mathbf{x})$ to encourage evaluating locations that we believe have large objective function values (known as an exploitation strategy) and its second term grows with $\sigma_n(\mathbf{x})$ to encourage evaluating regions of the objective function for which we have high uncertainty (known as exploration).

2.1.3 Information-theoretic Acquisition Functions

Information-theoretic BO chooses to make its evaluations with the sole aim of reducing global uncertainty in the location of high-performing areas of the search space. All information-theoretic acquisition functions are built on the same core idea: measuring the utility of a potential evaluation as the reduction in uncertainty it yields about a particular quantity of interest. In information-theory, we measure the uncertainty of a random variable $\mathbf{A}$ through its differential entropy H (see Cover and Thomas, 2012, for an introduction to information theory), as given by $H(\mathbf{A}) = -\mathbb{E}\left[\log p_{\mathbf{A}}(\mathbf{a})\right]$. The expected reduction in differential entropy provided by evaluating another related random variable $\mathbf{B}$ is denoted as the mutual information $MI(\mathbf{A};\mathbf{B}) = H(\mathbf{A}) - H(\mathbf{A}|\mathbf{B})$. Information-theoretic search can then be defined as seeking those evaluations that provide the maximal mutual information, with differing information-theoretic search strategies distinguished by the choice of quantity of interest and the employed approximation methods.

Input-space Entropy Search (Hennig and Schuler, 2012)

One particularly intuitive search strategy is to choose evaluations that maximally reduce uncertainty in the input space location of the maxima $\mathbf{x}^* = \text{argmax}_{\mathbf{x}\in\mathcal{X}}\, g(\mathbf{x})$, a random variable with a distribution induced by the surrogate GP. The resulting acquisition function is known as Entropy Search (Hennig and Schuler, 2012, ES) and measures the utility of an observation as

$$\alpha_n^{ES}(\mathbf{x}) = MI(\mathbf{x}^*; y_{\mathbf{x}}|D_n) = H(\mathbf{x}^*|D_n) - \mathbb{E}_{y_{\mathbf{x}}}\left[H(\mathbf{x}^*|y_{\mathbf{x}}, D_n)|D_n\right], \tag{2.1.4}$$

where $y_{\mathbf{x}}$ denotes the yet-unobserved (and potentially noisy) evaluations of the objective function g at $\mathbf{x}$, as predicted by our GP surrogate model once conditioned on the previous evaluations D_n.

By exploiting the symmetric property of mutual information (i.e. $MI(\mathbf{A};\mathbf{B}) = MI(\mathbf{B};\mathbf{A})$), the ES acquisition function can be equivalently rewritten as

$$\alpha^{PES}(\mathbf{x}) = MI(y; \mathbf{x}^*|D_n) = H(y_{\mathbf{x}}|D_n) - \mathbb{E}_{\mathbf{x}^*}\left[H(y_{\mathbf{x}}|\mathbf{x}^*, D_n)|D_n\right], \tag{2.1.5}$$

yielding the Predictive Entropy Search (Hernández-Lobato et al., 2014, PES) acquisition function. Crucially, the first term of PES is simply the entropy of a multi-variate Gaussian distribution with a convenient closed-form expression, yielding a practical implementation advantage over ES.

Unfortunately, neither ES or PES can be computed analytically, primarily due to lack of closed form expression for the d-dimensional random variables $\mathbf{x}^*$ and $\mathbf{x}^*|y$ in (2.1.4) and $\mathbf{x}^*$ present in (2.1.5). Therefore, these information-theoretic acquisition functions incur significant computational overheads through expensive and complicated sampling-based approximations of the differential entropy of these d-dimensional quantities.

Output-space Entropy Search (Wang and Jegelka, 2017)

In order to provide computationally light-weight information-theoretic acquisition functions, search strategies that seek to reduce output uncertainty rather than input

uncertainty have become popular. Unlike ES, PES, which seek to reduce the uncertainty in the d-dimensional quantity $\mathbf{x}^*$, output-space entropy search seeks to reduce uncertainty in the single dimensional maximum value $g^* = \text{argmax}_{\mathbf{x}\in\mathcal{X}}\, g(\mathbf{x})$. Although still without closed-form expressions, g^* is a single dimensional quantity regardless of the dimensions of the objective function, and so is significantly easier to approximate than the d-dimensional $\mathbf{x}^*$ whilst still providing a meaningful search strategy (Wang et al., 2016).

The Max-value Entropy Search (MES) acquisition function of Wang and Jegelka (2017), with similar formulations considered by Hoffman and Ghahramani (2015) and Ru et al. (2018), was the first popular output-space acquisition function and can be formally expressed as

$$\alpha^{MES}(\mathbf{x}) = MI(y; g^*|D_n) = H(y_\mathbf{x}|D_n) - \mathbb{E}_{g^*}\left[H(y_\mathbf{x}|g^*, D_n)|D_n\right]. \tag{2.1.6}$$

As well as inheriting the analytic first term of PES, Wang and Jegelka (2017) note that, for problems with exact objective function evaluations, $y_\mathbf{x}|g^*$ is equivalent to $y_\mathbf{x}|y_\mathbf{x} < g^*$, i.e a truncated Gaussian distribution which has a closed-form expression for its differential entropy. Moreover an efficient sampling strategy for g^* is proposed, allowing a purely analytical expression for (2.1.6) through a Monte-Carlo approximation over this sampled set of maximum values $\mathcal{M}$, i.e

$$\alpha_n^{MES}(\mathbf{x}) \approx \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}}\left[\frac{\gamma_m(\mathbf{x})\phi\left(\gamma_m(\mathbf{x})\right)}{2\Phi\left(\gamma_m(\mathbf{x})\right)} - \log\Phi\left(\gamma_m(\mathbf{x})\right)\right],$$

where $\gamma_m(\mathbf{x}) = \frac{m-\mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}$, and ϕ and Φ are the standard Gaussian probability density and cumulative density functions. Therefore, after sampling a small collection of maximum values (a process required only once per BO step), MES has a closed form expression with accessible gradients that permits efficient inner-loop maximisation.

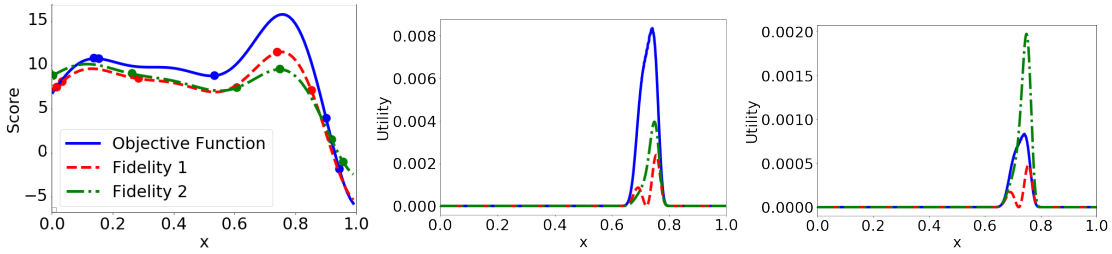
2.2 Extensions

BO has recently been extended to support a broader class of common high-cost optimisation tasks, including multi-fidelity, batch, constrained and multi-objective BO,

as-well as for discrete and highly-structured input spaces. Multi-fidelity BO leverages cheap approximations of the objective function to speed up optimisation, for example through exploiting coarse resolution simulations when calibrating large climate models (Prieß et al., 2011) or designing photonic nanostructures (Song et al., 2018). Batch BO considers scenarios where parallel computing resources can be exploited to allow multiple objective functions to be queried during each individual BO step, a scenario arising in science applications where multiple experiments can be ran concurrently, for example when training a collection of robots to cook (Junge et al., 2020). In contrast, multi-objective BO tackles problems that require the simultaneous maximisation of K separate objective functions, examples including building a chemical reactor that is both efficient and reliable (Park et al., 2018) or designing aerodynamic structures that perform well across multiple atmospheric conditions (Zuhal et al., 2018). Finally, constrained BO considers problems where certain areas of the search space cannot be queried, for example when designing aerofoils that follow certain shape constraints (Chaitanya and Vellanki, 2020) or designed molecules that follow synthesis-ability constraints (Griffiths and Hernández-Lobato, 2020). We now delve a little deeper into the BO extensions considered in this thesis.

2.2.1 Multi-fidelity Bayesian Optimisation

Multi-fidelity BO (also described as multi-task BO by Swersky et al. (2013)) considers problems where instead of querying the objective function g directly, we can alternatively query a (possibly infinite) collection functions somehow related to g (henceforth referred to as our fidelity space $\mathcal{F}$). If these alternative functions, as indexed by $\mathbf{s} \in \mathcal{F}$, are cheaper to evaluate and we can learn their relationship with the true objective function, then we can access cheap information sources that can be used to efficiently maximise g. Common low-fidelity estimates are those providing biased or noisy estimates of the true objective function and are typically modelled with multi-fidelity GPs (Kennedy and O'Hagan, 2000; Le Gratiet and Garnier, 2014; Klein et al., 2017a; Perdikaris et al., 2017; Cutajar et al., 2019). A popular application of Multi-fidelity BO is in hyper-parameter tuning, where the reliability (in terms of



(a) Collected observations. (b) Entropy reduction. (c) Entropy reduction per unit cost.

Figure 2.2.1: Maximising the negative Forrester function with access to two low-fidelity approximations at $\frac{1}{2}$ (red) and $\frac{1}{5}$ (green) the cost of querying the true objective. Although we learn the most from querying the objective function directly (blue), we can learn more per unit cost by querying the roughest fidelity. This figure is adapted from Moss et al. (2020d).

bias and noise) of each hyper-parameter evaluation can be dynamically controlled by choosing the proportion of data used when training models. Successful frameworks include those of Lam et al. (2015); Klein et al. (2017a); Kandasamy et al. (2016) and Kandasamy et al. (2017), all of which can reduce the computational cost of tuning complicated models by orders of magnitude over standard BO.

In practical terms, each step of multi-fidelity BO needs to choose a location-fidelity pair $\mathbf{z} = (\mathbf{x}, \mathbf{s}) \in \mathcal{Z} = \mathcal{X} \times \mathcal{F}$ upon which to collect the (possibly noisy) next evaluation $y_{\mathbf{z}} = f(\mathbf{z}) + \epsilon_{\mathbf{z}}$, where $f(\mathbf{z})$ is the result of querying parameter $\mathbf{x}$ on fidelity $\mathbf{s}$. To provide resource-efficient optimisation, we must balance how much we expect to learn about g^* with the computational cost of the evaluation (see Figure 2.2.1). Therefore, it is common to use a cost-weighted acquisition function (Swersky et al., 2013; Klein et al., 2017a; McLeod et al., 2017; Zhang et al., 2017), with the next evaluation chosen to satisfy

$$\mathbf{z}_{n+1} = \operatorname*{argmax}_{\mathbf{z} \in \mathcal{Z}} \frac{\alpha_n(\mathbf{z})}{c(\mathbf{z})},$$

where $c : \mathcal{Z} \to \mathbb{R}^+$ measures the cost of evaluating location $\mathbf{x}$ on fidelity $\mathbf{f}$. This cost function could be known *a priori* or estimated from observed costs following Snoek et al. (2012). Many of the standard acquisition functions discussed earlier have been

extended to multi-fidelity BO, for example there exist variants of KG (Poloczek et al., 2017; Wu et al., 2019), EI (Swersky et al., 2013; Picheny et al., 2013; Lam et al., 2015), UCB (Kandasamy et al., 2016, 2017), ES (Swersky et al., 2013) and PES (Zhang et al., 2017).

2.2.2 Batch Bayesian Optimisation

Two distinct scenarios have been considered for batch BO. Firstly, synchronous batch BO considers problems where B objective function evaluations can be queried in parallel, yielding their results at the same time. In contrast, asynchronous batch BO controls a collection of B independent workers that can query the objective function and return evaluations separately. The primary practical distinction (as summarised in Figure 2.2.2) is that, while synchronous batch acquisition functions must be able to measure the utility of jointly evaluating B locations, asynchronous batch BO has to measure the utility of a making a further single evaluation whilst taking into account the utility likely to be provided by the $B - 1$ pending evaluations. Asynchronous batch BO is useful for scenarios where function queries take varying amounts of time, for example when performing multi-fidelity optimisation with access to fast low-fidelity approximations. However, many problems in BO require full synchronous batch support. Application of synchronous batch BO include all problems where objective function query times are equal, or even multi-fidelity optimisation tasks where individual workers do not have sufficient autonomy to be controlled separately.

Much like in multi-fidelity BO, the popular acqusition functions for standard BO have also been extended to perform batch optimisation, for example EI (Chevalier and Ginsbourger, 2013; Marmin et al., 2015), UCB (Contal et al., 2013), KG (Wu and Frazier, 2016), PES (Shah and Ghahramani, 2015). In addition, heuristics for designing batches have been proposed that can extend any acquisition function to support batches, the most popular and empirically successful being the Local Penalisation of González et al. (2016a) and the DPP-based approach of Kathuria et al. (2016). Other approaches based on Stein methods (Gong et al., 2019) and Thompson sampling (Kandasamy et al., 2018a) have also been proposed.

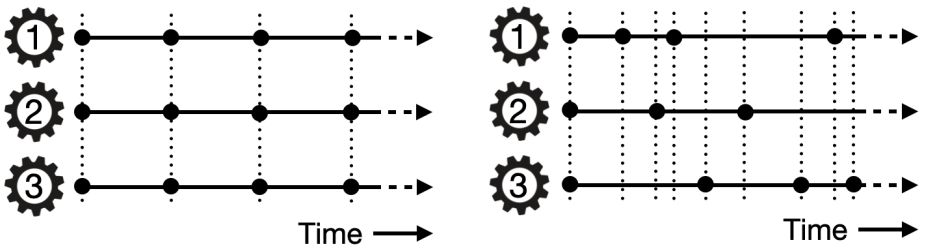


Figure 2.2.2: Synchronous (left panel) and asynchronous (right panel) batch BO under the capacity for $B = 3$ workers. Dots denote the return of an evaluation to the optimiser and the subsequent reallocation of workers is denoted with a vertical dotted line. Synchronous BO jointly allocates batches of B evaluations, whereas asynchronous BO must allocate workers individually whilst taking into account the $B - 1$ pending evaluations.

2.2.3 Bayesian Optimisation for Structured Search Spaces

Until very recently, the vast majority of BO approaches were designed for low dimensional and mostly continuous search spaces. However, BO for structured optimisation tasks is a fast growing frontier of the BO literature, with recent work applying BO to search spaces consisting of strings (Moss et al., 2020b; Swersky et al., 2020), combinatorial structures (Deshwal et al., 2020) and neural network architectures (Kandasamy et al., 2018b).

Objective functions in structured design tasks often satisfy a notion of smoothness, with small perturbations in their structure leading to only small changes in the objective function value. This smoothness can, in theory, be exploited by BO arguments to provide efficient optimisation. However, in practice, two practical considerations prevent the use of standard BO methodology to string optimisation. Firstly, standard GP models do not support discrete and often variable-length structures as their inputs, with their kernels requiring fixed-length continuous search spaces and incurring the curse of dimensionality when used to model high-dimensional spaces (Györfi et al., 2006). Secondly, devising a BO framework directly over discrete structures raises the question of how to maximise acquisition functions. Standard BO over Euclidean spaces uses standard numerical methods to maximise these functions, for example gradient

and local-search methods. However, these maximisers are not applicable when the inputs are discrete structures. Moreover, structured search spaces are often heavily constrained by complex underlying rules that determine the validity of structures they contain, making inner-loop maximisation even more challenging.

Consequently, many existing applications of BO to structured optimisation rely on projecting discrete structures into continuous and unconstrained latent spaces of low and fixed dimension, in which routine BO techniques can be applied. This projection approach has provided BO routines for designing molecules (Gómez-Bombarelli et al., 2018; Kusner et al., 2017), molecular graphs (Kajino, 2019), and networks (Zhang et al., 2019). As-well as adding (sometimes significantly) to the computational overheads of BO, learning projections that provide meaningful latent representations for the whole search space can can be very difficult in the low-data scenarios typical of BO, resulting in routines that exploring only limited regions of the search space.

More recently, BO approaches have been developed that operate directly on raw discrete structures through building custom Gaussian process kernels and employing sophisticated discrete optimisers for inner-loop maximisation. Although often providing more effective exploration of the search space, these direct approaches incur high inner-loop maximisation costs, for example when using evolutionary algorithms for BO-based neural network architecture design (Kandasamy et al., 2018b) or mathematical programming for combinatorial BO (Deshwal et al., 2020).

Chapter 3

MUMBO: MUlti-task Max-value Bayesian Optimisation

Status: Published as Moss H. B., Leslie D. S. & Rayson P., MUMBO: MUlti-task Max-value Bayesian Optimisation, *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020.

3.1 Preface

In this chapter we propose MUMBO, the first high-performing yet computationally efficient acquisition function for multi-task Bayesian optimisation. Here, the challenge is to perform efficient optimisation by evaluating low-cost functions somehow related to our true target function. This is a broad class of problems including the popular task of multi-fidelity optimisation. However, while information-theoretic acquisition functions are known to provide state-of-the-art Bayesian optimisation, existing implementations for multi-task scenarios have prohibitive computational requirements. Previous acquisition functions have therefore been suitable only for problems with both low-dimensional parameter spaces and function query costs sufficiently large to overshadow very significant optimisation overheads. In this chapter, we derive a novel multi-task version of entropy search, delivering robust performance with low computational overheads across classic optimisation challenges and multi-task hyper-parameter

tuning. MUMBO is scalable and efficient, allowing multi-task Bayesian optimisation to be deployed in problems with rich parameter and fidelity spaces.

3.2 Introduction

The need to efficiently optimise functions is ubiquitous across machine learning, operational research and computer science. Many such problems have special structures that can be exploited for efficient optimisation, for example gradient-based methods on cheap-to-evaluate convex functions, and mathematical programming for heavily constrained problems. However, many optimisation problems do not have such clear properties.

Bayesian Optimisation (BO) is a general method to efficiently optimise 'black-box' functions for which we have weak prior knowledge, typically characterised by expensive and noisy function evaluations, a lack of gradient information, and high levels of non-convexity (see Shahriari et al. (2016) for a comprehensive review). By sequentially deciding where to make each evaluation as the optimisation progresses, BO is able to direct resources into promising areas and so efficiently explore the search space. In particular, a highly effective and intuitive search is achieved through **information-theoretic** BO, where we seek to sequentially reduce our uncertainty (measured in terms of differential entropy) in the location of the optima with each successive function evaluation (Hennig and Schuler, 2012; Hernández-Lobato et al., 2014).

For optimisation problems where we can evaluate low-cost functions somehow related to our true objective function, **Multi-Task** (MT) BO (as first introduced by Swersky et al. (2013)) provides additional efficiency gains. A popular subclass of MT BO problems is **Multi-Fidelity** (MF) BO, where the set of related functions can be meaningfully ordered by their similarity to the objective function. Unfortunately, performing BO over MT spaces has previously required complicated approximation schemes that scale poorly with dimension (Swersky et al., 2013; Zhang et al., 2017), limiting the applicability of information-theoretic arguments to problems with both low-

dimensional parameter spaces and function query costs sufficiently large to overshadow very significant optimisation overheads. Therefore, MT BO has so far been restricted to considering simple structures at a large computational cost. Despite this restriction, MT optimisation has wide-spread use across physical experiments (Nguyen et al., 2013; Zheng et al., 2013; Pilania et al., 2017), environmental modelling (Prieß et al., 2011), and operational research (Huang et al., 2006; Xu et al., 2016; Yong et al., 2019).

For expositional simplicity, this chapter focuses primarily on examples inspired by tuning the hyper-parameters of machine learning models. Such problems have large environmental impact (Strubell et al., 2019), requiring multiple days of computation to collect even a single (often highly noisy) performance estimate. Consequently, these problems have been proven a popular and empirically successful application of BO (Snoek et al., 2012). MF applications for hyper-parameter tuning dynamically control the reliability (in terms of bias and noise) of each hyper-parameter evaluation (Kennedy and O'Hagan, 2000; Lam et al., 2015; Klein et al., 2017a; Kandasamy et al., 2016, 2017) and can reduce the computational cost of tuning complicated models by orders of magnitude over standard BO. Orthogonal savings arise from considering hyper-parameter tuning in another MT framework; FASTCV (Swersky et al., 2013) recasts tuning by K-fold cross-validation (CV) (Kohavi, 1995) into the task of simultaneously optimising the K different evaluations making a single K-fold CV estimate.

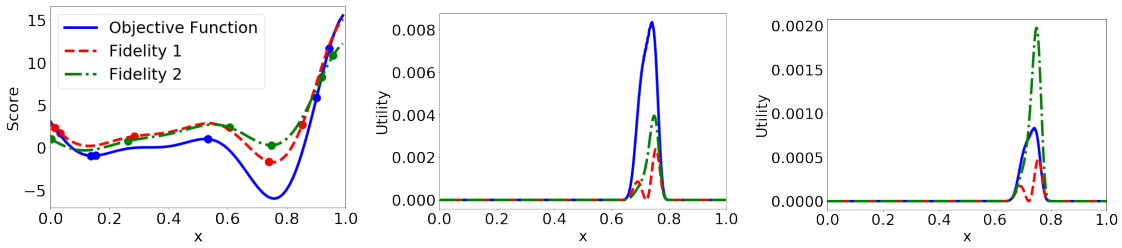
Information-theoretic arguments are particularly well suited to such MT problems as they provide a clear measure of the utility (the information gained) of making an evaluation on a particular sub-task. This utility then can be balanced with computational cost, providing a single principled decision (Swersky et al., 2013; Klein et al., 2017a; McLeod et al., 2017; Zhang et al., 2017). Despite MT BO being a large sub-field in its own right, there exist only a few alternatives to information-theoretic acquisition functions. Alternative search strategies include extensions of standard BO acquisition functions, including knowledge gradient (KG) (Poloczek et al., 2017; Wu et al., 2019), expected improvement (EI) (Swersky et al., 2013; Picheny et al., 2013; Lam et al., 2015), and upper-confidence bound (UCB) (Kandasamy et al., 2016, 2017). KG achieves efficient optimisation but incurs a high computational overhead. The MT

extensions of EI and UCB, although computationally cheap, lack a clear notion of utility and consequently rely on two-stage heuristics, where a hyper-parameter followed by a task are chosen as two separate decisions. Moreover, unlike our proposed work, the performance of MT variants of UCB and EI depends sensitively on problem-specific parameters which require careful tuning, often leading to poor performance in practical tasks. Information-theoretic arguments have produced the MF BO hyper-parameter tuner FABOLAS (Klein et al., 2017a), out-competing approaches across richer fidelity spaces based on less-principled acquisitions (Kandasamy et al., 2017). This success motivates our work to provide scalable entropy reduction over MT structures.

We propose **MUMBO**, a novel, scalable and computationally light implementation of information-theoretic BO for general MT frameworks. Inspired by the work of Wang and Jegelka (2017), we seek reductions in our uncertainty in the value of the objective function at its optima (a single-dimensional quantity) rather than our uncertainty in the location of the optima itself (a random variable with the same dimension as our search space). MUMBO enjoys three major advantages over current information-theoretic MT approaches:

- MUMBO has a simple and scalable formulation requiring routine one-dimensional approximate integration, irrespective of the search space dimensions,
- MUMBO is designed for general MT and MF BO problems across both continuous and discrete fidelity spaces,
- MUMBO outperforms current information-theoretic MT BO with a significantly reduced computational cost.

Parallel work (Takeno et al., 2019) presents essentially the same acquisition function but restricted to discrete multi-fidelity problems from the material sciences. Our chapter provides a different derivation and general presentation of the method which enables deployment with both discrete and continuous fidelity spaces in general MT BO (including MF).



(a) Collected Observations (b) Information gain (c) Gain per unit cost

Figure 3.3.1: Seeking the minimum of the 1D Forrester function (blue) with access to two low-fidelity approximations at $\frac{1}{2}$ (red) and $\frac{1}{5}$ (green) the cost of querying the true objective. Although we learn the most from directly querying the objective function, we can learn more per unit cost by querying the roughest fidelity.

3.3 Problem Statement and Background

We now formalise the goal of MT BO, introducing the notation used throughout this work. The goal of BO is to find the maximiser

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, g(\mathbf{x}) \tag{3.3.1}$$

of a function g over a d-dimensional set of feasible choices $\mathcal{X} \subset \mathbb{R}^d$ spending as little computation on function evaluations as possible.

Standard BO seeks to solve (3.3.1) by sequentially collecting noisy observations of g. By fitting a Gaussian process (GP) (Rasmussen, 2004a), a non-parametric model providing regression over all functions of a given smoothness (to be controlled by a choice of kernel k), we are able to quantify our current belief about which areas of the search space maximise our objective function. An acquisition function $\alpha_n(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ uses this belief to predict the utility of making any given evaluation, producing large values at 'reasonable' locations. A standard acquisition function (Hennig and Schuler, 2012) is the expected amount of information provided by each evaluation about the location of the maximum. Therefore after making n evaluations, BO will automatically next evaluate $\mathbf{x}_{n+1} = \operatorname{argmax}_{\mathbf{x}\in\mathcal{X}} \alpha_n(\mathbf{x})$.

3.3.1 Multi-task Bayesian Optimisation

Suppose that instead of querying g directly, we can alternatively query a (possibly infinite) collection of related functions indexed by $\mathbf{z} \in \mathcal{Z}$ (henceforth referred to as our fidelity space). We then collect the (noisy) observations $D_n = \{(\mathbf{x}_t, \mathbf{z}_t, y_t)\}$ for $y_t = f(\mathbf{x}_t, \mathbf{z}_t) + \epsilon_t$, where $f(\mathbf{x}, \mathbf{z})$ is the result of querying parameter $\mathbf{x}$ on fidelity $\mathbf{z}$, and ϵ_t is Gaussian noise. If these alternative functions f are cheaper to evaluate and we can learn their relationship with g, then we have access to cheap sources of information that can be used to help find the maximiser of the true task of interest.

3.3.2 Multi-task acquisition functions

The key difference between standard BO and MT BO is that our acquisition function must be able to not only choose the next location, but also which fidelity to evaluate, balancing computational cost with how much we expect to learn about the maximum of g. Therefore, we require an extended acquisition function $\alpha_n : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}$ and a cost function $c : \mathcal{X} \times \mathcal{Z} \to \mathbb{R}^+$, measuring the utility and cost of evaluating location $\mathbf{x}$ at fidelity $\mathbf{z}$ (as demonstrated in Figure 3.3.1c). In Section 3.5, we consider problems both where this cost function is known *a priori* and where it is unknown but estimated using an extra GP (Snoek et al., 2012). In this work, we seek to make the evaluation that provides the largest information gain per unit cost, i.e. maximising the ratio

$$(\mathbf{x}_{n+1}, \mathbf{z}_{n+1}) = \underset{(\mathbf{x},\mathbf{z})\in\mathcal{X}\times\mathcal{Z}}{\operatorname{argmax}} \frac{\alpha_n(\mathbf{x}, \mathbf{z})}{c(\mathbf{x}, \mathbf{z})}. \tag{3.3.2}$$

3.3.3 Multi-task models

To perform MT BO, our underlying Gaussian process model must be extended across the fidelity space. By defining a kernel over $\mathcal{X} \times \mathcal{Z}$, we can learn predictive distributions after n observations with means $\mu^n(\mathbf{x}, \mathbf{z})$ and co-variances $\Sigma^n((\mathbf{x}, \mathbf{z}), (\mathbf{x}', \mathbf{z}'))$ from which $\alpha_n(\mathbf{x}, \mathbf{z})$ can be calculated. Although increasing the dimension of the kernel for $\mathcal{X}$ to incorporate $\mathcal{Z}$ provides a very flexible model, it is argued by Kandasamy et al. (2017) that overly flexible models can harm optimisation speed by requiring too much learning, restricting the sharing of information across the fidelity space. Therefore, it

is common to use more restrictive separable kernels that better model specific aspects of the given problem.

A common kernel for discrete fidelity spaces is the intrinsic coregionalisation kernel of Bonilla et al. (2008) (as used in Figure 3.3.1). This kernel defines a co-variance between hyper-parameter and fidelity pairs of

$$k((\mathbf{x}, z), (\mathbf{x}', z')) = k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \times B(z, z'), \tag{3.3.3}$$

for a base kernel $k_{\mathcal{X}}$ and a positive semi-definite $|\mathcal{Z}| \times |\mathcal{Z}|$ matrix B (set by maximising the model likelihood alongside the other kernel parameters). B represents the correlation between different fidelities, allowing the sharing of information across the fidelity space. See Section 3.5 for additional standard MF kernels.

3.3.4 Information-theoretic MT BO

Existing methods for information-theoretic MT BO seek to maximally reduce our uncertainty in the location of the maximiser $\mathbf{x}^* = \text{argmax}_{\mathbf{x}\in\mathcal{X}}\, g(\mathbf{x})$. Following the work of Hennig and Schuler (2012), uncertainty in the value of $\mathbf{x}^*$ is measured as its differential entropy $H(\mathbf{x}^*) = -\mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}^*}}(\log p_{\mathbf{x}^*}(\mathbf{x}))$, where $p_{\mathbf{x}^*}$ is the probability density function of $\mathbf{x}^*$ according to our current GP model. For MT optimisation, we require knowledge of the amount of information provided about the location of $\mathbf{x}^*$ from making an evaluation at $\mathbf{x}$ on fidelity $\mathbf{z}$, measured as the mutual information

$$\text{MI}(y(\mathbf{x}, \mathbf{z}); \mathbf{x}^* | D_n) = H(\mathbf{x}^* | D_n) - \mathbb{E}_y\left[H(\mathbf{x}^* | y(\mathbf{x}, \mathbf{z}), D_n)\right]$$

between an evaluation $y(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}, \mathbf{z}) + \epsilon$ and $\mathbf{x}^*$, where the expectation is over $p(y(\mathbf{x}, \mathbf{z})|D_n)$ (see Cover and Thomas (2012) for an introduction to information theory).

Successively evaluating the parameter-fidelity pair that provides the largest information gain per unit of evaluation cost provides the entropy search acquisition function used by Swersky et al. (2013) and Klein et al. (2017a), henceforth referred to as the MTBO acquisition function. Unfortunately, the calculation of MTBO relies on sampling-based approximations to the non-analytic distribution of $\mathbf{x}^* \,|\, D_n$. Such approximations scale poorly in both cost and performance with the dimensions of our

search space (as demonstrated in Section 3.5). A modest computational saving can be made for standard BO problems by exploiting the symmetric property of mutual information, producing the predictive entropy search (PES) of Hernández-Lobato et al. (2014). However, PES still requires approximations of $\mathbf{x}^* \mid D_n$ and it is unclear how to extend this approach across MT frameworks.

3.4 MUMBO

In this work, we extend the computationally efficient information-theoretic acquisition function of Wang and Jegelka (2017) to MT BO. With their max-value entropy-search acquisition function (MES), they demonstrate that seeking to reduce our uncertainty in the value of $g^* = g(\mathbf{x}^*)$ provides an equally effective search strategy as directly minimising the uncertainty in the location $\mathbf{x}^*$, but with significantly reduced computation. Similarly, MUMBO seeks to compute the information gain

$$\begin{aligned}\alpha_n^{MUMBO}(\mathbf{x}, \mathbf{z}) =& \mathrm{MI}\left(y(\mathbf{x}, \mathbf{z}); g^* | D_n\right) \\ =& H(y(\mathbf{x}, \mathbf{z}) \mid D_n) - \mathbb{E}_{g^*}[H(y(\mathbf{x}, \mathbf{z}) \mid g^*, D_n)]\,, \end{aligned} \tag{3.4.1}$$

which can then be combined with the evaluation cost $c(\mathbf{x}, \mathbf{z})$ (following (3.3.2)). Here the expectation is over our current uncertainty in the value of $g^*|D_n$.

3.4.1 Calculation of MUMBO

Although extending MES to MT scenarios retains the intuitive formulation and the subsequent principled decision-making of the original MES, we require a novel non-trivial calculation method to maintain its computational efficiency for MT BO. We now propose a strategy for calculating the MUMBO acquisition function that requires the approximation of only single-dimensional integrals irrespective of the dimensions of our search space.

The calculation of our MUMBO acquisition function (3.4.1) for arbitrary $\mathbf{x}$ and $\mathbf{z}$ must be efficient as each iteration of BO requires a full maximisation of (3.4.1) over $\mathbf{x}$ and $\mathbf{z}$ (i.e 3.3.2). For ease of notation we drop the dependence on $\mathbf{x}$ and $\mathbf{z}$, so that

g denotes the target function value at $\mathbf{x}$, f denotes the evaluation of $\mathbf{x}$ at fidelity $\mathbf{z}$, and y denotes the (noisy) observed value of $f(\mathbf{x}, \mathbf{z})$. Since BO fits a Gaussian process to the underlying functions, our assumptions about g and y imply that their joint predictive distribution is a bivariate Gaussian; with expectation, variance and correlation derived from our GP (as shown in Appendix A.1) and denoted by (μ_g, μ_f), $(\sigma_g^2, \sigma_f^2 + \sigma^2)$ and ρ respectively. These values summarise our current uncertainty in g and f and how useful making an evaluation y will be for learning about g. Note that access to this simple two-dimensional predictive distribution is all that is needed to calculate MUMBO (3.4.1).

The first term of (3.4.1) is the differential entropy of a Gaussian distribution and so can be calculated analytically as $\frac{1}{2}\log(2\pi e(\sigma_f^2 + \sigma^2))$. The second term of (3.4.1) is an expectation over the maximum value of the true objective g^*, which can be approximated using a Monte Carlo approach; we use Wang and Jegelka (2017)'s method to approximately sample a set of N samples $G = \{g_1, \ldots, g_N\}$ from $g^* \,|\, D_n$, using a mean-field approximation and extreme value theory.

It remains to calculate the quantity inside the expectation for a given value of g^*. The equivalent quantity in the original MES (without fidelity considerations) was analytically tractable, but we show that for MUMBO this term is intractable. In particular, we show that $y \,|\, g < g^*$ follows an extended-skew Gaussian (ESG) distribution (Azzalini, 1985; Arnold et al., 1993) in Appendix A.1. Unfortunately, Arellano-Valle et al. (2013) have shown that there is no analytical form for the differential entropy of an ESG. Therefore, after manipulations presented also in Appendix A.1 and reintroducing dependence on $\mathbf{x}$ and $\mathbf{z}$, we re-express (3.4.1) as

$$\begin{aligned}\alpha_n^{MUMBO}(\mathbf{x}, \mathbf{z}) =& \frac{1}{N}\sum_{g^* \in G}\Bigg[\rho(\mathbf{x}, \mathbf{z})^2 \frac{\gamma_{g^*}(\mathbf{x})\phi(\gamma_{g^*}(\mathbf{x}))}{2\Phi(\gamma_{g^*}(\mathbf{x}))} - \log(\Phi(\gamma_{g^*}(\mathbf{x})))\\ &+ \mathbb{E}_{\theta \sim Z_{g^*}(\mathbf{x},\mathbf{z})}\left[\log\left(\Phi\Big\{\frac{\gamma_{g^*}(\mathbf{x}) - \rho(\mathbf{x}, \mathbf{z})\theta}{\sqrt{1 - \rho^2(\mathbf{x}, \mathbf{z})}}\Big\}\right)\right]\Bigg], \end{aligned} \tag{3.4.2}$$

where Φ and ϕ are the standard normal cumulative distribution and probability density functions, $\gamma_{g^*}(\mathbf{x}) = \frac{g^* - \mu_g(\mathbf{x})}{\sigma_g(\mathbf{x})}$ and $Z_{g^*}(\mathbf{x}, \mathbf{z})$ is an ESG (with probability density function provided in Appendix A.1).

Expression (3.4.2) is analytical except for the final term, which must be approximated for each of the N samples of g^* making up the Monte Carlo estimate. Crucially, this is just a single-dimensional integral of an analytic expression and, hence, can be quickly and accurately approximated using standard numerical integration techniques. We present MUMBO within a BO loop as Algorithm 1.

Algorithm 1 MUlti-fidelity and MUlti-task Max-value Bayesian Optimisation

function MUMBO(budget B, N samples of g^*)
  Initialise $n \leftarrow 0$, $b \leftarrow 0$
  Collect initial design D_0
  while $b < B$ **do**
    Begin new iteration $n \leftarrow n + 1$
    Fit GP to the collected observations D_{n-1}
    Simulate N samples of $g^*|D_{n-1}$
    Prepare $\alpha_{n-1}^{MUMBO}(\mathbf{x}, \mathbf{z})$ as given by Eq. (3.4.2)
    Find the next point and fidelity to query $(\mathbf{x}_n, \mathbf{z}_n) \leftarrow \text{argmax}_{(\mathbf{x},\mathbf{z})} \frac{\alpha_{n-1}^{MUMBO}(\mathbf{x},\mathbf{z})}{c(\mathbf{x},\mathbf{z})}$
    Collect the new evaluation $y_n \leftarrow f(\mathbf{x}_n, \mathbf{z}_n) + \epsilon_n$, $\epsilon_n \sim N(0, \sigma^2)$
    Append new evaluation to observation set $D_n \leftarrow D_{n-1} \bigcup \{(\mathbf{x}_n, \mathbf{z}_n), y_n\}$
    Update spent budget $b \leftarrow b + c(\mathbf{x}_n, \mathbf{z}_n)$
  return Believed optimum across $\{\mathbf{x}_1, .., \mathbf{x}_n\}$

3.4.2 Interpretation of MUMBO

We provide intuition for (3.4.2) by relating MUMBO to an established BO acquisition function. In the formulation of MUMBO (3.4.2), we see that for a fixed parameter choice $\mathbf{x}$ (and ignoring evaluation costs) this acquisition is maximised by choosing the fidelity z that provides the largest $|\rho(\mathbf{x}, \mathbf{z})|$, meaning that the stronger the correlation (either negatively or positively) the more we can learn about the true objective. In fact, if we find a fidelity $\mathbf{z}^*$ that provides evaluations that agree completely with g,

then we would have $\rho(\mathbf{x}, \mathbf{z}^*) = 1$ and (3.4.2) would collapse to

$$\alpha_n(\mathbf{x}, \mathbf{z}^*) = \frac{1}{N} \sum_{g^* \in G} \left[\frac{\gamma_{g^*}(\mathbf{x})\phi(\gamma_{g^*}(\mathbf{x}))}{2\Phi(\gamma_{g^*}(\mathbf{x}))} - \log(\Phi(\gamma_{g^*}(\mathbf{x}))) \right].$$

This is exactly the same expression presented by Wang and Jegelka (2017) in the original implementation of MES, appropriate for standard BO problems where we can only query the function we wish to optimise.

3.4.3 Computational Cost of MUMBO

The computational complexity of any BO routine is hard to measure exactly, due to the acquisition maximisation (3.3.2) required before each function query. However, the main contributor to computational costs are the resources required for each calculation of the acquisition function with respect to problem dimension d and the N samples of g^*. Each prediction from our GP model costs $O(d)$, and single-dimensional numerical integration over a fixed grid is $O(1)$. Therefore, a single evaluation of MUMBO can be regarded as an $O(Nd)$ operation. Moreover, as MUMBO relies on the approximation of a single-dimensional integral, we do not require an increase in N to maintain performance as the problem dimension d increases (as demonstrated in Section 3.5) and so MUMBO scales linearly with problem dimension. In contrast, the MT BO acquisition used by Swersky et al. (2013) and Klein et al. (2017a) for information-theoretic MT BO relies on sampling-based approximations of d-dimensional distributions, therefore requiring exponentially increasing sample sizes to maintain performance as dimension increases, rendering them unsuitable for even moderately-sized BO problems. In addition, we note that these current approaches require expensive sub-routines and the calculation of derivative information, making their computational cost for even small d much larger than that of MUMBO.

3.5 Experiments

We now demonstrate the performance of MUMBO across a range of MT scenarios, showing that MUMBO provides superior optimisation to all existing approaches, with

a significantly reduced computational overhead compared to current state-of-the-art. As is common in the optimisation literature, we first consider synthetic benchmark functions in a discrete MF setting. Next, we extend the challenging continuous MF hyper-parameter tuning framework of FABOLAS and use MUMBO to provide a novel information-theoretic implementation of the MT hyper-parameter tuning framework FASTCV, demonstrating that the performance of this simple MT model can be improved using our proposed fully-principled acquisition function. Finally, we use additional synthetic benchmarks to compare MUMBO against a wider range of existing MT BO acquisition functions.

Alongside the theoretical arguments of this paper, we also provide a software contribution to the BO community of a flexible implementation of MUMBO with support for Emukit (Paleyes et al., 2019). We use a DIRECT optimiser (Jones et al., 1993) for the acquisition maximisation at each BO step and calculate the single-dimensional integral in our acquisition (3.4.2) using Simpson's rule over appropriate ranges (from the known expressions of an ESG's mean and variance derived in Appendix A.1.1).

3.5.1 General Experimental Details

Overall, the purpose of our experiments is to demonstrate how MUMBO compares to other acquisition functions when plugged into a set of existing MT problems, focusing on providing a direct comparison with the existing state-of-the-art in information-theoretic MT BO used by Swersky et al. (2013) and Klein et al. (2017a) (which we name MTBO). Our main experiments also include the performance of popular low-cost MT acquisition functions MF-GP-UCB (Kandasamy et al., 2016) and MT expected improvement (Swersky et al., 2013). In Section 3.5.5 we expand our comparison to include a wider range of existing BO routines, chosen to reflect popularity and code availability. We include the MF knowledge gradient (MISO-KG)(Poloczek et al., 2017)[1], an acquisition function with significantly larger computational overheads than MUMBO (and MTBO), as-well as the low-cost acquisition functions of BOCA (Kandasamy

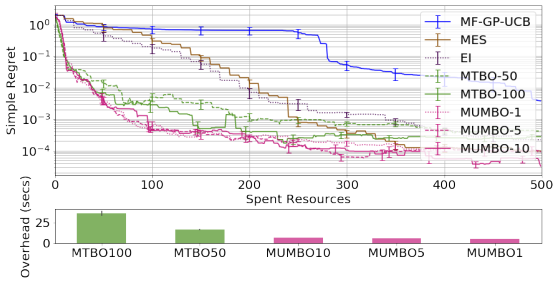
[1]As implemented by the original authors at *https://github.com/misokg/NIPS2017*

et al., 2017) and MF-SKO (Huang et al., 2006). Due to a lack of provided code, and the complexity of their proposed implementations, we were unable to implement multi-fidelity extensions of PES (McLeod et al., 2017; Zhang et al., 2017) or the variant of knowledge-gradient for continuous fidelity spaces (Wu et al., 2019). As both PES and knowledge gradient require approximations of quantities with dimensionality equal to the search space, their MT extensions will suffer the same scalability issue as MTBO (and MISO-KG). Finally, to demonstrate the benefit of considering MT frameworks, we also present the standard BO approaches of expected improvement (EI) and max-value entropy search (MES) which query only the true objective.

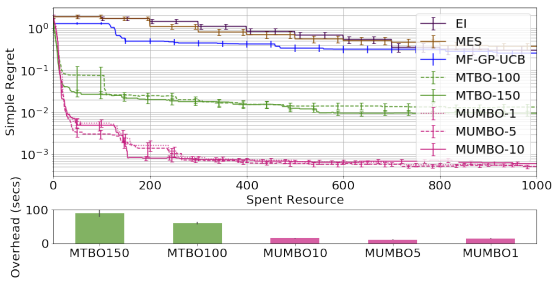
To test the robustness of the information-theoretic acquisitions we vary the number of Monte Carlo samples N used for both MUMBO and MTBO (denoted as MUMBO-N and MTBO-N). We report both the time taken to choose the next location to query (referred to as the optimisation overhead) and the performance of the believed objective function optimiser (the incumbent) as the optimisation progresses. For our synthetic examples, we measure performance after n evaluations as the simple regret $R_n = g(\mathbf{x}^*) - g(\hat{\mathbf{x}}_n)$, representing the sub-optimality of the current incumbent $\hat{\mathbf{x}}_n$. Experiments reporting wall-clock timings were performed on single core Intel Xeon 2.30GHz processors. Detailed implementation details are provided in Appendix A.2.

3.5.2 Discrete Multi-fidelity BO

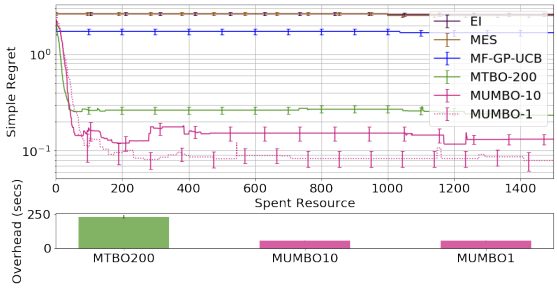
First, we consider the optimisation of synthetic problems, using the intrinsic coregionalization kernel introduced earlier (3.3.3). Figure 3.5.1 demonstrates the superior performance and light computational overhead of MUMBO across these test functions when we have access to continuous or discrete collections of cheap low-fidelity functions at lower query costs. Although MTBO and MUMBO initially provide comparable levels of optimisation, MUMBO quickly provides optimisation with substantially higher precision than MTBO and MF-GP-UCB. We delve deeper into the low performance of MF-GP-UCB in Appendix A.2.1. In addition, MUMBO is able to provide high-precision optimisation even when based on a single sample of g^*, whereas MTBO requires 50 samples for reasonable performance on the 2D optimisation task, struggles



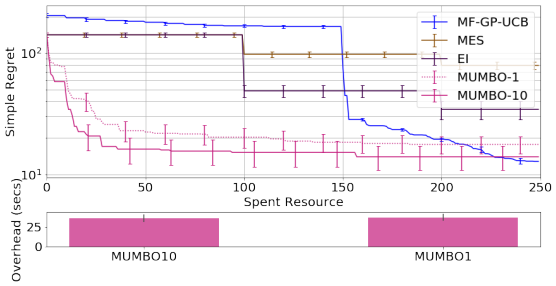
(a) Maximisation of the 2D Currin function (2 fidelity levels).



(b) Minimisation of 3D Hartmann function (3 fidelity levels).



(c) Minimisation of 6D Hartmann function (4 fidelity levels).



(d) Maximisation of the 8D Borehole function (2 fidelity levels).

Figure 3.5.1: MUMBO provides high-precision optimisation with low computational overheads for discrete MF optimisation. We show the means and standard errors across 20 random initialisations.

on the 6D task even when based on 200 samples (requiring 20 times the overhead cost of MUMBO), and proved computationally infeasible to provide reasonable 8D optimisation (and is therefore not included in Figure 3.5.1d).

Note that MUMBO based on a single sample of g^* is a more aggressive optimiser, as we only consider a single (highly-likely) max-value. Although less robust than MUMBO-10 on average across our examples, this aggressive behaviour can allow faster optimisation, but only for certain problems (Figure 3.5.1(c)).

3.5.3 Continuous Multi-fidelity BO: FABOLAS

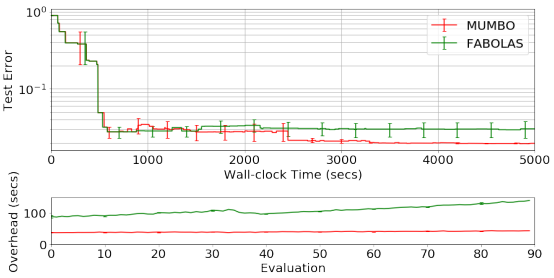
FABOLAS (Klein et al., 2017a) is a MF framework for tuning the hyper-parameter of machine learning models whilst dynamically controlling the proportion of available data

$z \in (0, 1]$ used for each hyper-parameter evaluation. By using the MTBO acquisition and imposing strong assumptions on the structure of the fidelity space, FABOLAS is able to achieve highly efficient hyper-parameter tuning. The use of a 'degenerate' kernel (Rasmussen, 2004a) with basis function $\phi(z) = (z, (1-z)^2)^T$ (i.e performing Bayesian linear regression over this basis) enforces monotonicity and strong smoothness across the fidelity space, acknowledging that when using more computational resources, we expect less biased and less noisy estimates of model performance. These assumptions induce a product kernel over the whole space $\mathcal{X} \times \mathcal{Z}$ of:

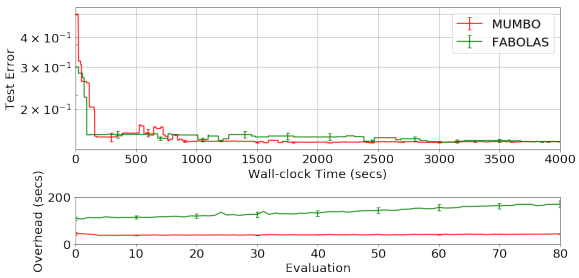
$$k((\mathbf{x}, z), (\mathbf{x}', z')) = k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')(\phi(z)^T \Sigma_1 \phi(z')),$$

where Σ_1 is a matrix in $\mathbb{R}^{2\times 2}$ to be estimated alongside the parameters of $k_{\mathcal{X}}$. Similarly, evaluation costs are also modelled in log space, with a GP over the basis $\phi_c(z) = (1, z)^T$ providing polynomial computational complexity of arbitrary degree. We follow the original FABOLAS implementation exactly, using MCMC to marginalise kernel parameters over hyper-priors specifically chosen to speed up and stabilise the optimisation.

In Figure 3.5.2 we replace the MTBO acquisition used within FABOLAS with a MUMBO acquisition, demonstrating improved optimisation on two examples from (Klein et al., 2017a). As the goal of MF hyper-parameter tuning is to find high-performing hyper-parameter configurations after using as few computational resources as possible, including both the fitting of models and calculating the next hyper-parameter and fidelity to query, we present incumbent test error (calculated offline after the full optimisation) against wall-clock time. Note that the entire time span considered for our MNIST example is still less than required to try just four hyper-parameter evaluations on the whole data and so we cannot include standard BO approaches in these figures. MUMBO's significantly reduced computational overhead allows twice as many hyper-parameter evaluations as MTBO for the same wall clock time, even though MUMBO consistently queries larger proportions of the data (on average 30% rather than 20% by MTBO). Moreover, unlike MTBO, with an overhead that increases as the optimisation progresses, MUMBO remains computationally light-weight throughout and has substantially less variability in the performance of



(a) Tuning C and gamma for an SVM to minimise MNIST digit classification error.



(b) Tuning C and gamma for an SVM to minimise Vehicle Registration classification error.

Figure 3.5.2: MUMBO provides MF hyper-parameter tuning with a much lower overhead than FABOLAS. We show the means and standard errors based on 5 runs.

the chosen hyper-parameter configuration. While we do not compare FABOLAS against other hyper-parameter tuning methods, we have demonstrated that, for this well-respected tuner and complicated MF BO problem, that MUMBO provides an improvement in efficiency and a substantial reduction in computational cost.

3.5.4 Multi-task BO: FASTCV

We now consider the MT framework of FASTCV (Swersky et al., 2013). Here, we seek the simultaneous optimisation of the K performance estimates making up K-fold CV. Therefore, our objective function g is the average score across a categorical fidelity space $\mathcal{Z} = \{1, .., K\}$. Each hyper-parameter is evaluated on a single fold, with the corresponding evaluations on the remaining folds inferred using the learned between-fold relationship. Therefore, we can evaluate K times as many distinct hyper-parameter choices as when tuning with full K-fold CV whilst retaining the precise performance estimates required for reliable tuning (Moss et al., 2018, 2019).

Unlike our other examples, this is not a MF BO problem as our fidelities have the same query cost (at $1/K^{th}$ the cost of the true objective). Recall that all we require to use MUMBO is the predictive joint (bi-variate Gaussian) distribution between an objective function $g(\mathbf{x})$ and fidelity evaluations $f(\mathbf{x}, z)$ for each choice of $\mathbf{x}$. For

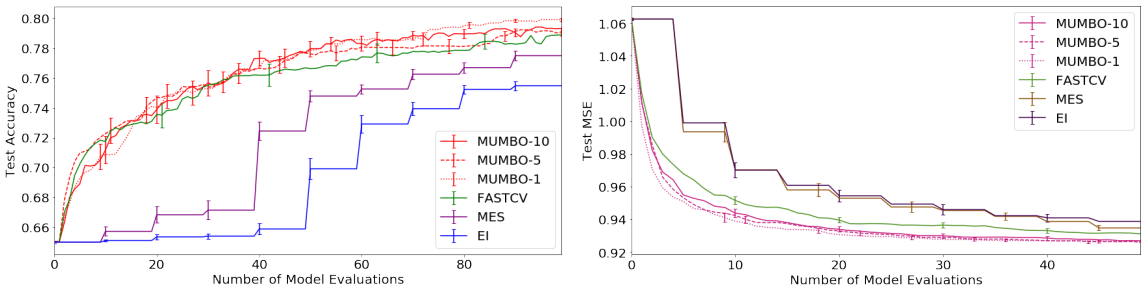
FASTCV, g corresponds with the average score across folds and so (following our earlier notation) our underlying GP provides;

$$\mu_g(\mathbf{x}) = \frac{1}{K}\sum_{z\in\mathcal{Z}}\mu^n(\mathbf{x},z), \quad \sigma_g(\mathbf{x})^2 = \frac{1}{K^2}\sum_{z\in\mathcal{Z}}\sum_{z'\in\mathcal{Z}}\Sigma^n((\mathbf{x},z),(\mathbf{x},z')),$$

where $\mu^n(\mathbf{x},z)$ is the predictive mean performance of $\mathbf{x}$ on fold z and $\Sigma^n((\mathbf{x},z),(\mathbf{x},z'))$ is the predictive co-variance between the evaluations of $\mathbf{x}$ on folds z and z' after n hyper-parameter queries. Similarly, we have the correlation between evaluations of $\mathbf{x}$ on fold z with the average score g as

$$\rho(\mathbf{x},z) = \frac{\frac{1}{K}\sum_{z'\in\mathcal{Z}}\Sigma^n((\mathbf{x},z),(\mathbf{x},z'))}{\sqrt{\sigma_g^2(\mathbf{x})\Sigma^n((\mathbf{x},z),(\mathbf{x},z))}},$$

providing all the quantities required to use MUMBO.



(a) Tuning two SVM hyper-parameters to maximise sentiment classification accuracy for IMDB movie reviews by 10-fold CV.

(b) Tuning four hyper-parameters for probabilistic matrix factorisation to minimise mean reconstruction error for movie recommendations using 5-fold CV.

Figure 3.5.3: MUMBO provides faster hyper-parameter tuning than the MT framework of FASTCV. We show the mean and standard errors across 40 runs. To measure total computational cost we count each evaluation by K-fold CV as K model fits. Experimental details are included in Appendix A.2.3.

In the original implementation of FASTCV, successive hyper-parameter evaluations are chosen using a two-step heuristic based on expected improvement. Firstly they choose the next hyper-parameter $\mathbf{x}$ by maximising the expected improvement of the

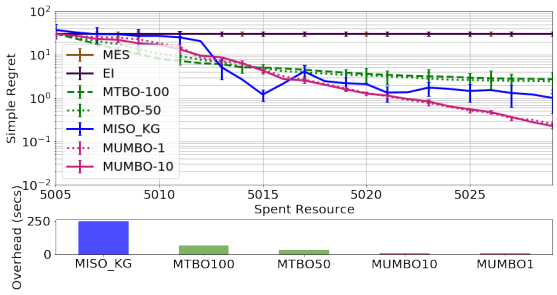


Figure 3.5.4: The 2D noisy Rosenbrock function (2 fidelities).

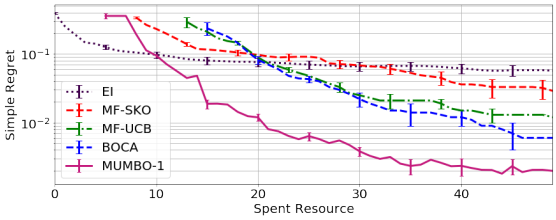


Figure 3.5.5: The 2-d Currin function (1-d continuous fidelity space)

predicted average performance and secondly choosing the fold that has the largest fold-specific expected improvement at this chosen hyper-parameter. We instead propose using MUMBO to provide a principled information-theoretic extension to FASTCV. Figure 3.5.3 demonstrates that MUMBO provides an efficiency gain over FASTCV, while finding high-performing hyper-parameters substantially faster than standard BO tuning by K-fold CV (where we require K model evaluations for each unique hyper-parameter query.

3.5.5 Wider Comparison With Existing Methods

Finally, we make additional comparisons with existing MT acquisition functions in Figures 3.5.4 and 3.5.5. Knowledge-gradient search strategies are designed to provide particularly efficient optimisation for noisy functions, however this high performance comes with significant computational overheads. Although providing reasonable early performance on a synthetic noisy MF optimisation task (Figure 3.5.4), we see that MUMBO is able to provide higher-precision optimisation and that, even for this simple 2-d search space, MISO-KG's optimisation overheads are magnitudes larger than MUMBO (and MTBO). Figure 3.5.5 shows that MUMBO substantially outperforms existing approaches on a continuous MF benchmark. MF-SKO, MF-UCB and BOCA's search strategies are guided by estimating g^* (rather than $\mathbf{x}^*$) and so have comparable computational cost to MUMBO, however, only MUMBO is able to provide high-precision optimisation with this low-computational overhead.

3.6 Conclusions

We have derived a novel computationally light information-theoretic approach for general discrete and continuous multi-task Bayesian optimisation, along with an open and accessible code base that will enable users to deploy these methods and improve replicability. MUMBO reduces uncertainty in the optimal value of the objective function with each subsequent query, and provides principled decision-making across general multi-task structures at a cost which scales only linearly with the dimension of the search space. Consequently, MUMBO substantially outperforms current acquisitions across a range of optimisation and hyper-parameter tuning tasks.

Chapter 4

BOSS: Bayesian Optimisation Over String Spaces

Status: Published as Moss H. B., Beck D., Leslie D. S., Gonzalez J. & Rayson P., Bayesian Optimisation over String spaces, *Advances in Neural Information Processing Systems*, 2020.

4.1 Preface

In this chapter, we take a brief respite from information-theoretic Bayesian optimisation (BO), instead focusing on building a BO framework suitable for high-cost string design problems. This chapter develops a BO method which acts directly over raw strings, proposing the first uses of string kernels and genetic algorithms within BO loops. Recent applications of BO over strings have been hindered by the need to map inputs into a smooth and unconstrained latent space. Learning this projection is computationally intensive and requires large amounts of data. Our approach instead builds a powerful Gaussian process surrogate model based on string kernels, naturally supporting variable length inputs, and performs efficient acquisition function maximisation for spaces with syntactical constraints. Experiments demonstrate considerably improved optimisation over existing approaches across a broad range of constraints, including the popular setting where syntax is governed by a context-free grammar. In Chapter 5, we revisit

and extend this framework to provide batch optimisation using information-theoretic search.

4.2 Introduction

Many tasks in chemistry, biology and machine learning can be framed as optimisation problems over spaces of strings. Examples include the design of synthetic genes (González et al., 2014; Tanaka and Iwata, 2018) and chemical molecules (Griffiths and Hernández-Lobato, 2020; Gómez-Bombarelli et al., 2018), as well as problems in symbolic regression (Kusner et al., 2017) and kernel design (Lu et al., 2018). Common to these applications is the high cost of evaluating a particular input, for example requiring resource and labor-consuming wet lab tests. Consequently, most standard discrete optimisation routines are unsuitable, as they require many evaluations.

Bayesian Optimisation (Shahriari et al., 2016, BO) has recently risen as an effective strategy to address the applications above, due to its ability to find good solutions within heavily restricted evaluation budgets. However, the vast majority of BO approaches assume a low dimensional, mostly continuous space; string inputs have to be converted to fixed-size vectors such as bags-of-ngrams or latent representations learned through an unsupervised model, typically a variational autoencoder (Kingma and Welling, 2014, VAE). In this work, we remove this encoding step and propose a BO architecture that operates directly on raw strings through the lens of convolution kernels (Haussler, 1999). In particular, we employ a Gaussian Process (Rasmussen, 2004a, GP) with a *string kernel* (Lodhi et al., 2002) as the surrogate model for the objective function, measuring the similarity between strings by examining shared non-contiguous sub-sequences. String kernels provide an easy and user-friendly way to deploy BO loops directly over strings, avoiding the expensive architecture tuning required to find a useful VAE. At the same time, by using a kernel trick to work in much richer feature spaces than the bags-of-ngrams vectors, string kernels can encode the non-contiguity known to be informative when modelling genetic sequences (Vert, 2007) and SMILES (Anderson et al., 1987) representations of molecules (Cao

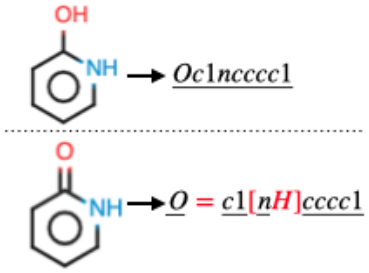


Figure 4.2.1: Similar molecules have SMILES strings with local differences (red) but common non-contiguous sub-sequences.

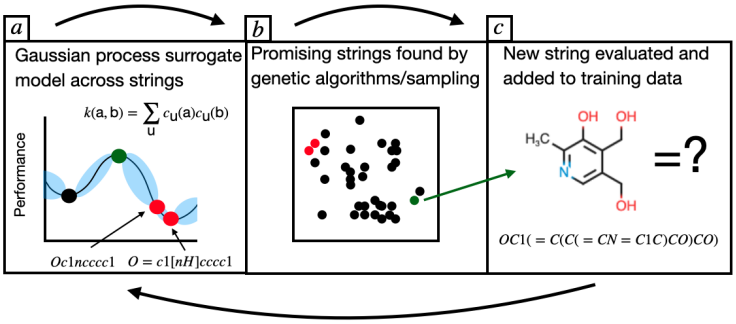


Figure 4.2.2: BO loop for molecule design using a string kernel surrogate model (a) and genetic algorithms for acquisition function maximisation (b).

et al., 2012)(see Figure 4.2.1). We show that our string kernel's two parameters can be reliably fine-tuned to model complex objective functions with just a handful of function evaluations, without needing the large collections of unlabelled data required to train VAEs.

Devising a BO framework directly over strings raises the question of how to maximise *acquisition functions*; heuristics used to select new evaluation points. Standard BO uses numerical methods to maximise these functions but these are not applicable when the inputs are discrete structures such as strings. To address this challenge, we employ a suite of genetic algorithms (Whitley, 1994) to provide efficient exploration of string spaces under a range of syntactical constraints.

Our contributions can be summarised as follows:

- We introduce string kernels into BO, providing powerful GP surrogate models of complex objective functions with just two data-driven parameters (Figure 4.2.2.a).
- We propose a suite of genetic algorithms suitable for efficiently optimising acquisition functions under a variety of syntactical constraints (Figure 4.2.2.b).
- We demonstrate that our framework out-performs established baselines across four scenarios encompassing a range of applications and diverse set of constraints.

4.3 Related Work

BO by feature extraction BO has previously been applied to find genes with desirable features: a high-cost string optimisation problem across a small alphabet of four bases. Genes are represented as either codon frequencies (a bags-of-ngrams of triplets of characters) (González et al., 2014), or as a one-hot-encoding of the genes at each location in the string (Tanaka and Iwata, 2018). Although these representations are sufficient to allow BO to improve over random gene designs, each mapping discards information known to be important when modelling genes. A bags-of-ngrams representation ignores positional and contextual information by modelling characters to have equal effect regardless of position or context, whereas a one-hot encoding fails to exploit translational invariance. Moreover, by assuming that all potential genes belong to a small fixed set of candidates, González et al. (2014) and Tanaka and Iwata (2018) ignore the need to provide an efficient acquisition optimisation routine. This assumption is unrealistic for many real gene design loops and is tackled directly in our work.

BO with VAEs Kusner et al. (2017); Gómez-Bombarelli et al. (2018) and Lu et al. (2018) use VAEs to learn latent representations for string spaces following the syntactical constraints given by context-free grammars (CFG). Projecting a variable-length and constrained string space to an unconstrained latent space of fixed dimensions requires a sophisticated mapping, which in turn requires a lot of data to learn. As BO problems never have enough string-evaluation pairs to learn a supervised mapping, VAEs must be trained to reconstruct a large collection of valid strings sampled from the CFG. A representation learned in this purely unsupervised manner will likely be poorly-aligned with the problem's objective function, under-representing variation and over-emphasising sub-optimal areas of the original space. Consequently, VAE's often explore only limited regions of the space and have 'dead' areas that decode to invalid strings (Griffiths and Hernández-Lobato, 2020). Moreover, performance is sensitive to the arbitrary choice of the closed region of latent space considered for BO.

Evolutionary algorithms in BO The closest existing idea to our work is that of Kandasamy et al. (2018b), where an evolutionary algorithm optimises acquisition functions over a space of neural network architectures. However, their approach does not support syntactically constrained spaces and, as it is based solely on local mutations, cannot perform the global search required for large string spaces. Moreover, as their kernel is based on an *optimal transport* distance between individual network layers, it does not model the non-contiguous features supported by string kernels. Contemporaneous work of Swersky et al. (2020) also considers BO over strings and proposes an evolutionary algorithm based on generative modelling for their acquisition function optimisation. However, their approach relies on ensembles of neural networks rather than GP surrogate models, is suitable for strings of up to only 100 characters and does not support spaces with syntactic constraints.

4.4 Preliminaries

Bayesian Optimisation In its standard form, BO seeks to maximise a smooth function $g : \mathcal{X} \to \mathbb{R}$ over a compact set $\mathcal{X} \subset \mathbb{R}^d$ in as few evaluations as possible. Smoothness is exploited to predict the performance of not yet evaluated points, allowing evaluations to be focused into promising areas of the space. BO loops have two key components - a *surrogate model* and an *acquisition function*.

Surrogate model To predict the values of g across $\mathcal{X}$, a surrogate model is fit to the previously collected (and potentially noisy) evaluations $D_t = \{(\mathbf{x}_i, y_i)\}_{i=1,..,t}$, where $y_i = g(\mathbf{x}_i) + \epsilon_i$ for iid Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. As is standard in the literature, we use a GP surrogate model (Rasmussen, 2004a). A GP provides non-parametric regression of a particular smoothness controlled by a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ measuring the similarity between two points.

Acquisition function The other crucial ingredient for BO is an acquisition function $\alpha_t : \mathcal{X} \to \mathbb{R}$, measuring the utility of making a new evaluation given the predictions of our surrogate model. We use the simple yet effective search strategy of expected improvement (EI): evaluating points yielding the largest improvement over current

	Sub-sequence Occurrence, $\mathbf{u}$			Sub-sequence Contribution, $c_{\mathbf{u}}(\mathbf{s})$		
String, $\mathbf{s}$	"genic"	"geno"	"ge"	"genic"	"geno"	"ge"
"genetics"	"**gen**et**ic**s"		"**g**e**n**etics" "**g**en**e**tics"	$\lambda_m^5\lambda_g^2$	0	$\lambda_m^2(1+\lambda_g^2)$
"genomic"	"**gen**om**ic**"	"**geno**mic"	"**g**enomic"	$\lambda_m^5\lambda_g^2$	λ_m^4	λ_m^2
"genomes"		"**geno**mes"	"**g**enome**s**"	0	λ_m^4	$\lambda_m^2(1+\lambda_g^4)$

Table 4.4.1: Occurrences (left panel) and respective contributions function values (right panel) of sample sub-sequences when evaluating the strings "genetics", "genomic" and "genomes".

evaluations. Although any BO acquisition function is compatible with our framework, we choose EI for this chapter as it provides an effective search whilst not incurring significant BO overheads, allowing focus on the aspects of these problems unique to string spaces. In Chapter 5, we examine an information-theoretic acquisition function for string optimisation. A single BO loop is completed by evaluating the location with maximal utility $\mathbf{x}_{n+1} = \text{argmax}_{\mathbf{x}\in\mathcal{X}}\, \alpha_n(\mathbf{x})$ and is repeated until the optimisation budget is exhausted.

String Kernels (SKs) SKs are a family of kernels that operate on strings, measuring their similarity through the number of *sub-strings* they share. Specific SKs are then formally defined by the particular definition of a sub-string they encompass, which defines the underlying feature space of the kernel. In this work, we employ the *Sub-sequence String Kernel* (SSK) (Lodhi et al., 2002; Cancedda et al., 2003), which uses *sub-sequences* of characters as features. The sub-sequences can be non-contiguous, giving rise to an exponentially-sized feature space. While enumerating such a space would be infeasible, the SSK uses the kernel trick to avoid computation in the primal space, enabled via an efficient dynamic programming algorithm. By matching occurrences of sub-sequences, SSKs can provide a rich contextual model of string data,

moving far beyond the capabilities of popular bag-of-ngrams representations where only contiguous occurrences of sub-strings are modelled.

Formally, an n^{th} order SSK between two strings $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$k_n(\mathbf{a}, \mathbf{b}) = \sum_{\mathbf{u} \in \Sigma^n} c_{\mathbf{u}}(\mathbf{a}) c_{\mathbf{u}}(\mathbf{b}) \quad \text{for} \quad c_{\mathbf{u}}(\mathbf{s}) = \lambda_m^{|\mathbf{u}|} \sum_{1<i_1<..<i_{|\mathbf{u}|}<|\mathbf{s}|} \lambda_g^{i_{|\mathbf{u}|}-i_1} \mathbb{1}_{\mathbf{u}}((s_{i_1}, .., s_{i_{|\mathbf{u}|}})),$$

where Σ^n denotes the set of all possible ordered collections containing up to n characters from our alphabet Σ, $\mathbb{1}_{\mathbf{x}}(\mathbf{y})$ is the indicator function checking if the strings $\mathbf{x}$ and $\mathbf{y}$ match, and the match decay $\lambda_m \in [0, 1]$ and gap decay $\lambda_g \in [0, 1]$ are kernel hyper-parameters. Intuitively, $c_{\mathbf{u}}(\mathbf{s})$ measures the contribution of sub-sequence $\mathbf{u}$ to string $\mathbf{s}$, and the choices λ_m and λ_g control the relative weighting of long and/or highly non-contiguous sub-strings (Table 4.4.1). To allow the meaningful comparison of strings of varied lengths, we use a normalised string kernel $\tilde{k}_n(\mathbf{a}, \mathbf{b}) = k_n(\mathbf{a}, \mathbf{b}) / \sqrt{k_n(\mathbf{a}, \mathbf{a}) k_n(\mathbf{b}, \mathbf{b})}$.

4.5 Bayesian Optimisation Directly On Strings

In string optimisation tasks, we seek the optimiser $\mathbf{s}^* = \text{argmax}_{\mathbf{s} \in S} g(\mathbf{s})$ of a function g across a set of strings S. In this work, we consider different scenarios for S arising from three different types of syntactical constraints and a sampling-based approach for when constraints are not fully known. In Section 4.6 we demonstrate the efficacy of our proposed framework across all four scenarios.

1. **Unconstrained** Any string made exclusively from characters in the alphabet Σ are allowed. S contains all these strings of any (or a fixed) length.
2. **Locally constrained** S is a collection of strings of fixed length, where the set of possible values for each character depends on its position in the string, i.e. the character s_i at location i belongs to the set $\Sigma_i \subseteq \Sigma$.
3. **Grammar constrained** S is the set of strings made from Σ that satisfy the syntactical rules specified by a context-free grammar.
4. **Candidate Set**. A space with unknown or very complex syntactical rules, but for which we have access to a large collection S of valid strings.

4.5.1 Surrogate Models for String Spaces

To build a powerful model across string spaces, we propose using an SSK within a GP. However, the vanilla SSK presented above is not immediately suitable due to its substantial computational cost. In contrast to most applications of GPs, BO surrogates are trained on small datasets and so the computational bottleneck is not inversion of Gram matrices. Instead, the primary contributors to cost are the many kernel evaluations required to maximise acquisition functions. Therefore, we develop two modifications to improve the efficiency and scalability of our SSK.

Efficiency Using the dynamic program proposed by Lodhi et al. (2002), obtaining a single evaluation of an n^{th} order SSK is $O(nl^2)$, where $l = \max(|\mathbf{a}|, |\mathbf{b}|)$. For our applications where many kernel evaluations are to be made in parallel, we found the vectorised formulation of Beck and Cohn (2017) to be more appropriate. Although, having a larger complexity of $O(nl^3)$, a vectorised formulation can exploit recent advancements in parallel processing and in practice was substantially faster. Moreover, Beck and Cohn (2017)'s formulation provides gradients with respect to the kernel parameters, allowing their fine-grained tuning to a particular optimisation task. We found the particular string kernel proposed by Beck and Cohn (2017) — with individual weights for each different sub-sequence length — to be overly flexible for our BO applications. We adapt their recursive algorithm for our SSK (Appendix B.1).

Scalability Even with a vectorised implementation, SSKs are computationally demanding for long strings. Comprehensively tackling the scalability of string kernels is beyond the scope of this work and is an area of future research. However, we perform a simple novel approximation to allow demonstrations of BO for longer sequences: we split sequences into m parts, applying separate string kernels (with tied kernel parameters) to each individual part and summing their values. This reduces the complexity of kernel calculations from $O(nl^3)$ to $O(nl^3/m^2)$ without a noticeable effect on performance (Section 4.6.2). Moreover, the m partial kernel calculations can be computed in parallel.

4.5.2 Acquisition function optimisation over String Spaces

We now present a suite of routines providing efficient acquisition function optimisation under different types of syntactical constraints. In particular, we propose using *genetic algorithms* (GA) (Whitley, 1994), biologically inspired optimisation routines that successively evaluate and evolve populations of n strings. Candidate strings undergo one of two stochastic perturbations: a *mutation* operation producing a new offspring string from a single parent, and a *crossover* operation combining attributes of two parents to produce two new offspring. GAs are a natural choice for optimising acquisition functions as they avoid local maxima by maintaining diversity and the evolution can be carefully constructed to ensure compliance to syntactical rules. We stress that GAs require many function evaluations and so are not suitable for optimising a high-cost objective function, just for this 'inner-loop' maximisation. To highlight robustness, the parameters of our GAs are not tuned to our individual experiments (Appendix B.4.3). When syntactical rules are poorly understood and cannot be encoded into the optimisation, we recommend the simple but effective strategy of maximising acquisition functions across a random sample of valid strings.

GAs for unconstrained and locally constrained string spaces For our first two types of syntactical constraints, standard definitions of crossover and mutation are sufficient. For mutation, a random position i is chosen and the character at this point is re-sampled uniformly from the set of permitted characters Σ_i (or just Σ) for locally constrained (unconstrained) spaces. For crossover, a random location is chosen within one of the parent strings and the characters up until the crossover point are swapped between the parents. Crucially, the relative positions of characters in the strings are not changed by this operation and so the offspring strings still satisfy the space's constraints.

GA for grammar-constrained string spaces Context free grammars (CFG) are collections of rules able to encode many common syntactical constraints (see Appendix B.2 and Hopcroft et al. (2001)). While it is difficult to define character-level mutation

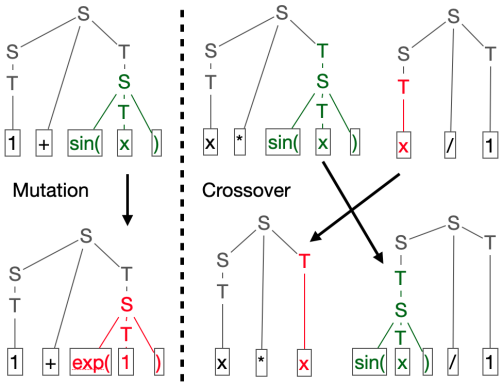


Figure 4.5.1: Mutations and crossover of arithmetic expressions following the grammar:
$S \rightarrow S$ '+' T | S '$*$' T | S '/' T | T
$T \rightarrow$ '$sin($' S '$)$' | '$exp($' S '$)$' | 'x' | '1'

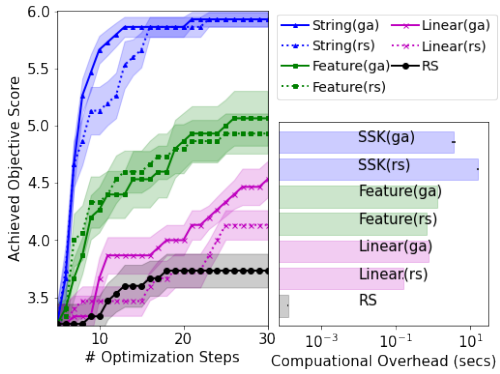


Figure 4.5.2: Performance and computational overhead when searching for binary strings of length 20 with the most non-overlapping occurrences of "101" (higher is better).

and crossover operations that maintain grammatical rules over strings of varying length, suitable operations can be defined over parse trees, structures detailing the grammatical rules used to make a particular string. Following ideas from grammar-based genetic programming (Mckay et al., 2010), mutations randomly replace sub-trees with new trees generated from the same head node, and crossover swaps two sub-trees sharing a head node between two parents (see Figure 4.5.1). When sampling strings from the grammar to initialise our GA and perform mutations, the simple strategy of building parse trees by recursively choosing random grammar rules produces long and repetitive sequences. We instead employ a sampling strategy that down-weights the probability of selecting a particular rule based on the number of times it has already occurred in the current parse tree branch (Appendix B.3).

4.6 Experiments

We now evaluate our proposed BO framework on tasks from a range of fields and syntactical constraints. Our code is available at *github.com/henrymoss/BOSS* and is built upon the Emukit Python package (Paleyes et al., 2019). All results are based

on runs across 15 random seeds, showing the mean and a single standard error of the best objective value found as we increase the optimisation budget. The computational overhead of BO (the time spent fitting the GP and maximising the acquisition function) is presented as average wall-clock times. Although acquisition function calculations could be parallelised across the populations of our GA at each BO step, we use a single-core Intel Xeon 2.30GHz processor to paint a clear picture of computational cost.

Considered BO approaches For problems with fixed string-length, we compare our SSK with existing approaches to define GP models over strings. In particular, we apply the squared exponential (SE) kernel (Rasmussen, 2004a) to a bags-of-ngrams feature representation of the strings. SSKs (feature) representations consider sub-sequences of up to and including five non-contiguous (contiguous) characters, with additional choices demonstrated in Appendix B.4.1. We also provide a linear kernel applied to one-hot encodings of each character, a common approach for BO over categorical spaces. The strategy of sequentially querying random strings is included for all plots and we introduce task-specific baselines alongside their results. After a random initialisation of $\min(5, |\Sigma|)$ evaluations, kernel parameters are re-estimated to maximise model likelihood before each BO step.

4.6.1 Unconstrained Synthetic String Optimisation

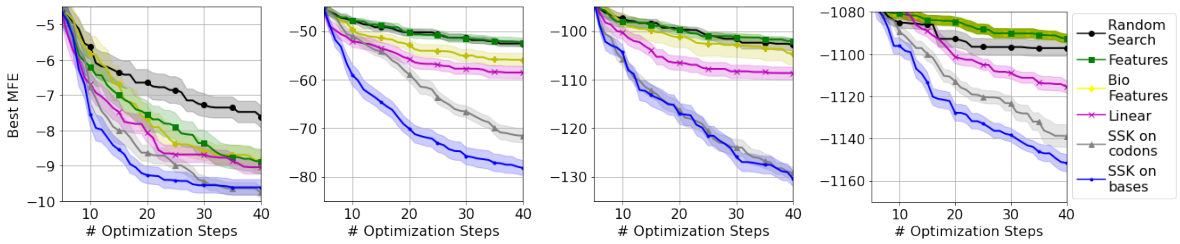
We first investigate a set of synthetic string optimisation problems over unconstrained string spaces containing all strings of a particular length built from a specific alphabet. Objective functions are then defined around simple tweaks of counting the occurrence of a particular sub-string. Although these tasks seem simple, we show in Appendix B.4.1 that they are more difficult than the synthetic benchmarks used to evaluate standard BO frameworks. The results for seven synthetic string optimisation tasks are included in Table 4.6.1, with a deeper analysis of a single task in Figure 4.5.2. Additional figures for the remaining tasks showing broadly similar behaviour are included in the supplement. To disentangle the benefits provided by the SSK and our GA, we consider

Problem Definition			Mean performance with std error (2 s.f.)				
Objective	Space	Steps	SSK (ga)	SSK (rs)	Feature (ga)	Linear (ga)	RS
# of "101"	$\{0,1\}^{20}$	10	**100 (0.0)**	96 (1.4)	97 (2.2)	58 (3.0)	58 (2.6)
# of "101", no overlaps	$\{0,1\}^{20}$	15	**98 (1.4)**	94 (2.6)	76 (4.1)	64 (2.6)	60 (3.1)
# of "10??1"	$\{0,1\}^{20}$	25	**98 (1.6)**	95 (1.6)	64 (2.0)	64 (3.3)	56 (2.0)
# of "101" in 1st 15 chars	$\{0,1\}^{30}$	40	**91 (2.6)**	83 (1.7)	67 (3.0)	69 (2.6)	61 (2.3)
# of "101" + $\mathcal{N}(0,2)$	$\{0,1\}^{20}$	25	**98 (2.1)**	95 (1.4)	51 (3.9)	40 (4.0)	45 (3.8)
# of "123"	$\{0,..,3\}^{30}$	20	**81 (2.3)**	35 (2.8)	69 (5.4)	23 (2.0)	17 (1.5)
# of "01??4"	$\{0,..,4\}^{20}$	50	**67 (4.5)**	38 (2.6)	35 (4.0)	33 (3.1)	29 (2.6)

Table 4.6.1: Optimisation of functions counting occurrences of a particular pattern within strings of varying lengths and alphabets ("?" matches any single character). Evaluations are standardised $\in [0, 100]$ and higher scores show superior optimisation. Our SSK provides particularly strong performance for complicated patterns (red) or when evaluations are contaminated with Gaussian noise (blue). Our GA acquisition maximiser is especially effective for large alphabets (yellow).

two acquisition optimisers: random search across $10,000$ sample strings (denoted *rs* and not to be confused with the random search used to optimise the original objective function) as well as our genetic algorithms (*ga*) limited to ≤ 100 evolutions of a population of size 100. The genetic algorithm is at most as computationally expensive (in terms of acquisition function evaluations) as the random search optimiser, but in practice is usually far cheaper due to the GA's early-stopping.

Figure 4.5.2 demonstrates that our approach provides highly efficient global optimisation, dramatically out-performing random search and BO with standard kernels. Interestingly, although the majority of our approach's advantage comes from the SSK, our genetic algorithm also contributes significantly to performance, out-performing the random search acquisition function optimiser in terms of both optimisation and computational overhead. Although SSKs incur significant BO overheads, they achieve high-precision optimisation after far fewer objective queries, meaning a substantial reduction in overall optimisation costs for all but the cheapest objective functions. Table 4.6.1 shows that our approach provides superior optimisation across a range of tasks



$(\ell, m) = (30, 1)$ $(\ell, m) = (186, 2)$ $(\ell, m) = (360, 8)$ $(\ell, m) = (3672, 64)$

Figure 4.6.1: Finding the representation with minimal *minimum free-folding energy* (MFE) for proteins of length ℓ. SSKs are applied to codon or base representations split into m or $3m$ parts, respectively.

designed to test our surrogate model's ability to model contextual, non-contiguous and positional information.

4.6.2 Locally Constrained Protein Optimisation

For our second set of examples, we consider the automatic design of genes that strongly exhibit some particular property. We follow the set-up of González et al. (2014), which optimises across the space of all the genes encoding a particular protein. Proteins are sequences made from 20 amino acids, but redundancy in genetic coding means that individual proteins can be represented by many distinct genes, each with differing biological properties. For this experiment, we seek protein representations with minimal *minimum free-folding energy*, a fundamental biological quantity determined by how a protein 'folds' in 3-D space. The prediction of the most likely free-folding energy for large sequences remains an important open problem (AlQuraishi, 2019), whereas calculating the minimal free-folding energy (across all possible folds) is possible for smaller sequences using the ViennaRNA software (Lorenz et al., 2011). We acknowledge that this task may not be biologically meaningful on its own, however, as free-folding energy is of critical importance to other down-stream genetic prediction tasks, we believe it to be a reasonable proxy for wet-lab-based genetic design loops. This results in a truly challenging black-box string optimisation, requiring modelling of positional and frequency information alongside long-range and non-contiguous relationships.

Each amino acid in a protein sequence can be encoded as one of a small subset of 64 possible codons, inducing a locally constrained string space of genes, where the set of valid codons depends on the position in the gene (i.e the particular amino acid represented by that position). Alternatively, each codon can be represented as triples of the bases (A,C,T,G), forming another locally constrained string space of three times the length of the codon representation but with a smaller alphabet size of 4. As well as applying the linear and feature kernels to the base representations, we also consider the domain-specific representation used by González et al. (2014) (denoted as Bio-Features) that counts codon frequencies and four specific biologically inspired base-pairs. Figure 4.6.1 demonstrates the power of our framework across 4 proteins of varying length. Additional details and wall-clock timing are provided in Appendix B.4.2. SSKs provide particularly strong optimisation for longer proteins, as increasing the length renders the global feature frequencies less informative (with the same representations used for many sequences) and the linear kernel suffers the curse of dimensionality. Note that unlike existing BO frameworks for gene design, our framework explores the large space of all possible genes rather than a fixed small candidate set.

4.6.3 Grammar Constrained String Optimisation

We now consider a string optimisation problem under CFG constraints. As these spaces contain strings of variable length and have large alphabets, the linear and feature kernel baselines considered earlier cannot be applied. However, we do consider the VAE-based approaches of Kusner et al. (2017) and Gómez-Bombarelli et al. (2018) denoted *GVAE* and *CVAE* for a grammar VAE and character VAE, respectively. We replicate the symbolic regression example of Kusner et al. (2017), using their provided VAEs pre-trained for this exact problem. Here, we seek a valid arithmetic expression that best mimics the relationship between a set of inputs and responses, whilst following the syntactical rules of a CFG (Appendix B.2). We investigate both BO and random search in the latent space of the VAEs, with points chosen in the latent space decoded back to strings for objective function evaluations (details in Appendix B.4.3). We

sample 15 strings for initialisation of our GPs, which, for the VAE-approaches, are first encoded to the latent space, before being decoded for evaluation. The invalid strings suggested by *CVAE* are assigned large error.

Figure 4.6.2 shows that our approach is able to provide highly efficient BO across a space with complicated syntactical constraints, out-performing the VAE methods which are beaten by even random search (a comparison not made by Kusner et al. (2017)). The difference in starting values for the performance curves in Figure 4.6.2 is due to stochasticity when encoding/decoding; initial strings are rarely decoded back to themselves but instead mapped back to a less diverse set. However, sampling directly in the latent space led to a further decrease in initialisation diversity. We stress that *CVAE* and *GVAE* were initially designed as models which, using BO-inspired arguments, could generate new valid strings outside of their training data. Consequently, they have previously been tested only in scenarios with significant evaluation budgets. To our knowledge, we are the first to analyse their performance in the low-resource loops typical in BO.

4.6.4 Optimisation Over a Candidate Set

Finally, we return to the task introduced briefly in Figure 4.2.2 of searching over SMILES strings to find molecules with desirable properties. As the validity of SMILES strings are governed by complex semantic and syntactic rules that can only be partly explained by a context-free grammar (Kraev, 2018), it is not obvious how to define a GA acquisition function optimiser that can explore the space of all valid molecules. Therefore, we consider an alternative task of seeking high-scoring molecules from within the large collection of $250,000$ candidate molecules used by Kusner et al. (2017) to train a *CVAE* and *GVAE*. Once again, we stress that Kusner et al. (2017)'s primary motivation is to use a large evaluation budget to generate new molecules outside of the candidate set, whereas we consider the simpler but still realistic task of efficiently exploring within the set's limits. At each BO step, we sample 100 candidates, querying those that maximise the acquisition function predicted by our SSK as well as by GPs with SE kernels over the VAE latent spaces. Figure 4.6.3 shows that only the SSK

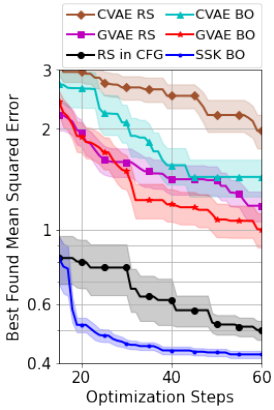


Figure 4.6.2: Searching for arithmetic expressions satisfying constraints from a CFG (lower is better).

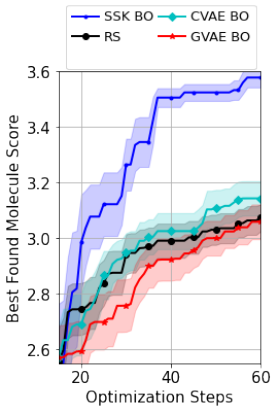


Figure 4.6.3: Searching a candidate set for molecules with desirable properties (higher is better).

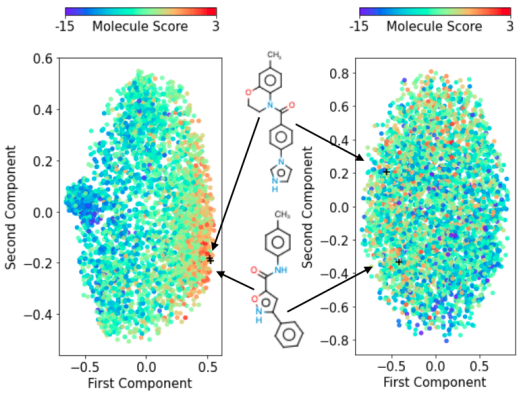


Figure 4.6.4: Top KPCA components for our SSK (left) and an SE kernel in the *GVAE* (right) for SMILES strings. Our SSK has a smoother internal representation, where 'close' points are structurally similar.

allows efficient exploration of the candidate SMILES strings. We hypothesise that the VAEs' poor performance may be partly due to the latent space's dimension which, at 56, is likely to severely hamper the performance of any BO routine.

SSK's internal representations A common way to investigate the efficacy of VAEs is to examine their latent representations. However, even if objective evaluations are smooth across this space (Kusner et al., 2017), this smoothness cannot be exploited by BO unless successfully encapsulated by the surrogate model. Although GPs have no explicit latent space, they have an intrinsic representation that can be similarly examined to provide visualisation of a surrogate model's performance. In particular, we apply kernel principal component analysis (KPCA) (Schölkopf et al., 1997) to visualise how SMILES strings map into the feature space. Figure 4.6.4 shows the first two KPCA components of our SSK and of an SE kernel within the *GVAE*'s latent space (additional visualisations in Appendix B.4.4). Although the latent spaces of the *GVAE* is known to exhibit some smoothness for this SMILES task (Kusner et al.,

2017), the smoothness is not captured by the GP model, in contrast with the SSK.

4.7 Discussion

Departing from fixed-length representations of strings revolutionises the way in which BO is performed over string spaces. In contrast to VAEs, where models are learned from scratch across thousands of parameters, an SSK's structure is predominantly fixed. By hard-coding prior linguistic intuition about the importance of incorporating non-contiguity, our SSKs have just two easily identifiable kernel parameters governing modelling of a particular objective function. We posit that the additional flexibility of VAEs is not advantageous in BO loops, where there is never enough data to reliably learn flexible models and where calibration is more important than raw predictive strength.

As well as achieving substantially improved optimisation, we provide a user-friendly BO building-block that can be naturally inserted into orthogonal developments from the literature with obvious applications within gene and chemical design loops, including batch (González et al., 2016a) (Chapter 5), multi-task (Swersky et al., 2013), multi-fidelity (Moss et al., 2020d) (Chapter 3) and multi-objective (Hernández-Lobato et al., 2016) BO, as well as BO with controllable experimental noise (Moss et al., 2020c) (Chapter 6). Moreover, our framework can be extended to other kinds of convolution kernels such as tree (Collins and Duffy, 2002) and graph kernels (Vishwanathan et al., 2010). This would allow the optimisation of other discrete structures that have previously been modelled through VAEs, including networks (Zhang et al., 2019) and molecular graphs (Kajino, 2019).

Chapter 5

GIBBON: General-purpose Information-Based Bayesian OptimisatioN

Status: In submission for *The Journal of Machine Learning Research*.

5.1 Preface

This chapter describes a general-purpose extension of max-value entropy search. By extending the MUMBO of Chapter 3, a novel approximation is proposed for the information gain — an information-theoretic quantity central to solving a range of popular BO problems, including noisy, multi-fidelity and batch optimisations in continuous and highly-structured discrete spaces. Previously, these problems have been tackled separately within information-theoretic BO, each requiring a different sophisticated approximation scheme, except for batch BO, for which no computationally-lightweight information-theoretic approach has previously been proposed. GIBBON (General-purpose Information-Based Bayesian OptimisatioN) provides a single principled framework suitable for all the above, out-performing existing approaches whilst incurring substantially lower computational overheads. In addition, unlike most information-theoretic acquisition functions, GIBBON's does not require the problem's search space

to be Euclidean and so is the first computationally-lightweight yet high-performance acquisition function that supports batch BO over general highly structured input spaces, for example when using BO to perform molecular search (see Chapter 4). Moreover, our principled derivation of GIBBON yields a natural extension for a popular heuristic for batch BO based on determinantal point processes free from user-specified parameters. Finally, the efficacy and efficiency of GIBBON is demonstrated across a suite of synthetic benchmark tasks as-well as within the molecular search loop described in Chapter 4. GIBBON is tested further in Chapter 6, where we use it to make principled decisions as part of a challenging batch multi-fidelity framework for BO problems with controllable experimental noise.

5.2 Introduction

A popular solution for the optimisation of high-cost black-box functions is Bayesian optimisation (Mockus et al., 1978, BO). By sequentially deciding where to make each evaluation as the optimisation progresses, BO can direct resources into evaluating promising areas of the search space to provide efficient optimisation. BO frameworks consist of two key components - a surrogate model and an acquisition function. By fitting a probabilistic surrogate model, typically a Gaussian process (Rasmussen, 2004a, GP), to the previously collected objective function evaluations, we are able to quantify our current belief about which areas of the search space maximize our objective function. An acquisition function then uses this belief to predict the utility of making a particular evaluation, producing large values at 'reasonable' locations. BO automatically evaluates the location that maximises this acquisition function and repeats until a sufficiently high-performing solution is found. A popular application of BO is hyper-parameter tuning, with successful applications in computer vision (Bergstra et al., 2013), text-to-speech (Moss et al., 2020a) (Chapter 7) and reinforcement learning (Chen et al., 2018b). Of particular note are the recent extensions of BO beyond Euclidean search spaces, for example when optimising synthetic genes (González et al., 2014; Tanaka and Iwata, 2018; Moss et al., 2020b) or performing molecular search (Gómez-Bombarelli

et al., 2018; Griffiths and Hernández-Lobato, 2020).

Various heuristic strategies have been developed to form BO acquisition functions, including Expected Improvement (Jones et al., 1998, EI), Knowledge Gradient (Frazier et al., 2008, KG) and Upper-Confidence Bound (Srinivas et al., 2009, UCB). More recently, a particularly intuitive and empirically effective class of acquisition functions has arisen based on information theory. Information-theoretic BO seeks to reduce uncertainty in the location of high-performing areas of the search space, as measured in terms of differential entropy. Such entropy-reduction arguments have motivated the three primary information-theoretic acquisition functions of Entropy Search (Hennig and Schuler, 2012, ES), Predictive Entropy Search (Hernández-Lobato et al., 2014, PES) and Max-value Entropy Search (Wang and Jegelka, 2017, MES), differing in their chosen measure of global uncertainty and employed approximation methods. Of particular popularity are acquisition functions based on MES, which reduce uncertainty in the maximum value attained by the objective function, a one-dimensional quantity. In contrast, both ES and PES seek to reduce uncertainty in the location of the maximum, a quantity which, as well as being well-defined only for Euclidean search spaces, requires prohibitively expensive approximation schemes. Due to the large number of acquisition function evaluations required to identify the next query point for each BO step, computational complexity is an important practical consideration when designing acquisition functions, particularly for applications with structured search spaces containing combinatorial elements.

Although the advent of MES acquisition functions has enabled the application of information-theoretic BO beyond problems with low-dimensional Euclidean search spaces, MES can not yet be regarded as a general-purpose acquisition function for two reasons.

1. Firstly, the existing extensions of MES supporting common BO extensions like Multi-fidelity BO (Moss et al., 2020d) (Chapter 3) and batch BO (Takeno et al., 2019) require additional approximations beyond those of vanilla MES, typically through the numerical integration of low-dimensional integrals. Multi-fidelity BO (also known as multi-task BO) leverages cheap approximations of

the objective function to speed up optimisation, for example through exploiting coarse resolution simulations when calibrating large climate models (Prieß et al., 2011), whereas batch BO allows multiple objective function evaluations to be queried in parallel, a scenario arising often in science applications, for example when training a collection of robots to cook (Junge et al., 2020). Therefore, although still cheaper than their ES- and PES-based counterparts, extensions of MES for multi-fidelity and batch BO do not inherit the simplicity and low-cost of vanilla MES.

2. Secondly, missing from the current extensions of MES is support for general batch BO, with just asynchronous batch BO supported (a distinction discussed in depth by Kandasamy et al. (2018a)). The asynchronous MES formulation of Takeno et al. (2019) supports scenarios where each of B workers are allocated individually to evaluate different areas of the search space, returning queries and being re-allocated one by one. In contrast, synchronous batch BO considers scenarios where where B workers are to be allocated in parallel, as is the case for many real-world settings including those relying on wet-lab evaluations, physical experiments, or any framework where workers do not have sufficient autonomy to be controlled separately. In addition, extending the asynchronous MES framework of Takeno et al. (2019) to synchronous BO require prohibitively expensive approximations. Therefore, batch applications of MES have so far relied on generic batch heuristics suitable for any BO acquisition function, including greedy allocation through local penalisation (González et al., 2016a; Alvi et al., 2019) or using probabilistic repulsion models like determinantal point processes (Kathuria et al., 2016; Dodge et al., 2017), both of which support only Euclidean search spaces.

In this work we provide a single generalisation of MES suitable for BO problems arising from any combination of noisy, batch, single-fidelity, and multi-fidelity optimisation tasks. Crucially, unlike existing extensions of MES, our general-purpose acquisition function retains the computational cost of vanilla MES, with no requirement

for numerical integration schemes, provides the first high-performing yet computationally light-weight framework for synchronous batch BO and the first high-performance batch framework suitable for search spaces consisting of discrete structures.

Our primary contributions are as follows:

1. We propose an approximation for a general-propose extension of MES named General-purpose Information-Based Bayesian Optimisation (GIBBON). This approximation enables application of MES to a wide variety of problems, including those with combinations of synchronous batch BO, multi-fidelity BO and non-Euclidean highly-structured input spaces.

2. Analysis of GIBBON leads to a novel connection between information-theoretic search, determinantal point processes (Kulesza et al., 2012, DPP) and local penalisation (González et al., 2016a), providing the first theoretical justification for key attributes of these two popular heuristics previously chosen arbitrarily by users.

3. We analyse the computational complexity of GIBBON in the wider context of information-theoretic acquisition functions, providing the first comprehensive evaluation of the computational overheads of information-theoretic BO.

4. We demonstrate the performance of GIBBON across a suite of popular benchmark optimisation tasks, including the first application of information-theoretic acquisition functions to high-cost string optimisation tasks.

The remainder of the chapter is structured as follows. Section 5.3 reviews prior work on MES and introduces the extended acquisition function that will be the focus of this work. In section 5.4, we propose the GIBBON acquisition function, before examining GIBBON in the context of existing heuristics for batch BO (Section 5.5). In Section 5.6 we consider the computational complexity of GIBBON in the wider context of information-theoretic BO. Finally, Section 5.7 provides a thorough empirical evaluation.

5.3 Max-value Entropy Search for Black-Box Function Optimisation

We now introduce max-value entropy search (MES) for BO, providing an information-theoretic motivation for the general-purpose framework that is the focus of this manuscript. We then introduce existing work that has applied more restrictive formulations of MES to deal with specific BO tasks, before briefly summarising additional popular acquisition functions that are not based on MES.

BO routines seek the global maximum

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, g(\mathbf{x})$$

of a 'smooth' but expensive to evaluate black-box function $g : \mathcal{X} \to \mathbb{R}$. By sequentially choosing where and how to make each evaluation, BO directs resources into promising areas to efficiently explore the search space $\mathcal{X} \subset \mathbb{R}^d$ and provide fast optimisation. In its simplest formulation (henceforth referred to as standard BO), BO controls the locations $\mathbf{x} \in \mathcal{X}$ at which to collect (potentially noisy) queries of the objective function. A more general framework is that of multi-fidelity BO (Swersky et al., 2013) (also known as multi-task BO), where the 'quality' of each function query can also be controlled, for example by choosing the level of noise or bias across a (possibly continuous) space of fidelities $\mathbf{s} \in \mathcal{F}$. If these lower-fidelity estimates of g are cheaper to evaluate, then BO has access to cheap but approximate information sources that can be used to efficiently maximise g. In practical terms, each step of multi-fidelity BO needs to choose a location-fidelity pair $\mathbf{z} = (\mathbf{x}, \mathbf{s}) \in \mathcal{Z} = \mathcal{X} \times \mathcal{F}$ upon which to make the next evaluation. A further extension arises as batch BO, where we wish to exploit parallel resources by choosing a set of $B \geq 1$ locations $\{\mathbf{z}_1, .., \mathbf{z}_B\} \in \mathcal{Z}^B$ to be evaluated in parallel.

BO's decisions are governed by two primary components - a surrogate model and an acquisition function. The surrogate model makes probabilistic predictions of the objective function at not-yet-evaluated locations using the already collected location-evaluation tuples $D_n = \{(\mathbf{z}_i, y_i)\}_{i=1,..,n}$. The most most popular choice of model is a

Gaussian process (Rasmussen, 2004a, GP). GPs provide non-parametric regression over all functions of a smoothness controlled by a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Crucially, our GP conditioned on D_n produces a tractable Gaussian predictive distribution that quantifies our current belief about the objective function across the whole search space. GP models can also be defined for multi-fidelity optimisation tasks (Kennedy and O'Hagan, 2000; Le Gratiet and Garnier, 2014; Klein et al., 2017a; Perdikaris et al., 2017; Cutajar et al., 2019) and when modelling highly-structured input spaces like strings (Beck and Cohn, 2017), trees (Beck et al., 2015) and molecules (Moss and Griffiths, 2020).

Given such a probabilistic model over the search space, all that remains to perform an iteration of BO is an acquisition function measuring the utility of making evaluations. The Max-value Entropy Search (MES) of Wang and Jegelka (2017), with similar formulations considered by Hoffman and Ghahramani (2015) and Ru et al. (2018), seeks to query the objective function at locations that reduce our current uncertainty in the maximum value of our objective function $g^* = \text{argmax}_{\mathbf{x}\in\mathcal{X}}\, g(\mathbf{x})$. In information theory (see Cover and Thomas, 2012, for a comprehensive introduction), uncertainty in the unknown g^* is measured by its differential entropy $H(g^*|D_n) = -\mathbb{E}_{g^*}\left[\log p(g^*)\right]$, where p is the predictive probability distribution function for g^* (as induced by the surrogate model). In particular, the utility of making an evaluation is measured as the reduction in the uncertainty of g^* it provides, a quantity known as the mutual information (MI).

Although initially proposed for just standard BO problems, an MES-based search strategy can be readily formulated for the general batch multi-fidelity framework (described above) by measuring the utility of evaluating a batch of fidelity evaluations as their joint mutual information with the maximum value. To the author's knowledge, we are the first to consider this general formulation, which we name General-purpose MES (GMES), formally expressed in Definition 5.3.1.

Definition 5.3.1 (The GMES acquisition function)**.** *The GMES acquisition function*

is defined as

$$\begin{aligned}\alpha_n^{GMES}(\{\boldsymbol{z}_i\}_{i=1}^B) :=& MI(g^*; \{y_{\boldsymbol{z}_i}\}_{i=1}^B | D_n) \\ =& H(g^*|D_n) - \mathbb{E}_{\{y_{\boldsymbol{z}_i}\}_{i=1}^B}\left[H(g^*|D_n \cup \{y_{\boldsymbol{z}_i}\}_{i=1}^B)\right], \end{aligned} \tag{5.3.1}$$

where $\{\boldsymbol{z}_i\}_{i=1}^B$ *denotes the location-fidelity pairs of the batch elements and* $y_{\boldsymbol{z}}$ *denote the yet-unobserved results of querying location-fidelity pair* $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{s}) \in \mathcal{X} \times \mathcal{F}$.

Note that standard BO, batch BO and multi-fidelity BO are trivial special cases of this general-purpose framework obtained by either or both of fixing the fidelity space $\mathcal{F}$ to a singleton containing just the true objective function or setting $B = 1$.

To provide resource-efficient optimisation, we must balance how much we expect to learn about g^* with the computational cost of the evaluations. Therefore, following the arguments of Swersky et al. (2013), each BO step chooses to evaluate the set of B locations that maximises the cost-weighted mutual information, i.e

$$\{\mathbf{z}_{|D_n|+1}, .., \mathbf{z}_{|D_n|+B}\} = \underset{\{\mathbf{z}_i\}_{i=1}^B \in \mathcal{Z}^B}{\operatorname{argmax}} \frac{\alpha_n^{\mathrm{GMES}}(\{\mathbf{z}_i\}_{i=1}^B)}{c(\{\mathbf{z}_i\}_{i=1}^B)},$$

where $c : \mathcal{Z}^B \to \mathbb{R}^+$ measures the costs of evaluating the batch. This cost function could be known *a priori* or estimated from observed costs (Snoek et al., 2012). The optimisation of acquisition functions is known as the *inner-loop* maximisation and, when considering continuous search spaces, is typically performed with a gradient-based optimiser. For discrete search spaces it is common to use local optimisation routines like DIRECT (Jones et al., 1993) or genetic algorithms (Moss et al., 2020b). For search spaces with discrete and continuous dimensions, hybrid optimisers can be used (Ru et al., 2019).

Unfortunately, calculating GMES in its full generality is challenging and providing a practically viable approximation strategy is the major contribution of this work. The primary difficulty in its computation arises from the lack of closed-form expression for the distribution of g^*, as required for all differential entropy calculations. We now end this section by discussing the three scenarios where specific sub-cases of GMES have already been used to provide highly effective BO — a noiseless variant of standard BO, multi-fidelity BO, and a special case of batch BO.

5.3.1 Max-value Entropy Search for noiseless standard BO

Firstly, we consider the original MES formulation of Wang and Jegelka (2017), where they perform standard BO with noiseless observations. This acquisition function is formally expressed as

$$\alpha_n^{\text{MES}}(\mathbf{x}) \coloneqq \text{MI}(y_{\mathbf{x}}; g^* | D_n) = H(y_{\mathbf{x}}|D_n) - \mathbb{E}_{g^*}\left[H(y_{\mathbf{x}}|g^*, D_n)|D_n\right]. \tag{5.3.2}$$

Here, the symmetric property of mutual information has been used to swap $y_{\mathbf{x}}$ and g^* in its definition, yielding an equivalent (albeit less intuitive) expansion. Crucially, the first term of the expansion of (5.3.2) is now simply the entropy of a multivariate Gaussian distribution with a convenient closed-form. Moreover, Wang and Jegelka (2017) note that under the assumption of exact objective function evaluations (where $y_{\mathbf{x}} = g(\mathbf{x})$), the distribution of $y_{\mathbf{x}}$ conditional on its maximum possible value (i.e knowing that $y_{\mathbf{x}} \leq g^*$) is simply that of a truncated Gaussian, also with a closed-form differential entropy. All that remains to calculate MES is to approximate an expectation over g^*. Wang and Jegelka (2017) build a Monte-Carlo estimate of the expectation with a set of samples $\mathcal{M}$ from g^*, providing a closed-form approximation of MES as

$$\alpha_n^{MES}(\mathbf{x}) \approx \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \left[\frac{\gamma_{\mathbf{x}}(m)\phi\left(\gamma_{\mathbf{x}}(m)\right)}{2\Phi\left(\gamma_{\mathbf{x}}(m)\right)} - \log \Phi\left(\gamma_{\mathbf{x}}(m)\right) \right],$$

where Φ and ϕ are the standard normal cumulative distribution and probability density functions (as arising from the expression for the differential entropy of a truncated Gaussian) and $\gamma_{\mathbf{x}}(m) = \frac{m - \mu_n(\mathbf{x})}{\sigma_n(\mathbf{x})}$. Here, $\mu_n(\mathbf{x})$ and $\sigma_n^2(\mathbf{x})$ are the predictive mean and standard deviation for the objective function value g at location $\mathbf{x}$ as easily extracted from our surrogate model. The set of sample max-values $\mathcal{M}$ is built by modelling the empirical cumulative distribution function of g^* with a Gumbel distribution (see Wang and Jegelka (2017) for details) which can be sampled to yield M cheap but approximate sampled max-values. This Gumbel approximation provides a fast sampling strategy and has been successful across a wide range of BO applications (Wang and Jegelka, 2017; Moss et al., 2020d,c; Takeno et al., 2019) (see Chapters 3 and 6)

For the limited set of BO problems supported by this original MES acquisition function, MES has had great empirical success, typically outperforming other information-theoretic BO methods with an order of magnitude smaller computational overhead.

However, once MES arguments are extended to support the more sophisticated BO frameworks (or even just to support noisy function evaluations), we will see that the second term of (5.3.2) is no longer (the expectation of) the differential entropy of a truncated Gaussian and additional approximations have to be made.

5.3.2 Max-value Entropy Search for multi-fidelity BO

MES-based search strategies have also been previously used for multi-fidelity BO through the MUlti-task Max-value Bayesian Optimisation (MUMBO) acquisition function of Chapter 3 (proposed in parallel by Takeno et al. (2019)) and, just like original MES, MUMBO has been shown to perform highly efficient BO. However, unlike when collecting exact observations of g, fidelity evaluations $y_{\mathbf{z}}|g^*$ no longer follow a truncated Gaussian distribution and instead follow an extended skew Gaussian distribution (as shown by Moss et al. (2020d) and re-derived in Section 5.4) which has no closed-form expression for its differential entropy (Azzalini, 1985). Therefore, the MUMBO acquisition function does not inherit all the computational savings of standard MES, requiring numerical integration. Note that by considering a single fidelity system, where low-fidelity evaluations are just noisy observations of the true objective function, a multi-fidelity formulation of MES also serves as an extended standard (single-fidelity) MES suitable for when evaluations are contaminated with observation noise.

5.3.3 Max-value Entropy Search for Batch BO

Motivated by the empirical success of MES-based acquisition functions, it is natural to wonder if they can be used for batch BO. However, of the two popular batch scenarios of asynchronous and synchronous batches commonly considered in the BO literature, only asynchronous batch BO is currently supported by a MES-based acquisition function (Takeno et al., 2019). The primary practical distinction is that, while synchronous batch acquisition functions must be able to measure the total reduction in entropy provided by the joint evaluation of B locations, asynchronous batch BO

has only to measure the relative reduction in entropy provided by making a single evaluation whilst taking into account the $B-1$ pending evaluations. Through clever algebraic manipulations, Takeno et al. (2019) require only single-dimensional numerical integrations when calculating the relative entropy reduction required for asynchronous batch BO. Unfortunately, as demonstrated in Section 5.4, complex interactions between each of the B fidelity evaluations $y_{\mathbf{z}_i}$ once conditioned on g^* (as present in the second term of (5.3.1)) prevents the approximation strategies employed by Takeno et al. (2019) being extended to the synchronous batch setting. In particular, a naive extension of Takeno et al. (2019)'s approach requires the prohibitively expensive numerical approximations of B-dimensional multivariate Gaussian cumulative density functions. In this work, we propose a novel approximation strategy for (5.3.1) completely free from numerical integrations, thus providing the first computationally light-weight information-theoretic acquisition function for synchronous batch BO.

5.3.4 Alternatives to Max-value Entropy Search

As discussed in Section 5.2, MES is not the only information-theoretic BO acquisition function and is a descendent of ES and PES. However, the original ES and PES, as-well as their extensions for batch BO (Hernández-Lobato et al., 2017) and multi-fidelity BO (Swersky et al., 2013; Zhang et al., 2017), seek to reduce the differential entropy of the d-dimensional maximiser $\mathbf{x}^*$ (rather than the single dimensional g^* targeted by MES). The calculation of this entropy is challenging, requiring sophisticated and expensive approximation strategies (see Section 5.6). As well as being substantially more expensive than MES, the reliance of ES and PES on coarse approximations means they provide less effective optimisation (Wang and Jegelka, 2017; Moss et al., 2020d; Takeno et al., 2019) (Chapter 3). Moreover, the approximation strategy employed by PES restricts its use to only Euclidean search spaces

Of course, attempts have been made to adapt other standard acquisition functions to multi-fidelity and batch BO, with examples including EI (Picheny et al., 2010; Chevalier and Ginsbourger, 2013; Marmin et al., 2015), UCB (Contal et al., 2013; Kandasamy et al., 2016, 2017) and KG (Wu and Frazier, 2016, 2018). However,

extensions of EI and UCB, although computationally cheap and often enjoying strong theoretical guarantees, are typically lacking in performance and even though KG-based methods can provide highly effective optimisation, their large computational cost restricts them to problems with function query costs large enough to overshadow very significant overheads (as demonstrated in Section 5.7). For batch BO, additional heuristic strategies have been developed that are compatible with any acquisition function, with the most popular and empirically successful being the Local Penalisation of González et al. (2016a) and DPP-based approach of Kathuria et al. (2016) (see Section 5.5 for a thorough discussion). Alternative but less performant heuristics include approaches based on Stein methods (Gong et al., 2019) and Thompson sampling (Kandasamy et al., 2018a).

5.4 A Novel Approximation of General-purpose Max-value Entropy Search

In this section, we present the key theoretical contribution of this work: a novel approximation of the GMES acquisition function proposed in Section 5.3. In particular, we formulate GMES in terms of a the Information Gain (IG) — a measure of entropy reduction often used when pruning decision tree classifiers (Raileanu and Stoffel, 2004) and when selecting features for statistical models of textual data (Yang and Pedersen, 1997). The remainder of the section then details a novel approximation strategy for the information gain based on simple well-known information-theoretic inequalities, before demonstrating explicitly how this IG approximation can be used to approximate the GMES acquisition function.

5.4.1 GMES as a Function of Information Gain

Recall our proposed GMES acquisition function (5.3.1), defined as the mutual information between a set of B fidelity evaluations and the objective function's maximum value g^*. As in the derivation of the original MES acquisition function (5.3.2), the

symmetric property of mutual information can be used to yield the expansion

$$\alpha_n^{\text{GMES}}(\{\mathbf{z}_i\}_{i=1}^B) \coloneqq H(\{y_{\mathbf{z}_i}\}_{i=1}^B|D_n) - \mathbb{E}_{g^*}\left[H(\{y_{\mathbf{z}_i}\}_{i=1}^B|D_n, g^*)|D_n\right]. \tag{5.4.1}$$

For ease of notation, we now define $A_i = y_{\mathbf{z}_i}$ and $C_i = g(\mathbf{x}_i)$ for each of the B candidate location-fidelity tuples $\mathbf{z}_i$, as well as the multivariate random variables $\mathbf{A} = (A_1, .., A_B)$ and $\mathbf{C} = (C_1, .., C_B)$. The information gain is then defined as the reduction in the entropy of $\mathbf{A}$ provided by knowing the maximal value of $C^* = \max \mathbf{C}$, i.e.

$$IG_n(\mathbf{A}, m|D_n) \coloneqq H(\mathbf{A}|D_n) - H(\mathbf{A}|C^* < m, D_n), \tag{5.4.2}$$

Comparing (5.4.1) and (5.4.2), it follows that the GMES acquisition function can be expressed in terms of IG as

$$\alpha_n^{\text{GMES}}(\{\mathbf{z}_i\}_{i=1}^B) = \mathbb{E}_{m\sim g^*}\left[IG_n(\mathbf{A}, m|D_n)\right].$$

We can now see that efficiently calculating (5.4.2) in general scenarios will allow principled max-value entropy search across a wide range of BO settings. This goal is therefore the focus of the remainder of this section.

5.4.2 Required Predictive Quantities

Before presenting our proposed approximation for IG, it is convenient to discuss the distributional forms induced by our surrogate GP model. All random variables are now assumed to be conditioned on the arbitrary information set D_n, which, alongside references to n, is henceforth dropped from our notation.

Courtesy of our GP surrogate model, we have that

$$\mathbf{A} \sim N(\boldsymbol{\mu}^A, \Sigma^A), \quad \mathbf{C} \sim N(\boldsymbol{\mu}^C, \Sigma^C) \quad \text{and} \quad \text{Corr}(A_i, C_i) = \rho_i,$$

for predictive means $\boldsymbol{\mu}^C, \boldsymbol{\mu}^A \in \mathbb{R}^B$, predictive covariances $\Sigma^C, \Sigma^A \in \mathbb{R}^{B\times B}$ and a vector of pairwise predictive correlations $\boldsymbol{\rho} \in \mathbb{R}^B$ (Rasmussen, 2004a; see Appendix C.1 for details on how these predictive quantities are easily extracted from a GP).

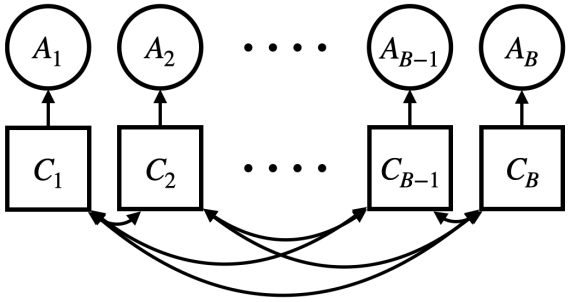


Figure 5.4.1: The considered dependency structure between the two set of random variables $\{A_1, .., A_B\}$ and $\{C_1, .., C_B\}$. Arrows denote the direction of dependence and latent variables are drawn in squares.

In addition to these well-known distributional forms, we can exploit the specific conditional structure of our GP surrogate model (which we describe below and summarise in Figure 5.4.1) to derive the conditional distribution of the random variable $\mathbf{A}$ given that $C^* < m$. In particular, our planned BO applications ensure that each A_j is conditionally independent of $\{C_i\}_{i \neq j}$ given C_j. This condition holds trivially for single-fidelity BO, where the difference between each A_i and C_i is just independent Gaussian noise. For multi-fidelity BO, this condition corresponds exactly to the *multi-fidelity Markov property* that is a key assumption underlying multi-fidelity GP modelling (Kennedy and O'Hagan, 2000; Le Gratiet and Garnier, 2014; Perdikaris et al., 2017). This is not a restrictive assumption, with O'Hagan (1998) showing that the multivariate Markov property holds for any GP surrogate model with a kernel that can be factorised into a product of kernels, one defined across the fidelity and one across the search space.

Under these dependence assumptions, Theorem 5.4.1 provides the distribution of $\mathbf{A}|C^* < m$ in closed-form, yielding a probability density function that, to the authors' knowledge, has not been previously considered in the statistics literature. Theorem 5.4.1 provides our first intuition for why the efficient calculation of the differential entropy $H(\mathbf{A}|C^* < m)$ is challenging, i.e. the presence of the multivariate Gaussian cumulative density in its probability density function.

Theorem 5.4.1 (Distribution of $\mathbf{A}$ given $C^* < m$)**.** *Consider two b-dimensional*

multivariate Gaussian random variables $\boldsymbol{A}$ *and* $\boldsymbol{C}$ *where* $\boldsymbol{C} \sim N(\boldsymbol{\mu}^C, \Sigma^C)$ *and each individual component of* $\boldsymbol{A}$ *is distributed as* $A_j \sim N(\mu_j^A, \Sigma_{j,j}^A)$. *Suppose further that each pair* $\{A_j, C_j\}$ *are jointly Gaussian with correlation* ρ_j, *and that each* A_j *is conditionally independent of* $\{C_i\}_{i \neq j}$ *given* C_j. *Define* $C^* = \max \boldsymbol{C}$. *Then the conditional density of* $\boldsymbol{A}$ *given that* $C^* < m$ *is given by*

$$\frac{1}{\mathbb{P}(C^* < m)} \phi_{\boldsymbol{Z}_1}(\boldsymbol{a}) \Phi_{\boldsymbol{Z}_2}(\boldsymbol{m}),$$

where $\boldsymbol{m} = (m, .., m) \in \mathbb{R}^B$ *and* $\phi_{\boldsymbol{Z}_1}$ *and* $\Phi_{\boldsymbol{Z}_2}$ *are the probability density and cumulative density functions for the multivariate Gaussian random variables*

$$\boldsymbol{Z}_1 \sim \boldsymbol{N}\left(\boldsymbol{\mu}^A, S + D\Sigma^C D\right) \quad \textit{and} \quad \boldsymbol{Z}_2 \sim \boldsymbol{N}\left(\boldsymbol{\mu}^C + \Sigma^{-1} D S^{-1}(\boldsymbol{a} - \boldsymbol{\mu}^A), \Sigma^{-1}\right),$$

where $\Sigma^A = D\Sigma^C D + S$ *for* D *and* S, *diagonal matrices with elements* $D_{j,j} = \rho_j \sqrt{\frac{\Sigma_j^A}{\Sigma_{j,j}^C}}$ *and* $S_{j,j} = (1 - \rho_j^2)\Sigma_j^A$, *and* $\Sigma = \left(\left(\Sigma^C\right)^{-1} + D S^{-1} D\right)$.

Proof. See Appendix C.2 □

Note that in the uni-variate case (i.e $B = 1$ and $C^* = C_1$), Theorem 5.4.1 collapses to the settings already considered when calculating MES and MUMBO in Section 5.3. Firstly, under the strong restriction that $C_1 = A_1$ (arising from BO without observation noise), $A_1 | C^* < m$ follows the well-known truncated Gaussian distribution, which can be seen directly from Theorem 5.4.1 by setting $\rho_j = 1$, $\mu_{j,j}^C = \mu_j^A$ and $\Sigma_{j,j}^C = \Sigma_j^A$. This truncated Gaussian has a simple analytical expression for its differential entropy which is exploited by standard MES. Secondly, if C_j and A_j are not perfectly correlated, we see that the density of Theorem 5.4.1 reduces to that of an Extended Skew Gaussian (ESG) distribution (Azzalini, 1985) as required for the MUMBO acquisition function (see Chapter 3). Although the differential entropy of an ESG has no closed-form expression (Arellano-Valle et al., 2013), we will later exploit the fact that its variance has an analytical form

$$\mathrm{Var}(A_j | C_j < m) = \Sigma_j^A \left(1 - \rho_j^2 \frac{\phi(\gamma_j(m))}{\Phi(\gamma_j(m))} \left[\gamma_j(m) + \frac{\phi(\gamma_j(m))}{\Phi(\gamma_j(m))}\right]\right), \tag{5.4.3}$$

where $\gamma_j(m) = (m - \mu_j^C)/\sqrt{\Sigma_{j,j}^C}$. We stress that, due to the complex interactions between each $A_j | C^* < m$, the joint distribution of $\mathbf{A} | C^* < m$ is not the multivariate ESG discussed by Azzalini and Valle (1996)).

5.4.3 Approximating Information Gain

We now present a lower bound $\text{IG}^{\text{APPROX}}$ for IG as Theorem 5.4.2. This bound is to be used as an approximation $IG \approx IG^{\text{Approx}}$. We stress that replacing the maximisation of an intractable quantity with the maximisation of a lower bound is a well established strategy in the ML literature, for example in variational inference (Blei et al., 2017).

Theorem 5.4.2 (A lower bound for information gain)**.** *Under the assumptions of Theorem 5.4.1, it holds that* $IG(\boldsymbol{A}, m) \geq IG^{Approx}(\boldsymbol{A}, m)$*, where*

$$IG^{Approx}(\boldsymbol{A}, m) \coloneqq \frac{1}{2}\log|R^A| - \frac{1}{2}\sum_{i=1}^{B}\log\left(1 - \rho_i^2\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\left[\gamma_i(m) + \frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\right]\right), \tag{5.4.4}$$

where $R^A \in \mathbb{R}^{B\times B}$ *is the predictive correlation matrix of* $\boldsymbol{A}$ *with entries* $R^A_{i,j} = \Sigma^A_{i,j}/\sqrt{\Sigma^A_{i,i}\Sigma^A_{j,j}}$.

Proof. Recall the definition of information gain $IG(\mathbf{A}, m) \coloneqq H(\mathbf{A}) - H(\mathbf{A}|C^* < m)$. The first term of IG is simply the differential entropy of a multivariate Gaussian distribution and so can be written in closed-form as $H(\mathbf{A}) = \frac{1}{2}\log\left[(2\pi e)^B\left|\Sigma_A\right|\right]$, where $\left|\Sigma_A\right|$ is the determinant of the $B \times B$ co-variance matrix of $\mathbf{A}$. Unfortunately calculating the second term of IG is significantly more complicated, with a closed form expression only in the limited cases discussed above.

We now build an analytical upper bound for $H(\mathbf{A}|C^* < m)$ by exploiting three common information-theoretic inequalities. As derived in Cover and Thomas (2012), we know that,

$$H(\mathbf{A}) \leq \sum_{i=1}^{B} H(A_i), \quad H(A_i|C^* < m) \leq H(A_i), \quad \text{and} \quad H(A_i) \leq \frac{1}{2}\log 2\pi\text{eVar}(A_i).$$

Applying the first two of these inequalities in sequence to $\mathbf{A}|C^* < m$ yields the upper-bound

$$H(\mathbf{A}|C^* < m) \leq \sum_{i=1}^{B} H(A_i|C_i < m).$$

Then, as we know that $A_j|C_j < m$ is an ESG (with a closed form expression for its variance), we can apply the third information-theoretic inequality to yield the

analytical upper bound

$$\begin{aligned} H(\mathbf{A}|C^* < m) \leq & \frac{1}{2}\sum_{i=1}^{B}\log(2\pi e\mathrm{Var}(A_i|C_i < m)) \\ = & \frac{1}{2}\sum_{i=1}^{B}\log 2\pi e\Sigma^A_{i,i}\left(1-\rho_j^2\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\left[\gamma_i(m)+\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\right]\right). \end{aligned}$$

Substituting this upper bound into (5.4.2), we have a lower bound for the information gain

$$\begin{aligned} IG^{\mathrm{Approx}}(\mathbf{A},m) \coloneqq & \frac{1}{2}\log\left|\Sigma^A\right| - \frac{1}{2}\sum_{i=1}^{B}\log\Sigma^A_{i,i}\left(1-\rho_j^2\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\left[\gamma_i(m)+\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\right]\right) \\ = & \frac{1}{2}\log\left|\Sigma^A\right| + \frac{1}{2}\log\prod_{i=1}^{b}\left(\Sigma^A_{i,i}\right)^{-1} - \\ & \frac{1}{2}\sum_{i=1}^{b}\log\left(1-\rho_j^2\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\left[\gamma_i(m)+\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\right]\right), \end{aligned}$$

which after defining the predictive correlation matrix R^A (with entries $R^A_{i,j} = \Sigma^A_{i,j}/\sqrt{\Sigma^A_{i,i}\Sigma^A_{j,j}}$) and noting that

$$\begin{aligned} \frac{1}{2}\log\left|\Sigma^A\right| + \frac{1}{2}\log\prod_{i=1}^{b}(\Sigma^A_{i,i})^{-1} &= \frac{1}{2}\log\left|\begin{pmatrix}\frac{1}{\sqrt{\Sigma^A_{1,1}}} & & 0\\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\Sigma^A_{b,b}}}\end{pmatrix}\Sigma^A\begin{pmatrix}\frac{1}{\sqrt{\Sigma^A_{1,1}}} & & 0\\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\Sigma^A_{b,b}}}\end{pmatrix}\right| \\ &= \frac{1}{2}\log\left|R^A\right|, \end{aligned}$$

provides the claimed expression. □

5.4.4 GIBBON: General-purpose Information-Based Bayesian OptimisatioN

We end this section with explicitly demonstrating how IG_{Approx} can be used to approximate the GMES acquisition function. Recall that GMES can be expressed in terms of IG as

$$\alpha_n^{\mathrm{GMES}}(\{\mathbf{z}_i\}_{i=1}^B) = \mathbb{E}_{m\sim g^*}\left[IG_n\left(\mathbf{A},m|D_n\right)\right].$$

We have already provided an approximation for IG and so all that remains to approximate GMES is to deal with its outer expectation over g^*. Following the arguments of Wang and Jegelka (2017), we build a Monte-Carlo approximation of this expectation using a Gumbel-based sampler. Therefore, given a set of sampled max-values $\mathcal{M} = \{m_1, .., m_M\}$ of $g^*|D_n$ and access to the predictive distributions

$$\{y_{\mathbf{z}_i}\}_{i=1}^B|D_n \sim N(\boldsymbol{\mu}^y, \Sigma^y), \quad \{g(\mathbf{x}_i)\}_{i=1}^B|D_n \sim N(\boldsymbol{\mu}^g, \Sigma^g) \quad \text{and} \quad \text{Corr}(y_{\mathbf{z}_i}, g(\mathbf{x}_i)|D_n) = \rho_i,$$

we can approximate GMES with

$$\alpha_n^{\text{GIBBON}}(\{\mathbf{z}\}_{i=1}^B) = \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}} IG^{APPROX}(\{y_{\mathbf{z}_1}, .., y_{\mathbf{z}_b}\}, m).$$

This construction is henceforth referred to as the General Information-Based Bayesian OptimisatioN (GIBBON) acquisition function and is defined as the closed-form expression in Definition 5.4.3 and demonstrated within a BO loop as Algorithm 2.

Definition 5.4.3 (The GIBBON acquisition function.)**.** *The GIBBON acquisition function is defined as*

$$\alpha_n^{GIBBON}(\{\boldsymbol{z}\}_{i=1}^B) = \frac{1}{2}\log|R| - \frac{1}{2|\mathcal{M}|}\sum_{m\in\mathcal{M}}\sum_{i=1}^{B}\log\left(1-\rho_i^2\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\left[\gamma_i(m)+\frac{\phi(\gamma_i(m))}{\Phi(\gamma_i(m))}\right]\right),$$

where R is the correlation matrix with elements $R_{i,j} = \Sigma^y_{i,j}/\sqrt{\Sigma^y_{i,i}\Sigma^y_{j,j}}$ and $\gamma_i(m) = \frac{m-\mu_i^g}{\sqrt{\Sigma^g_{i,i}}}$.

At first glance, GIBBON's analytical form looks complex. However, as GIBBON contains only simple algebraic operations, it can be easily calculated in just a few lines of code, unlike existing ES-based and PES-based acquisition functions and all existing extensions of MES (as discussed in depth in Section 5.6). An important practical consideration for GIBBON is that, for continuous search spaces, it has accessible gradients that can easily be derived from its analytical expression, allowing efficient inner-loop optimisation.

We end this section with a visual analysis of the accuracy of the GIBBON approximation. We consider a standard BO task with exact objective function evaluations

Algorithm 2 GIBBON for general-purpose BO tasks.

function GIBBON(Resource budget R, Batch size B, Gumbel sample size N)

Initialise $n \leftarrow 0$ and spent resource counter $r \leftarrow 0$

Propose initial design I

while $r \leq R$ **do**

Begin new iteration $n \leftarrow n + 1$

Fit GP model to collected evaluations D_n

Simulate N samples from $g^*|D_n$

Compute α_n^{GIBBON} as given by Definition 5.4.3

B locations $\{\mathbf{z}_i\}_{i=1}^B$ maximising $\frac{\alpha_n^{\text{GIBBON}}(\{\mathbf{z}_i\}_{i=1}^B)}{c(\{\mathbf{z}_i\}_{i=1}^B)}$

Evaluate new locations and collect evaluations $D_{n+1} \leftarrow D_n \bigcup \{(\mathbf{z}_i, y_i)\}_{i=1}^B$

Update spent budget $r \leftarrow r + c(\{\mathbf{z}_i\}_{i=1}^B)$

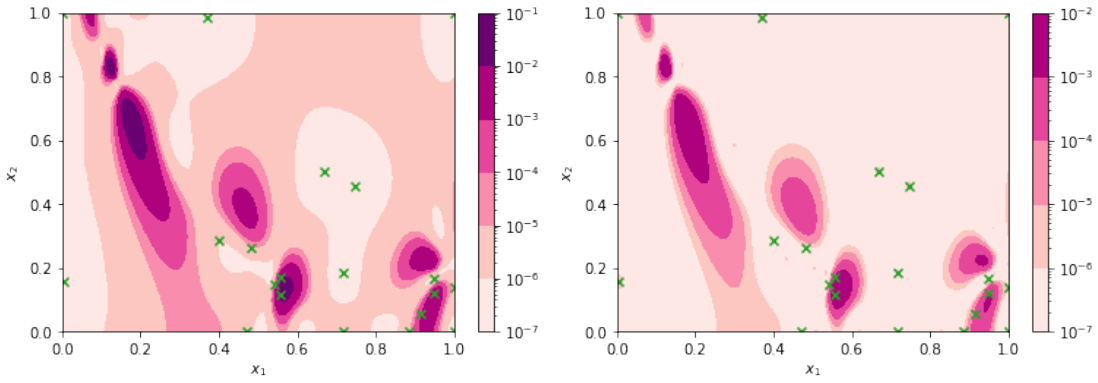
return Believed maximiser $\text{argmax}_{\mathbf{x} \in D_n} g(\mathbf{x})$

(i.e not multi-fidelity or batch optimisation) as, in this setting, the MES acquisition function provides an exact calculation of the entropy reductions. In Figure 5.4.2 we see that the approximation provided by GIBBON is very close to the ground truth provided by MES, with GIBBON and MES sharing modes and differing only in areas of the space that would never be selected by BO, i.e those locations with very low utility.

5.5 Relationship Between GIBBON and Heuristics for Batch Bayesian Optimisation

We now provide insights into the batch capabilities of our GIBBON acquisition function by drawing equivalences between GIBBON and two popular heuristics for batch BO — determinantal point processes (Section 5.5.1) and local penalisation (Section 5.5.2).

Recall that performing an iteration of BO requires the identification of optimal candidate points across the search space, i.e the maximisation of our acquisition function. For GIBBON, this *inner-loop* maximisation task corresponds to allocating a



(a) MES acquisition function surface (ground truth).

(b) GIBBON acquisition function surface.

Figure 5.4.2: Comparison of the MES and GIBBON acquisition functions for a two-dimensional BO task where MES can calculate entropy reductions exactly. The crosses denote the locations already queried by the BO routine. GIBBON provides a very close approximation of MES that reliably captures all its modes.

batch of B locations as

$$\{\mathbf{z}_{|D_n|+1}, .., \mathbf{z}_{|D_n|+B}\} = \underset{\mathbf{z}\in\mathcal{Z}}{\operatorname{argmax}}\, \alpha_n^{\text{GIBBON}}(\{\mathbf{z}_i\}_{i=1}^B).$$

Before introducing the two batch BO heuristics, it is convenient to provide an alternative expression for the GIBBON acquisition function. From Definition 5.4.3, we see that the GIBBON acquisition function for a candidate batch of B location-fidelity tuples can be decomposed into a sum of B GIBBON acquisition function evaluated separately for each tuple with an additional determinant term as

$$\alpha_n^{\text{GIBBON}}(\{\mathbf{z}\}_{i=1}^B) = \frac{1}{2}\log\left|R\right| + \sum_{i=1}^{B} \alpha_n^{\text{GIBBON}}(\mathbf{z}_i), \tag{5.5.1}$$

where R is the predictive correlation matrix of the batch. Note that the first term of this decomposition encourages diversity within the batch (achieving high values for points with low predictive correlation) whereas the second term ensures that evaluations are targeted in areas of the search space providing large amounts of information about g^*.

5.5.1 Relationship with Determinantal Point Processes

We can now interpret GIBBON in the context of a popular heuristic approach for batch design based on probabilistic models of repulsion known as Determinantal Point Processes (DPPs) (Kulesza et al., 2012). This comparison provides the first theoretical justification for choices of key DPP attributes which previously had to be chosen arbitrarily by practitioners.

DPPs provide a probability distribution over sets of points, such that sets of high-quality points (as measured by a quality function $q : \mathcal{X} \to \mathbb{R}$) with a diverse spread (as measured by a similarity kernel $s : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$) occur with high probability. More precisely, a particular set of points $\{\mathbf{x}_i\}_{i=1}^B$ occurs with probability.

$$\mathbb{P}(\{\mathbf{x}_j\}_{j=1}^B) \propto |L(\{\mathbf{x}_j\}_{j=1}^B)|, \tag{5.5.2}$$

where $L(\{\mathbf{x}_j\}_{j=1}^B)$ is a $b \times b$ matrix with elements $L_{i,j} = q(\mathbf{x}_i)q(\mathbf{x}_j)s(\mathbf{x}_i, \mathbf{x}_j)$.

Generating diverse but high-quality collections of points is exactly what we seek when allocating batches in BO problems. Unfortunately, a lack of understanding of how to choose appropriate quality functions and similarity kernels *a-priori* have previously limited the performance of DPP methods in BO, with existing applications requiring users to plug in arbitrary choices. The primary complication is that the relative scales of q and s trade-off the quality and diversity of batches, and so, for high-performance BO, these measures must be carefully chosen to complement (rather than dominate) each other. Consequently, the most common approach for using DPPs for BO is as part of *pure exploration* strategies, where the quality function is ignored ($q(\mathbf{x}) = 1$) and a DPP with a radial basis function kernel as its similarity measure is sampled to allocate a whole batch (Dodge et al., 2017), or to allocate the $B-1$ elements remaining after choosing an initial point through a standard sequential BO routine (Kathuria et al., 2016). Related approaches have also been used for high-dimensional BO (Wang et al., 2017b), where DPPs are used to sample a subset of the available search space dimensions. Note that these existing applications of DPPs to batch BO are limited in scope, supporting only single-fidelity problems over Euclidean search spaces, i.e those over which a standard similarity kernel can easily be defined.

We now explicitly show that our GIBBON acquisition function is equivalent to a DPP with specific choices of quality functions and similarity kernels. First define the exponential of our GIBBON acquisition function (with $B = 1$) as a quality function $q^{\mathrm{G}}(\mathbf{z}) = \exp\left(\alpha^{\mathrm{GIBBON}}(\mathbf{z})\right)$ and the predictive correlation (as specified by our GP surrogate model) as a similarity kernel $s^{\mathrm{G}}(\mathbf{z}_i, \mathbf{z}_j) = R_{i,j}$. Then, after defining $L^{\mathrm{G}}(\{\mathbf{z}_j\}_{j=1}^{B})$ as the matrix with elements $L^{\mathrm{G}}_{i,j} = q^{\mathrm{G}}(\mathbf{z}_i)q^{\mathrm{G}}(\mathbf{z}_j)s^{\mathrm{G}}(\mathbf{z}_i, \mathbf{z}_j)$, simple algebraic manipulations allow the batch GIBBON acquisition function (5.5.1) to be expressed as

$$\alpha_n^{\mathrm{GIBBON}}(\{\mathbf{z}_j\}_{j=1}^{B}) = \frac{1}{2}\log|L^{\mathrm{G}}|,$$

i.e the maximisation of our acquisition function corresponds to allocating the batch with maximal $|L^G|$, known as the *maximum a posteriori* (MAP) problem of DPPs. This is known to be NP-hard (Ko et al., 1995). However, the submodularity of DPPs ensures reasonable performance of greedy approximate solutions (as demonstrated by Gillenwater et al., 2012), explaining the observed effectiveness of a greedy batch-filling strategy when optimising our GIBBON acquisition function (see Section 5.7).

Recasting GIBBON as a DPP provides the first theoretical motivation for using DPPs for batch BO, with the particular choices of quality and similarity function arising from our information-theoretical derivation leading to significant improvements over existing DPP heuristics (Section 5.7). Moreover, we have greatly increased the generality of DPP-based BO, providing the first formulation that supports multi-fidelity and structured search spaces, or any other framework using a surrogate model where posterior correlation is easily accessible.

5.5.2 Relationship with Local Penalisation

Another class of popular heuristics for batch BO are those based on local penalisation (LP) (González et al., 2016a; Alvi et al., 2019). Rather than explicitly balancing the diversity and quality of batches as two additive contributions, LP methods apply a multiplicative scaling to down-weight an acquisition function around locations already present in the batch, thus ensuring the selection of a diverse set of points. We now show that GIBBON can be interpreted as a penalisation strategy and consequently, for the

first time in the literature, we make an explicit link between DPP- and LP-based BO routines. By recasting GIBBON as a local penalisation, we are able to derive a novel theoretically-justified penalisation function that outperforms existing LP methods.

For any choice of acquisition function $\alpha_n : \mathcal{X} \to \mathbb{R}$ taking positive values, an LP strategy greedily chooses the i^{th} element of the $n+1^{th}$ batch as

$$\mathbf{x}_{n+1,i} = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, \alpha_n(\mathbf{x}) \prod_{j=1}^{i-1} \psi(\mathbf{x}; \mathbf{x}_{n+1,j}),$$

where $\psi(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \to [0, 1]$ is a *penalisation function*. By requiring that $\psi(\mathbf{x}, \mathbf{x}')$ is a non-increasing function of $||\mathbf{x} - \mathbf{x}'||$, we ensure that penalisation is largest when considering $\mathbf{x}$ close to elements already present in the batch. The most popular penalisation function is the soft penaliser of González et al. (2016a)

$$\psi_{soft}(\mathbf{x}, \mathbf{x}') = \frac{1}{2}\text{erfc}(-z) \quad \text{for} \quad z = \frac{1}{\sqrt{\sigma_n^2(\mathbf{x}')}} \left(L||\mathbf{x} - \mathbf{x}'|| - g^* + \mu_n(\mathbf{x}')\right),$$

where erfc is the complementary error function and g^* is the current believed optimum. An important practical consideration of LP routines is that their performance is sensitive to predicting a Lipschitz constant L (i.e $|g(\mathbf{x}) - g(\mathbf{x}')| \leq L||\mathbf{x} - \mathbf{x}'|| \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$), for which point-estimates must be carefully extracted from previous function evaluations. Note that this Lipschitz constant can only be defined for Euclidean search spaces.

We now show that allocating batches by performing a greedy maximisation of GIBBON can be interpreted as an LP routine for specific choices of acquisition and penalisation functions. Define a re-scaled GIBBON acquisition function $\alpha_n^{scaled}(\mathbf{x}) = \left(e^{\alpha_n^{gibbon}(\mathbf{x})}\right)^2$ and a penaliser $\psi_{corr}(\mathbf{x}; \{\mathbf{x}_j\}_{j=1}^{i-1}) = \left|R(\{\mathbf{x}_j\}_{j=1}^{i-1} \cup \{\mathbf{x}\})\right|$ as the determinant of the batch's predictive correlation. After routine algebraic manipulations, we can see that allocating the i^{th} element of the $n+1^{th}$ batch according to a greedy maximisation of our GIBBON acquisition function is equivalently expressed as

$$\begin{aligned}\mathbf{x}_{n+1,i} &= \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, \alpha_n^{\text{GIBBON}}\left(\{\mathbf{x}\} \cup \{\mathbf{x}_{n+1,j}\}_{j=1}^{i-1}\right) \\ &= \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, \alpha_n^{scaled}(\mathbf{x}) \psi_{corr}(\mathbf{x}; \{\mathbf{x}_{n+1,j}\}_{j=1}^{i-1}),\end{aligned}$$

i.e. the predictive correlation term in GIBBON can be interpreted as a form of local penalisation. However, unlike ψ_{soft} and ψ_{hard}, ψ_{corr} does not require the estimation of L, instead just using the easily accessible predictive correlation of our GP. In fact the superior performance of our proposed approach over existing LP methods suggests that complicated penalisation functions are not needed at all.

5.6 The Computational Complexity of Information-theoretic Bayesian Optimisation

In this final section before our experimental results, we analyse the computational overhead incurred by GIBBON and compare with all other existing information-theoretic acquisition functions, many of which are included in our experimental results of Section 5.7. To the authors' knowledge, this analysis forms the first such comparative analysis across all information-theoretic BO. We discuss the complexity of the information-theoretic acquisition functions mentioned in Sections 5.2 and 5.3: Entropy Search (Hennig and Schuler, 2012, ES), Predictive Entropy Search (Hernández-Lobato et al., 2014, PES) and its extensions PPES (Hernández-Lobato et al., 2017) and MF-PES (Zhang et al., 2017), Max-value Entropy Search (Wang and Jegelka, 2017, MES) and its extensions MUMBO (Moss et al., 2020d) (Chapter 3) and MF-MES (Takeno et al., 2019), as well as the Fast Information-Theoretic BO of Ru et al. (2018, FITBO). Although MFMES was originally designed for asynchronous batch BO, Takeno et al. (2019) do discuss (in their Appendix D.4) an alteration that allows the support for the synchronous batch BO problems targeted in this work but with large computational cost. It is this variant of MFMES that we consider in this Section and for our experimental results (Section 5.7).

The computational complexity of BO routines is hard to measure exactly as we do not know *a-priori* how many evaluations are required to maximise the highly multi-modal acquisition function in each inner loop. However, there are two main contributors to the computational cost of information-theoretic BO that can be analysed: a one-off initialisation calculation required to 'prepare' the acquisition functions for each

Method	Noise?	Multi-Fidelity ?	Batch?	Non-Euclidean ?	Initialisation costs	Acquisition query costs
ES	✓	✓	×	✓	$n^2e^{2d} + e^{3d}$	n^2e^d
PES	✓	×	×	×	$n^2e^{2d} + (n+d)^3e^d$	$n^2 + (n+d)e^d$
PPES	✓	×	✓	×	$n^2e^{2d} + (n+d)^3e^d$	$B^2n^2 + (B^3 + n + d)e^d$
MF-PES	✓	✓	×	×	$n^2e^{2d} + (n+d)^3e^d$	$n^2 + (n+d)e^d$
FITBO	×	×	×	×	1	n^2
MES	×	×	×	✓	n^2e^d	n^2
MUMBO	✓	✓	×	✓	n^2e^d	n^2
MF-MES	✓	✓	✓	✓	n^2e^d	$B^2n^2 + B^3 + e^B$
GIBBON	✓	✓	✓	✓	n^2e^d	$B^2n^2 + B^3$

Table 5.6.1: Computational complexity of existing entropy-based acquisition functions. d denotes the dimensions of the search space, n is the number of observations already collection, and B denotes batch size. Complexity results are correct to highest order terms only and ignore constant factors.

separate BO step, and the costs of each acquisition function query required for the inner-loop maximisation. These two complexity contributions are presented in Table 5.6.1, alongside a summary of the type of extended BO problems supported by each acquisition function, i.e whether they permit noisy, multi-fidelity, batch observations or non-Euclidean search spaces. We now derive the stated complexity results for initialisation and acquisition function query costs.

5.6.1 Acquisition Function Initialisation Costs

All BO routines incur a computational cost at the start of each individual BO step through the fitting of the surrogate model. The primary contribution to the cost of fitting a GP surrogate model on n data points is an $n \times n$ matrix inversion, i.e an $O(n^3)$ computation. Extracting a single predictive mean or co-variance from this GP then costs $O(n)$ and $O(n^2)$, respectively. As the overhead of fitting the GP is incurred

across all BO routines, we leave out its contribution from our complexity analysis. We instead focus purely on the initialisation overheads specific to each information-theoretic acquisition function incurred when collecting sets of samples required for their approximation strategies. This set is reused for all acquisition function evaluations during a single inner-loop maximisation but re-sampled for each BO step.

All the samples required for information-theoretic acquisition functions can be separated into two distinct classes — those approximating single-dimensional quantities and those approximating quantities with the same dimensions as the search space. To paint a clear picture of computational cost, we consider BO problems with a search space of fixed dimension d and focus primarily on how the costs scale with respect to d, the batch size B, and the number of previously queried points n. Although all sample sizes are user-controllable, the efficiency of the resulting acquisition function depends sensitively on appropriately large sample sizes (as demonstrated for PES and MES by Wang and Jegelka (2017)). Therefore, sample sizes used when approximating d-dimensional quantities must grow exponentially as $O(e^d)$ in order to preserve approximation accuracy. In contrast, the sample sizes required for effective approximations of single dimensional quantities can be chosen independently of d and so are denoted as $O(1)$ in our complexity analysis.

As discussed in Section 5.3, MES-based acquisition functions (including GIBBON), uses a Gumbel sampler to access samples of the maximum value g^*. This sampler evaluates our GP surrogate model's posterior (at $O(n^2)$ cost) across $O(e^d)$ points to form a discretisation of the d-dimensional search space. Each of the required $O(1)$ samples of g^* (a single dimensional quantity) can then be extracted with $O(1)$ cost, yielding an overall complexity of $O(n^2e^d)$. As shown in Table 5.6.1, GIBBON's initialisation costs are substantially lower than those of the acquisition functions based on PES and ES. Only FITBO has a lower initialisation cost, however it has not seen widespread use as it supports only noiseless standard BO tasks and employs a complicated construction requiring linear approximations of non-central χ^2 process (operations not supported by GP libraries). For the ES and PES-based acquisition functions, which require samples from the d-dimensional objective function maximiser

$\mathbf{x}^*$, initialisation costs are substantial.

In ES, each sample of $\mathbf{x}^*$ is the maximum of a sample function drawn from the GP across an $O(e^d)$ discretisation of the search space. Simulating these function draws requires a one-off $O(e^{3d})$ computation for the Cholesky factor of the predictive co-variance matrix evaluated across the discretisation, as accessed with an $O(n^2)$ cost for each of its $O(e^{2d})$ elements. Consequently, the initialisation of ES incurs a sizeable $O(n^2e^{2d} + e^{3d})$ complexity scaling. PES also requires samples of $\mathbf{x}^*$ but instead maximises the sample draws from a finite feature approximation of the GP surrogate model (Rahimi and Recht, 2008), requiring just an $O(n^2)$ cost for each of the required $O(e^d)$ samples. However, unlike ES, PES incurs the additional cost of pre-computing an $n + d$-dimensional matrix inversion for each sample. Therefore, PES has a total initialisation cost of $O(n^2e^d + (n + d)^3e^d))$. Note that the finite feature approximation employed by PES and its variants is only rigorously defined for GPs with stationary kernels and Euclidean search spaces.

5.6.2 Acquisition Function Query Costs

We now discuss the computational complexity of each individual acquisition function query. As highlighted in Table 5.6.1, not only does the GIBBON acquisition function match the lowest query costs attained by any information-theoretic acquisition functions, but it is the first truly general acquisition function suitable for standard, stochastic, multi-fidelity and batch optimisation.

To calculate GIBBON and the other MES-based acquisition functions, we require the joint predictive distribution across B proposed batch locations. Accessing these B^2 predictive co-variance terms from a GP surrogate model and then taking its determinant cost $O(B^2n^2)$ and $O(B^3)$, respectively. Finally, GIBBON calculates an analytical expression for each of the $O(1)$ samples from g^* and across each of the batch elements, yielding an overall complexity of $O(B^2n^2 + B^3)$. MF-MES has a similar construction to GIBBON, but requires the additional calculation of a B-dimensional integral, each to be approximated numerically with $O(e^B)$ cost. Although in the non-batch setting, all MES-based acquisition functions have $O(n^2)$ cost, we stress

that FITBO, MUMBO and MF-MES all require numerical integration (a significant constant factor cost not picked up in our highest order complexity analysis), whereas GIBBON and standard MES do not. Consequently, the experiments of Section 5.7 show that GIBBON is substantially cheaper than MUMBO and MF-MES in practice.

The ES and PES-based acquisition functions incur a substantially larger query cost than GIBBON. Their primary computational bottleneck is the requirement of separate calculations for each of their $O(e^d)$ samples of $\mathbf{x}^*$. In ES, each evaluation requires an n^2 prediction from the GP for each location across a small $O(1)$-sized collection of points for each sampled $\mathbf{x}^*$. In contrast, PES requires only a single prediction from the GP but additional $O(n+d)$ manipulations for each of its $O(e^d)$ pre-computed kernel matrices. For batch BO, PPES requires B^2 GP predictions and a B^3 calculation to access the determinant of the batch's posterior co-variance, as well as an additional B^3 determinant calculations a for each pre-computed kernel matrix.

5.7 Experiments

We now finish this chapter with a comprehensive empirical evaluation of our GIBBON acquisition function. In particular, we consider batch (Section 5.7.1) and multi-fidelity (Section 5.7.2) synthetic benchmarks, as-well as well as a molecular design loop over a non-Euclidean and highly-structured search space (Section 5.7.3).

For clarity, all of our experiments follow a similar format. We run each of the considered BO methods across 50 random seeds, plotting mean performance and a single standard error. For batch algorithms, we count the evaluation of a batch as a single BO iteration. Suboptimality of the current believed optimum $\hat{\mathbf{x}}$ is measured by the regret $g(\mathbf{x}^*) - g(\hat{\mathbf{x}})$, where $\mathbf{x}^*$ is the true maximiser. For some experiments we also measure the time taken to choose the next query points (referred to as the optimisation overhead). This computational cost of performing BO includes fitting the GP surrogate model as well as initialising and maximising the acquisition function. All experiments reporting optimisation overheads were performed on a quad core Intel Xeon 2.30GHz processor.

Across all our experiments, we see the same general behaviour: GIBBON at least matches, and often exceeds, the performance of existing high-performance acquisition functions whilst incurring an order of magnitude lower computational overhead. Moreover, the breadth of our experiments showcases that GIBBON is truly a general-purpose acquisition function, forming the first computationally light-weight acquisition function suitable for standard BO extensions, batch high-cost string design problems and sophisticated synchronous batch multi-task BO frameworks.

Overall, the purpose of our experiments is to demonstrate how GIBBON performs relative to other BO acquisition functions, with a primary focus on existing MES-based approaches. For completeness, we also compare against a range of additional methods, chosen to reflect their popularity, code availability and suitability for the particular experiment. To this end, we compare GIBBON with all the acquisition functions supported by BoTorch and Emukit, as-well as our own implementations of the batch heuristics discussed in Section 5.5. We will introduce these competitors alongside the relevant empirical results. Unfortunately, the PES-based methods discussed in Section 5.6 do not have implementations in BoTorch or Emukit. Moreover, we could not find any other comparable maintained software implementations, likely due to demonstrably worse performance of PES than MES (as shown by Wang and Jegelka, 2017) and PES's difficult-to-implement subroutines (Section 5.6).

5.7.1 Standard and Batch Optimisation

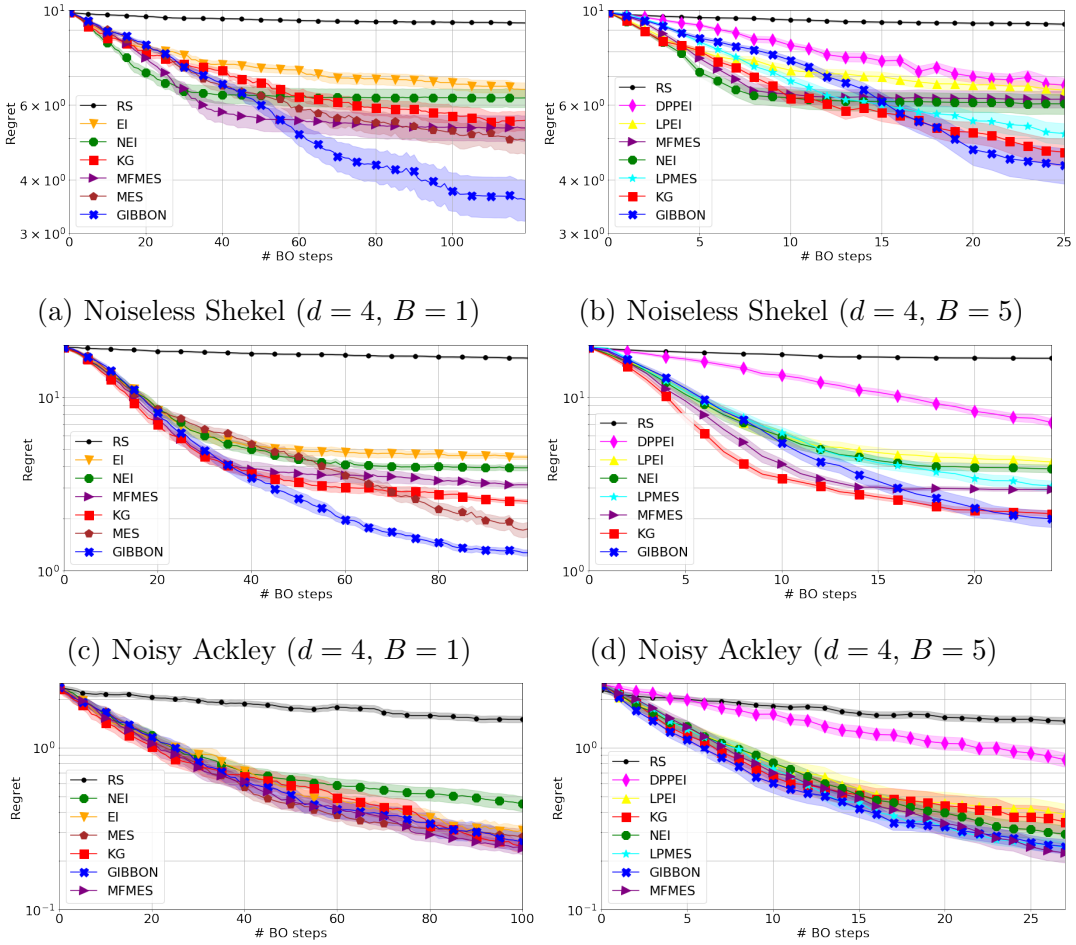
For our first set of experiments, we consider a set of synthetic functions provided with the BoTorch package. In particular, we recreate two of the experiments of Balandat et al. (2019) by maximising the Hartmann ($d = 6$) and Ackley functions ($d = 4$), each with observations perturbed by centred Gaussian noise with a variance of 0.25. In addition, we also consider the Shekel function ($d = 4$) under exact observations. For details of these synthetic functions, we refer readers to Appendix C.3.1. Following the setup of Balandat et al. (2019), we initialise all routines by evaluating $2d + 2$ random locations, refit our GP's kernel parameters after each BO step, and choose the current believed optimum $\mathbf{x}^*$ by maximising the posterior mean of the GP surrogate model.

For each experiment, we separately consider purely sequential BO ($B = 1$) and batch BO ($B = 5$), recording the evaluation of the whole batch as a single optimisation step.

As well as reporting the performance of GIBBON, MES and Expected Improvement (EI), we also ran the acquisition functions already supported in BoTorch, i.e Knowledge Gradient (KG), Noisy Expected Improvement (NEI) (Picheny et al., 2010), and MFMES (the multi-fidelity MES extension of Takeno et al. (2019),used here to support noisy observations). We stress that MFMES was designed to provide computationally light-weight asynchronous batch BO and we will see that its adaptation to synchronous problems (as implemented by BoTorch and discussed earlier in Sections 5.3 and 5.6) incurs a substantial computational overhead. For our batch problems, we also implemented BoTorch versions of Local Penalisation (LPEI) and the DPP heuristic (DPPEI) of Kathuria et al. (2016), both using EI as their base acquisition function (as considered by González et al. (2016a) and Kathuria et al. (2016)). In addition, we also provide local penalisation with an MES base acquisition function (LPMES), a combination not tested by González et al. (2016a) but found to be particularly effective in our experimentation. All MES-based acquisition functions (including GIBBON) use 5 max-values sampled from a Gumbel distribution fit to surrogate model predictions at $10,000 * d$ random locations and are re-sampled for each BO step. All other implementation parameters follow the BoTorch defaults.

For acquisition function maximisation we use BoTorch's gradient-based maximiser. However, as this inner-loop maximisation can be challenging since it corresponds to a highly multi-modal maximisation across a $B \times d$-dimensional space. Therefore most batch BO routines build batches greedily by breaking batch design into B separate d-dimensional maximisations. Consequently, for all approaches (including our GIBBON acquisition function) except KG , batches are constructed in this greedy manner with a maximisation budget of $10 * d$ random restarts for each element of the batch. Although KG is able to jointly allocate batches, its large computational cost restricted us to 20 restarts (the amount recommended by the BoTorch authors).

Across the three synthetic experiments (Figure 5.7.1) we see that GIBBON provides efficient high-precision optimisation, yielding small regret in competitively few iterations



(a) Noiseless Shekel ($d = 4$, $B = 1$)

(b) Noiseless Shekel ($d = 4$, $B = 5$)

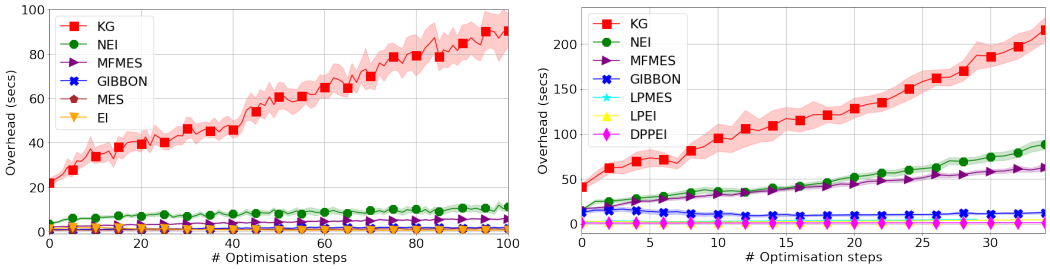
(c) Noisy Ackley ($d = 4$, $B = 1$)

(d) Noisy Ackley ($d = 4$, $B = 5$)

(e) Noisy Hartmann ($d = 6$, $B = 1$)

(f) Noisy Hartmann ($d = 6$, $B = 5$)

Figure 5.7.1: Optimisation of synthetic benchmark functions. GIBBON provides efficient and high-precision optimisation, matching or exceeding the performance of existing approaches.



(a) B=1 (b) B=5

Figure 5.7.2: The computational overheads incurred while optimising the Hartmann function. GIBBON's costs remains low throughout the optimisation, whereas the other high-performing batch acquisition functions costs increase dramatically as the optimisation progresses.

for both sequential and batch BO. Of particular note is the order of magnitude smaller overhead incurred by GIBBON over the other high-performing acquisition functions (NEI, KG and MFMES) as summarised in Table 5.7.1a (for $B = 1$) and Table 5.7.1b (for $B = 5$). In particular, batch KG incurs at least a 10 times larger overhead than GIBBON. Moreover, while the computational overhead of batch KG, MFMES and NEI increase substantially as the optimisation progresses, GIBBON's overhead remains the same (see Figure 5.7.2). Figure 5.7.3 confirms our earlier claim that GIBBON is indeed a high-performance yet computationally light-weight acquisition function, showing that GIBBON performs better than all competing acquisition functions while incurring a computational overhead only slightly worse than the simple but low-performance approaches. We were surprised to see that GIBBON is able to outperform standard MES in the noiseless optimisation task of Figure 5.7.1a, as it is for such scenarios that standard MES is exact. As GIBBON approximates MES, we expected it to perform strictly worse for this example. We delve deeper into this phenomenon in Appendix C.4.

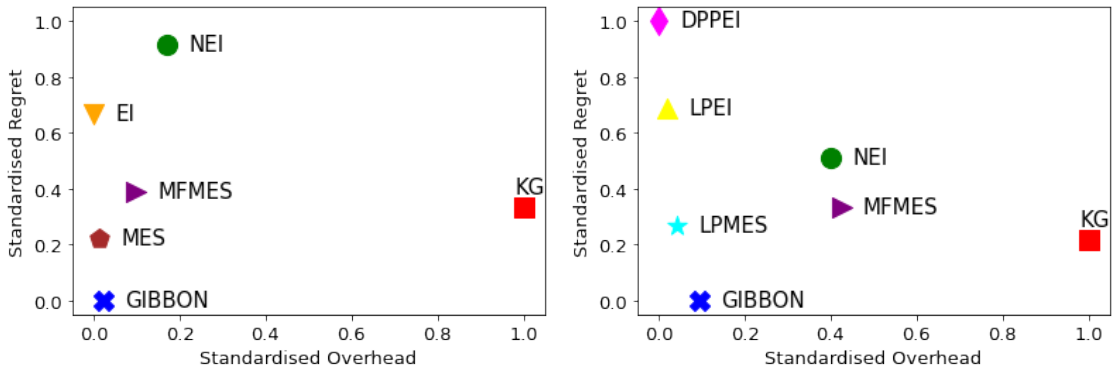
	Computational Overhead (seconds 1 d.p.)		
	Shekel (d=4)	Ackley (d=4)	Hartmann (d=6)
EI	0.2 (±0.0)	0.2 (±0.1)	0.8 (±0.1)
MES	0.5 (±0.1)	**0.5 (±0.0)**	1.0 (±0.1)
NEI	3.5 (±0.3)	3.0 (±0.2)	8.9 (±0.7)
MFMES	**3.0 (±0.4)**	0.7 (±0.1)	4.5 (±0.2)
KG	13.0 (±0.8)	22 (±1.0)	**66.6 (±4.6)**
GIBBON	**0.6 (±0.1)**	**0.8 (±0.1)**	**1.5 (±0.1)**

(a) Computational overheads for sequential BO ($B = 1$).

	Computational Overhead (seconds 1 d.p.)		
	Shekel (d=4)	Ackley (d=4)	Hartmann (d=6)
DPPEI	0.8 (±0.1)	0.8 (±0.0)	1.2 (±0.0)
LPEI	1.4 (±0.2)	2.3 (±0.1)	2.9 (±0.1)
LPMES	2.9 (±0.1)	3.3 (± 0.1)	3.5 (± 0.1)
NEI	21.3 (±1.8)	23.4 (±0.6)	43.0 (±2.6)
MFMES	24.4 (±2.3)	26.7 (±0.6)	**38.6 (±1.9)**
KG	**58.1 (±4.4)**	**53.0 (±3.1)**	103.4 (±6.2)
GIBBON	**5.0 (±0.5)**	**5.8 (±0.7)**	**13.3 (±1.3)**

(b) Computational overheads for batch BO ($B = 5$)

Table 5.7.1: Computational overheads for the synthetic benchmarks of Figure 5.7.1 averaged over the whole optimisation run. The two algorithms achieving lowest regret for each task are highlighted, demonstrating that GIBBON at least matches the overhead of other high-performing sequential acquisition functions and incurs a significantly lower overhead than other batch high-performing acquisition functions.



(a) B=1 (b) B=5

Figure 5.7.3: Comparison of the final regret achieved by each BO method with their computational overheads. Scores are standardised to sit within $[0, 1]$ and averaged across the three synthetic benchmark tasks. Lower scores on the x and y axis represent a smaller computational overheads and more effective optimisation, respectively.

5.7.2 Multi-fidelity Optimisation

We now turn to multi-fidelity optimisation, where the current state-of-the-art acquisition functions are the effectively equivalent MUMBO (Moss et al., 2020d) (Chapter 3) and MFMES (Takeno et al., 2019) acquisition functions. Moss et al. (2020d) demonstrates comprehensively that MUMBO outperforms a wide range of existing multi-fidelity acquisition functions, including the entropy search-based approach of Swersky et al. (2013), the upper-confidence bound variants of Kandasamy et al. (2016) and Kandasamy et al. (2017), as well as extensions of EI (Huang et al., 2006) and KG (Wu and Frazier, 2016). Therefore, to test GIBBON's multi-fidelity optimisation capabilities, it is sufficient to compare with MUMBO. To this end, we provide an implementation of GIBBON for the Emukit Python library and recreate exactly the synthetic experiments from Figure 3.5.1 of Chapter 3 (or Figure 2 of Moss et al. (2020d)). These experiments consider popular synthetic multi-fidelity benchmarks with discrete fidelity spaces consisting of between 2 and 4 fidelity levels (each with differing query costs) and search space dimensions ranging from 2 to 8 dimensions (see Appendix C.3.2 for the analytical forms of these synthetic benchmarks). In these

	Overhead for Multi-fidelity Optimisation (Seconds 1 d.p.)			
	Curin (d=4)	Hartmann (d=3)	Hartmann (d=6)	Borehole (d=8)
ES	16.6 (±0.7)	59.7 (±4.2)	229.8 (±15.3)	-
MUMBO	13.7 (±0.6)	18.6 (±1.0)	79.9 (±6.2)	51.5 (±7.5)
GIBBON	**4.0 (±0.2)**	**9.9 (±0.7)**	**50.2 (±4.0)**	**46.1 (±7.5)**

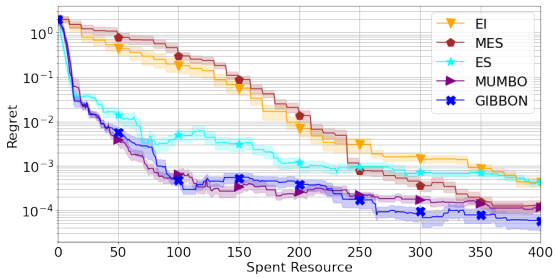
Table 5.7.2: Computational overheads of the multi-fidelity synthetic benchmarks of Figure 5.7.4. GIBBON enjoys the lowest overheads for all the tasks (as highlighted in bold), often less than half those of MUMBO.

experiments, we use the linear multi-fidelity GP model of Kennedy and O'Hagan (2000) as our surrogate model, initialise the GP with a random sample of $2 * d$ points queried across all fidelity levels, and fit the GP's kernel parameters to maximise model marginal likelihood after each BO step.

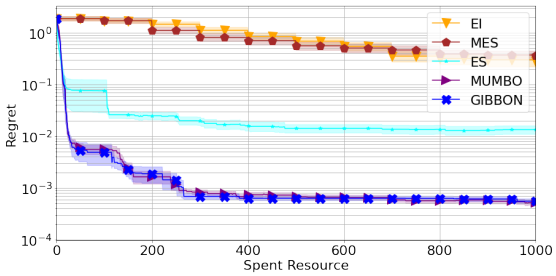
Figure 5.7.4 shows that GIBBON provides at least as effective optimisation as MUMBO and Table 5.7.2 shows that GIBBON has a significantly lighter computational overhead. To provide context for the high performance and low overhead of GIBBON we also present the performance of EI and MES when restricted to just querying the true objective function (i.e no access to low-fidelity observations) and the performance of the ES acquisition function, used to perform multi-fidelity optimisation by Swersky et al. (2013). Although the difference in overhead between MUMBO and GIBBON decreases as we consider higher-dimensional search spaces (primarily due to the growing cost of the Gumbel sampler used by both approaches), the difference in achieved regret increases in GIBBON's favour.

5.7.3 Batch Molecular Search

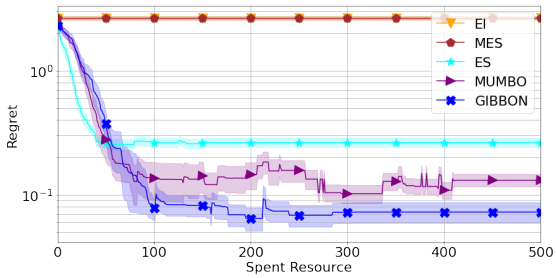
In Chapter 4, we applied BO to high-cost string design problems, considering, among other problems, the task of optimising over molecules. Such tasks are well-suited for BO, due to the high cost of evaluating candidate molecules via wet-lab experiments. Chapter 4 proposed a BO framework that fits a GP surrogate model to a popular string-



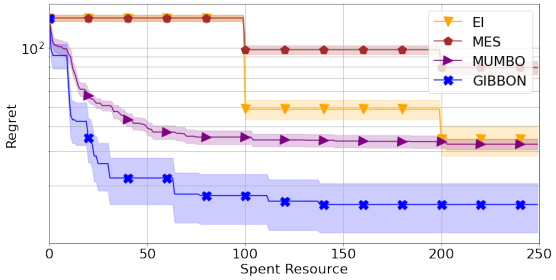
(a) Maximisation of the 2D Currin function (2 fidelity levels with evaluation costs 10 and 1).



(b) Minimisation of 3D Hartmann function (3 fidelity levels with evaluations costs 100, 10 and 1).



(c) Minimisation of 6D Hartmann function (4 fidelity levels).



(d) Maximisation of the 8D Borehole function (2 fidelity levels with evaluation costs 10 and 1).

Figure 5.7.4: GIBBON provides high-precision multi-fidelity optimisation with low computational overheads across a range of synthetic multi-fidelity benchmarks. Due to the high-cost of MTES, we were not able to run it on the higher-dimensional Borehole task. As is standard in multi-fidelity optimisation, the x-axis for these results measures the resources spent on function evaluations (rather than raw BO steps).

based representation of molecules known as SMILES strings (Anderson et al., 1987) through a string kernel GP (Beck et al., 2015). Standard EI arguments are then applied, yielding a highly effective strategy for searching large candidate set of molecules. One practical limitation of this framework, however, is the large computational cost of string kernels, as incurred for each prediction from the surrogate model GP. Consequently, the framework of Chapter 4 is limited to acquisition functions that require a small number of surrogate model predictions. Aside from GIBBON, our other considered high-performing batch acquisition functions (MFMES, NEI and KG) require many kernel evaluations for each acquisition function query and the low-cost approaches of DPPEI and LPEI are limited to only Euclidean search spaces. In contrast, GIBBON requires only B surrogate model predictions to measure the utility of a candidate batch and makes no assumptions on the properties of the search space. Therefore, GIBBON can be used to extend the framework of Chapter 4 to provide the first information-theoretical and the first batch approach for BO sting design. Batch design are particularly attractive for molecular search applications where it is common practice to synthesis collections of candidate molecules in parallel.

We now recreate the Zinc example considered by Moss et al. (2020b), where they explore a large collection of 250,000 molecules. The task is then to quickly find molecules that score highly according to a chemically-inspired metric, i.e. forming a proxy molecular design loop. At each BO step, we randomly sample $1,000$ molecules from which we (greedily) choose to evaluate the B molecules that maximise our GIBBON acquisition function. We fit our Gumbel sampler on this same sample, re-sampling both the max-values required for GIBBON and the considered $1,000$ molecules at the start of each BO step. We evaluate 20 randomly chosen molecules to initialise our GP and then allow BO to choose 100 further molecules, either one by one or as 20 batches of 5 molecules or 10 batches of 10 molecules. Figure 5.7.5 shows that even in the purely sequential case, GIBBON provides a modest boost in performance over EI (the acquisition function previously used by Moss et al. (2020b)). More importantly, Figure 5.7.5 also shows that GIBBON is able to provide effective batch optimisation over batches of size 5 and 10, therefore providing an extension of

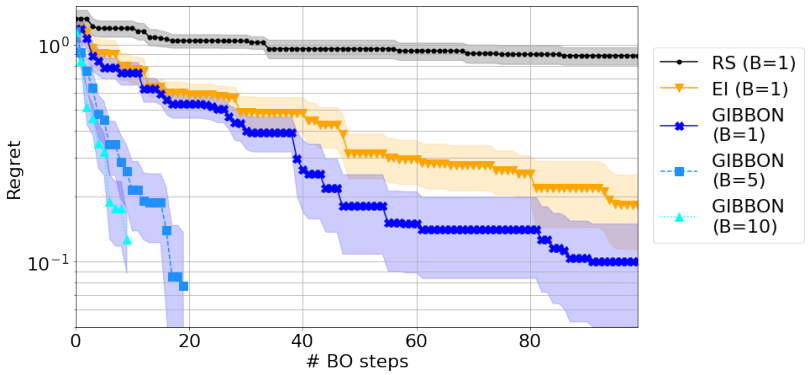


Figure 5.7.5: Exploring the Zinc database of molecules with GIBBON. In the purely sequential case, GIBBON finds higher-scoring molecules than EI. The batched GIBBON approaches reach roughly the same final regret after the same total number of 100 synthesised molecules, demonstrating that GIBBON is able to effectively leverage parallel synthesis resources.

Moss et al. (2020b)'s framework where parallel synthesising resources can be used to speed up the molecular search.

5.8 Conclusions and Future Work

We have presented GIBBON, a general-purpose acquisition function that extends max-value entropy search to provide computationally light-weight yet high performing optimisation for a wide range of BO problems. The efficiency of GIBBON relies on a novel information-theoretical approximation. Moreover, the derivation of this approximation allowed the exploration of the first explicit connection between information-theoretic search, determinantal point process and local penalisation, tying together large sections of the BO literature previously developed and analysed independently.

Not only does GIBBON provide competitive optimisation for common BO extensions like batch and multi-fidelity optimisation, but it forms the first high-performance batch acquisition function suitable for applying BO across highly-structured search spaces, as we demonstrated within a molecular design loop. BO for structured optimisation tasks is a fast growing frontier of the BO literature, with recent work tackling

BO for strings (Moss et al., 2020b; Swersky et al., 2020), combinatorial spaces (Deshwal et al., 2020) and spaces of neural network architectures (Kandasamy et al., 2018b). Therefore, we believe that GIBBON (and our flexible software implementation) will have substantial utility for the machine learning community.

As a final comment, we would like to point out that, although we have already shown GIBBON to have wide applicability, GIBBON can be readily applied to an even wider collection of BO problems. For example, GIBBON can be combined with MESMO (Belakaria et al., 2019), an extension of MES for multi-objective optimisation, to provide the first computationally light-weight acquisition function for batch multi-objective BO. Similarly, GIBBON can also provide a computationally light-weight approach for batch constrained optimisation by extending the MES-based approach of Belakaria et al. (2020). Finally, GIBBON can be used to improve the performance and reduce the computational cost of any framework relying on batch BO heuristics, for example in non-myopic BO (González et al., 2016b; Jiang et al., 2020).

Chapter 6

BOSH: Bayesian Optimisation by Sampling Hierarchically

Status: A condensed version of the work in this chapter was presented at the Workshop on Real World Experimental Design and Active Learning during *The International Conference on Machine Learning*, 2020.

6.1 Preface

In this chapter, we turn to a challenging multi-fidelity and batch BO framework inspired by BO problems with controllable observation noise. Existing deployments of Bayesian Optimisation (BO) in this setting, such as parameter tuning via cross validation and simulation optimisation, typically optimise an average of noisy realisations of the objective function as induced by a fixed collection of random seeds. However, disregarding the true objective function in this manner means that BO finds a high-precision optimum of the wrong function. To solve this problem, we propose *Bayesian Optimisation by Sampling Hierarchically* (BOSH), a novel BO routine pairing a hierarchical Gaussian process with a custom information-theoretic framework to generate a growing pool of seeds as the optimisation progresses. We demonstrate

that BOSH provides more efficient and higher-precision optimisation than standard BO across synthetic benchmarks, simulation optimisation, reinforcement learning and hyper-parameter tuning tasks. Two empirical studies also published during the PhD but outside the focus of this thesis provide additional compelling justification for the need for this BOSH framework when performing model selection and hyper-parameter tuning in natural language processing (Moss et al., 2018, 2019).

6.2 Introduction

Bayesian optimisation (BO) (Mockus, 2012) is a well-studied global optimisation routine for finding the optimiser

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, g(\mathbf{x}), \tag{6.2.1}$$

of a 'smooth' but expensive to evaluate function g over a compact domain $\mathcal{X} \in \mathbb{R}^d$. BO is particularly popular for problems where we have access to only noisy evaluations of g and has had many successful applications optimising high-cost stochastic functions including fine-tuning machine learning (ML) models (Snoek et al., 2012), optimising simulations in operational research (Kleijnen, 2009), and designing physical science experiments (Frazier and Wang, 2016).

For many stochastic optimisation tasks, it is commonplace to disregard the original objective function g and instead optimise the average of a collection of K specific realisations f_s, each generated by fixing a source of randomness through the specification of a random seed. Common examples include the K data partitions used to estimate ML model performance through K-fold cross validation (CV)(Kohavi, 1995) (see Figure 6.2.1) or considering K fixed initial conditions to create sample average approximations Kleywegt et al. (2002) for simulation optimisation or reinforcement learning. This small collection of seeds $S = \{s_1, ..s_K\}$ is typically randomly initialised, but then fixed for the remainder of the optimisation. We henceforth refer to S as an **evaluation strategy**, with its optimisation seeking

$$\mathbf{x}^*_S = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}} \left(\tilde{g}_S(\mathbf{x})\right), \tag{6.2.2}$$

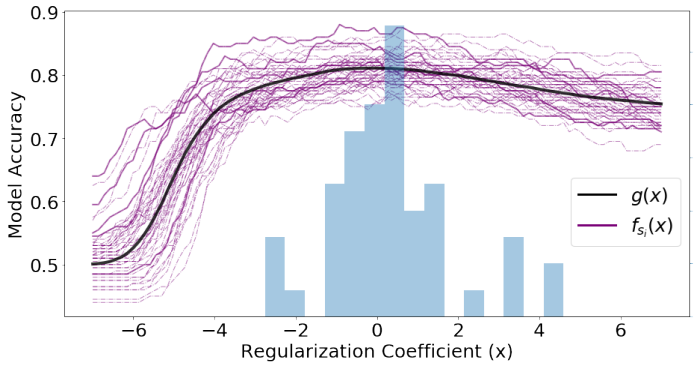


Figure 6.2.1: Estimated performance according to different train-test splits when tuning the amount of regularisation for a logistic regression classifier of sentiment in IMDB movie reviews. Individual performance estimates are plotted as purple lines (with five highlighted), and the score from a large test set (a proxy for true performance) is plotted in black. The histogram of chosen regularisation (performance curve maxima) shows many train-test splits choosing sub-optimal regularisation (-4% accuracy).

where $\tilde{g}_S(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} f_{s_i}(\mathbf{x})$.

Evaluations of $\tilde{g}_S(\mathbf{x})$ enjoy a substantial reduction in variance compared to a single stochastic evaluation of the true objective function $g(\mathbf{x})$. However, there is no guarantee that $\mathbf{x}^*_S \approx \mathbf{x}^*$, as $\mathbf{x}^*_S$ is a function of the randomly selected S. In fact, the expected suboptimality $\mathbb{E}_S[g(\mathbf{x}^*) - g(\mathbf{x}^*_S)]$ is a positive quantity decaying as $O(\frac{1}{K})$ where $K = |S|$ (as derived in Appendix D.1). Therefore, regardless of the sophistication of our optimisation routine, if K is set too low we cannot optimise g to an arbitrary precision level. However, as each individual evaluation of $\tilde{g}_S$ costs K times that of evaluating g, setting K too large wastes computational resources on unnecessarily expensive evaluations. Therefore, as demonstrated by Moss et al. (2018) for hyper-parameter tuning and Kim et al. (2015) for simulation optimisation, the efficiency and effectiveness of a fixed evaluation strategy crucially depends on the choice of K, taking into account evaluation variability and the desired optimisation precision.

In this work, we propose **BOSH** (Bayesian Optimisation by Sampling Hierarchically), an optimisation routine that sidesteps the complications of choosing a fixed function evaluation strategy by instead maintaining a pool of candidate seeds that grows as the optimisation progresses, providing efficient optimisation of the true objective function to arbitrary precision. By using a **Hierarchical Gaussian Process** (HGP) (Hensman et al., 2013) to model function evaluations for each random seed as separate perturbations of the latent 'true' object function, we can quantify the uncertainty in our current evaluation strategy. Consequently, BOSH is able to compare the utility of making further evaluations on each individual seed in the current pool with the benefit of considering a new seed (See Figure 6.2.2), avoiding over-fitting to a particular evaluation strategy or wasting resources evaluating poor choices across multiple seeds.

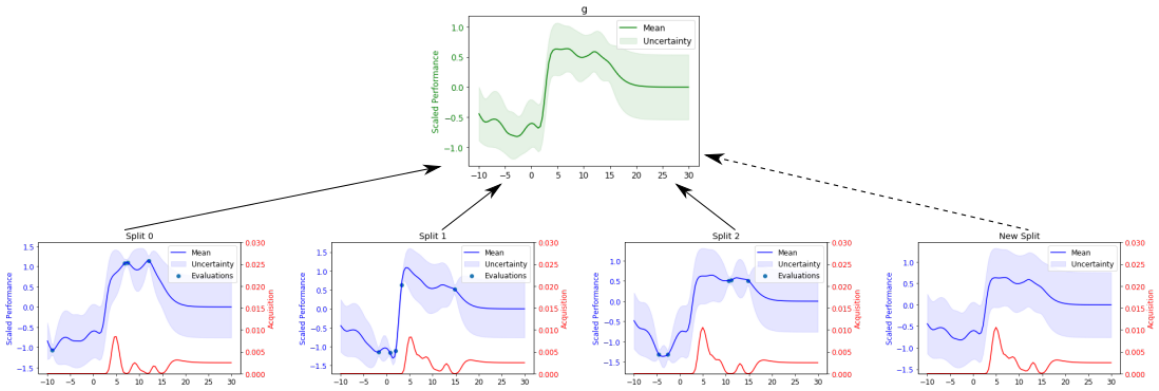


Figure 6.2.2: Tuning SVM regularisation on IMDB data using BOSH. We see the aggregation of knowledge from hyper-parameter evaluations spread among three train-test splits to produce predictions for the true accuracy g (in green) and belief about the behaviour of a potential new train-test split. The red lines show the predicted utility of making a new evaluation on each of the considered splits, showing lower values around hyper-parameters already evaluated on another split, and almost zero if already evaluated on that split.

6.3 Related Work

The idea of using low-cost approximations to speed up the optimisation of expensive functions is well-studied in the BO literature. Multi-task (MT) BO (Swersky et al., 2013; Poloczek et al., 2017) can provide efficient optimisation for problems with access to a finite collection of low-cost alternative functions holding some relationship with the true objective function. For problems where we can directly control the quality of objective function evaluations to produce a hierarchy of related functions, multi-fidelity (MF) BO (Wu and Frazier, 2018; McLeod et al., 2017; Kandasamy et al., 2016) can provide an additional improvement in optimisation efficiency. A particularly popular application of MF BO is hyper-parameter tuning, where routines can control the amount of data and training time used to train models (Klein et al., 2017a; Kandasamy et al., 2017; Falkner et al., 2018). Although these routines can provide incredibly fast rough hyper-parameter tuning, their reliance on very low-quality performance estimates typically limits their ability to perform high-precision optimisation (Section 6.6).

As BOSH controls the number of low-cost approximations it considers and can never query the true objective directly, BOSH does not fit into any current MT or MF frameworks. The closest existing idea to BOSH is FASTCV (Swersky et al., 2013), an extension of MT BO which, by evaluating the individual K train-test splits making up K-fold CV, speeds up hyper-parameter tuning under fixed evaluation strategies. However, FASTCV's intrinsic coregionalisation kernel (Bonilla et al., 2008) cannot predict performance on previously un-observed splits. Moreover, FASTCV chooses hyper-parameters and splits using a two-stage heuristic that has no clear extension to recommend batches of points.

A key component of BOSH is its careful choice of both x and the specific function realisation f_s used for each evaluation. However, when parallel computing resources allow the full evaluation of $\tilde{g}_S$ in the same time as f_s, there is no longer a computational saving from evaluating a single realisation. A batch deployment of BOSH can make better use of such parallel resources to simultaneously evaluate a batch of (x_i, s_i) pairs,

resulting in even greater gains. Although there are many heuristics for batch design within BO (Shah and Ghahramani, 2015; Wu and Frazier, 2016; González et al., 2016a; Hernández-Lobato et al., 2017; Kandasamy et al., 2018a), these approaches do not support the allocation of batches across a seed pool.

Recent work of Pearce et al. (2019) from the operational research literature address a similar problem but in a different way. They seek to reduce stochasticity in simulation optimisation problems by exploiting common random numbers and propose a framework similar to BOSH, where performance is measured according to individual random seeds. They deploy a complex model with multiple kernel parameters, making it very difficult to fit (requiring a Gibbs-style optimisation of the kernel hyper-parameters), and decisions are made using an extension of the knowledge gradient (KG) acquisition function (Frazier et al., 2008). Although enjoying theoretical guarantees, KG incurs significant computational overheads, and requires discretisation of the search space $\mathcal{X} \in \mathbb{R}^d$ and so has computational cost growing exponentially with d. In contrast, our proposed light-weight information-theoretic approach makes principled decisions with a linearly scaling cost and is able to recommend batches of points.

6.4 Bayesian Optimisation

By sequentially deciding where to make each evaluation as the optimisation progresses, BO can direct resources into promising areas to efficiently explore the search space and provide fast optimisation. BO's decisions are governed by two components - a **surrogate model** and an **acquisition function**.

Surrogate Model. Standard BO fits a Gaussian process (GP) (Rasmussen, 2004a) to the collected (potentially noisy) evaluations $D_n = \{(\mathbf{x}_i, y_i)\}_{i=1,..,n}$, where we assume $y_i = \tilde{g}_S(\mathbf{x}_i) + \epsilon_i$ for iid Gaussian noise ϵ_i with zero mean and variance σ^2. GPs provide non-parametric regression over all functions of a smoothness controlled by a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Crucially, our GP conditioned on D_n is still a GP, producing a Gaussian predictive distribution for $\tilde{g}_S(\mathbf{x})$ with mean $\mu_n(\mathbf{x}) = \mathbf{k}_n(\mathbf{x})^T(\mathbf{K}_n + \sigma^2 I)^{-1}\mathbf{y}_n$ and variance $\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_n(\mathbf{x})^T(\mathbf{K}_n + \sigma^2 I)^{-1}\mathbf{k}_n(\mathbf{x})$, where we define $\mathbf{K}_n =$

$[k(\mathbf{x}_i, \mathbf{x}_j)]_{(\mathbf{x}_i,\mathbf{x}_j)\in D_n}$, $\mathbf{k}_n(\mathbf{x}) = [k(\mathbf{x}_i, \mathbf{x})]_{\mathbf{x}_i\in D_n}$ and $\mathbf{y} = [y_i]_{i=1,..,n}$. Therefore we have a tractable predictive distribution quantifying our current belief about the shape of $\tilde{g}_S$ across the whole of $\mathcal{X}$.

Acquisition Function. The other crucial ingredient for BO is an acquisition function $\alpha_n(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$, measuring the utility of making a new evaluation at any given $\mathbf{x}$. In this work, we focus on highly successful information-theoretic acquisition functions which measure the amount of information provided about the location of the optimal $\mathbf{x}$ by evaluating a specific location in the search space and have achieved state-of-the-art performance across a variety of BO tasks (Wang and Jegelka, 2017; Klein et al., 2017a; Takeno et al., 2019). Information-theoretic arguments are particularly well suited to BOSH as they provide a clear measure of the utility of making an evaluation on a particular function realisation. Other multi-task BO acquisition functions, such as Swersky et al. (2013); Picheny et al. (2013); Lam et al. (2015); Kandasamy et al. (2016) lack such a clear notion of utility and consequently rely on two-stage heuristics that require tuning to a particular task. After making n evaluations, we next evaluate $\mathbf{x}_{n+1} = \text{argmax}_{\mathbf{x}\in\mathcal{X}}(\alpha_n(\mathbf{x}))$.

6.5 BOSH

The key difference between BOSH and existing BO routines is that instead of only modeling $\tilde{g}_S$ for a fixed evaluation strategy S, BOSH separately models realizations f_s for each seed $s \in S$. By assuming that each f_s is some perturbation of the true objective function g, we can fit a hierarchical model that learns the correlations between g and each f_s in our current seed pool S. Knowledge of this correlation structure provides information about the likely behavior of a yet unobserved seed. Therefore, BOSH can measure the benefit of expanding the current seed pool and make principled decisions about which seed to use for the next evaluation from the set of candidate seeds $S^* = S \cup \{s^*\}$ — either a seed from the current evaluation strategy S or generating a new seed s^* (to be absorbed into S for subsequent optimization steps). This allows BOSH to target g directly, instead of targeting $\tilde{g}_S$ for a fixed evaluation

strategy S.

6.5.1 The BOSH Surrogate Model

Hierarchical Gaussian Process. A natural model for modeling function realizations as perturbations of a true objective function is an HGP (Hensman et al., 2013), where the true objective function is modeled as a GP with an 'upper' kernel k_g, and the deviations to all the individual realizations f_s modeled by another GP with a 'lower' kernel k_f. This structure is equivalently understood as modeling each f_s as separate GPs with a shared mean function g, i.e.

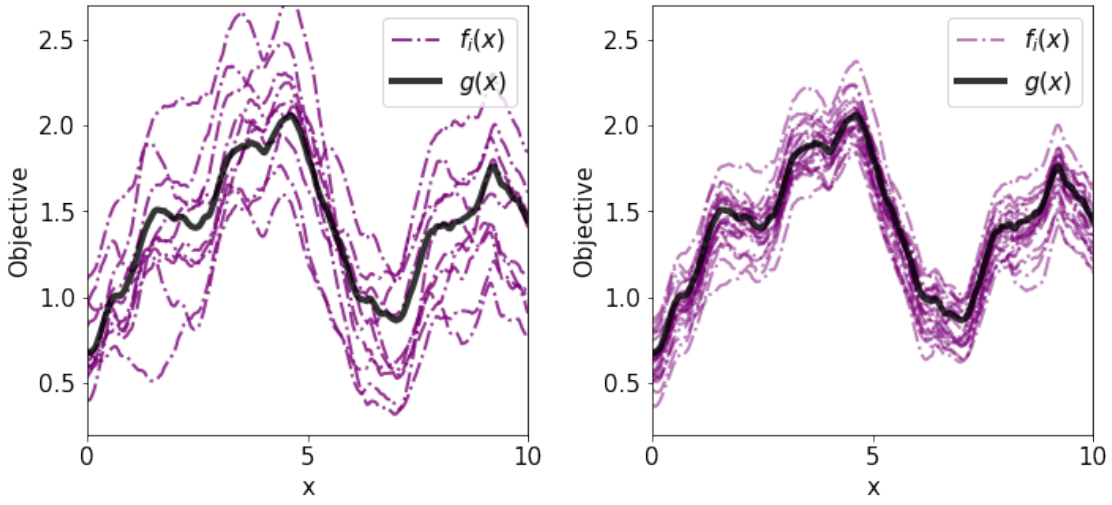
$$\begin{aligned} g &\sim \mathcal{GP}(0, k_g) \\ f_s &\sim \mathcal{GP}(g, k_f) \\ y_i &= f_{s_i}(\mathbf{x}_i) + \epsilon_i, \end{aligned} \tag{6.5.1}$$

where y_i is the evaluation of f_{s_i} at $\mathbf{x}_i$ and $\epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$. This formulation induces the following prior covariance:

$$\begin{aligned} Cov(f_s(\mathbf{x}), f_{s'}(\mathbf{x}'))) &= k_g(\mathbf{x}, \mathbf{x}') + \mathbb{I}_{s=s'} k_f(\mathbf{x}, \mathbf{x}') \\ Cov(f_s(\mathbf{x}), g(\mathbf{x}')) &= k_g(\mathbf{x}, \mathbf{x}'), \end{aligned} \tag{6.5.2}$$

where $\mathbb{I}$ is an indicator function. Samples from this prior are provided in Figure 6.5.1.

Predictive Distribution. Once conditioned on the collected evaluations D_n, we can predict evaluations at $y_s(\mathbf{x})|D_n$ (for any $s \in S^*$) and $g(\mathbf{x})|D_n$ across any location $\mathbf{x} \in \mathcal{X}$ (see Figure 6.2.2). In particular, for any $s \in S^*$ and any $\mathbf{x} \in \mathcal{X}$, our HGP provides a bi-variate Gaussian joint predictive distribution for $y_s(\mathbf{x})|D_n$ and $g(\mathbf{x})|D_n$ - the only quantities required to calculate our chosen acquisition function (Section 6.5.2). We provide closed-form expressions for these quantities in Appendix D.2. The computational cost of predictions is equivalent to a standard GP, with a cost dominated by an $O(n^3)$ matrix inversion during the n^{th} BO step.



(a) (b)

Figure 6.5.1: Simulations from two HGPs, demonstrating their capacity for modelling scenarios like Figure 6.2.1. The purple lines show 25 sampled $f_s(x)$ and the true objective $g(x)$ is plotted in black. Tiles (a) and (b) demonstrate lower kernels with large and small lower kernel flexibility respectively.

6.5.2 The BOSH Acquisition Function

Information-theoretic BO. An intuitive search strategy is to make evaluations that maximally reduce our uncertainty in the maximiser $\mathbf{x}^*$ of the true objective function. As is common in the BO literature (Hennig and Schuler, 2012; Hernández-Lobato et al., 2014), we measure our uncertainty in terms of differential entropy (see Cover and Thomas, 2012, for an introduction to information theory). In particular, following the arguments of Wang and Jegelka (2017), Moss et al. (2020d) (Chapter 3) and Takeno et al. (2019), we seek to reduce the differential entropy of our current belief about the maximum value of the objective function $g^* = g(\mathbf{x}^*)$, given by $H(g^*) = -\mathbb{E}_{g\sim p_{g^*}}(\log p_{g^*}(g))$, where p_{g*} is the probability density function of $g^*|D_n$ according to our current HGP model. The reduction in entropy of g^* provided by a single (possibly noisy) evaluation $y_s(\mathbf{x})$ is measured as their mutual information I, defined as

$$I(y_s(\mathbf{x}); g^*|D_n) \coloneqq H(y_s(\mathbf{x})|D_n) - \mathbb{E}_{g^*|D_n}\left[H(y_s(\mathbf{x})|g^*, D_n)\right]. \tag{6.5.3}$$

Performing principled information-theoretic BO corresponds to defining an acquisition function $\alpha_n(\mathbf{x}, s)$ as the mutual information (6.5.3) and choosing to make the $n+1^{th}$ evaluation at $\mathbf{z}_{n+1} = \text{argmax}_{\mathbf{z}\in\mathcal{Z}}\, \alpha_n(\mathbf{z})$ where, for ease of notation, we define $\mathcal{Z} = \mathcal{X}\times S^*$ and $\mathbf{z} = (\mathbf{x}, s)$.

In practice, distributed computing resources can be used to calculate the evaluations forming a particular evaluation strategy in parallel, for example the K model fits required for a single K-fold CV estimate. Therefore, we extend the information-theoretic framework (6.5.3) to recommend sets of B evaluations $\{y_{s_j}(\mathbf{x}_j)\}_{j=1}^B$ at each iteration, i.e. we allocate our batch of points to maximise the quantity of information provided by a batch about the unknown g^*, resulting in acquisition function $\alpha_n(\{(\mathbf{x}_j, s_j)\}_{j=1}^B)$ equal to

$$I(\{y_{s_j}(\mathbf{x}_j)\}_{j=1}^B; g^*|D_n) \coloneqq H(\{y_{s_j}(\mathbf{x}_j)\}_{j=1}^B|D_n) - \mathbb{E}_{g^*|D_n}\left[H(\{y_{s_j}(\mathbf{x}_j)\}_{j=1}^B|g^*, D_n)\right]. \tag{6.5.4}$$

Performing principled information-theoretic batch BO corresponds to allocating our $n+1^{th}$ batch by solving

$$(\mathbf{z}_{n+1,1}, ..\mathbf{z}_{n+1,B}) = \underset{(\mathbf{z}_1,..,\mathbf{z}_B)\in\mathcal{Z}^B}{\text{argmax}}\ \alpha_n(\{\mathbf{z}_j\}_{j=1}^B), \tag{6.5.5}$$

where $\mathbf{z}_{n,i}$ is be i^{th} element of the n^{th} batch.

GIBBON Approximation Unfortunately, closed-form expressions for the distribution of $g^*|D_n$ or the differential entropy of $y_s(\mathbf{x}|g^*, D_n)$ do not exist. Therefore to implement information-theoretic BO, the second term of (6.5.3) and (6.5.4) must be approximated. Our GIBBON acquisition function of Chapter 5, provides one such approximation for (6.5.3) suitable for BOSH. We demonstrate this acquisition function within BOSH in Figure 6.2.2.

Acquisition Maximisation. Even though we can now easily calculate the utility of evaluating any given candidate batch, it still remains to determine the optimal elements for a batch. We found that allocating a whole batch by naively performing the $B \times d$-dimensional maximisation of the multi-modal acquisition (6.5.5) posed too great computational challenge for even low dimensional search spaces and batch sizes. We, therefore, propose using a greedy strategy to fill the batch, breaking batch design

into B separate sequential decisions. Therefore, our batch BOSH formulation comes with only the insignificant overhead of calculating between batch correlations over B sequential BOSH steps. Algorithm 3 shows a high-level summary of BOSH.

Algorithm 3 BO by Sampling Hierarchically: BOSH

Input: Number of steps N, batch size B

Initialise $n \leftarrow 0$

Collect initial design D_0 (Section 6.6)

while $n \leq N$ **do**

Begin new iteration $n \leftarrow n + 1$

Fit a HGP to observations D_{n-1}

for $i = 1, .., B$ **do**

Prepare $\alpha_{n,i}(\mathbf{z}) \leftarrow \alpha_n^{GIBBON}\left(\{\mathbf{z}_{n,j}\}_{j=1}^{i-1} \cup \{\mathbf{z}\}\right)$

Find $\mathbf{z}_{n,i} \leftarrow$ maximiser of $\alpha_{n,i}$

Query batch $y_{n,i} \leftarrow f_{s_{n,i}}(\mathbf{x}_{n,i})$ for $i \in \{1, .., B\}$

Update $D_n \leftarrow D_{n-1} \bigcup \{(\mathbf{x}_{n,i}, s_{n,i}, y_{n,i})\}_{i=1}^{B}$

return $\text{argmax}_{\mathbf{x} \in \{\mathbf{x}_1, .., \mathbf{x}_N\}} (g(\mathbf{x}) | D_N)$

6.6 Experiments

We now demonstrate the performance of BOSH across a wide range of stochastic optimisation tasks from different fields. We provide full details for each experiment in Appendix D.3.

General experimental framework. For clarity, all of our experiments follow a similar format. We compare the performance of BOSH when producing batches of size B against the performance of existing BO routines based on fixed evaluation strategies of size B (denoted with the suffix 'fixed B') and existing routines based on random evaluation strategies of size B (denoted with the suffix 'rand B'), where we query objective functions B times but do not fix seeds between BO steps. Performance of our algorithms is measured by the number of individual optimisation steps required

to reach a certain incumbent performance, including resources spent evaluating the initial design. For a fair reflection of the increasing availability of parallel computing resources, we record the evaluation of the whole batch (or evaluation strategy) as a single optimisation step. Each method is run across 100 random seeds (except the more expensive RL task which was run 50 times) and we plot mean performance with a single standard error. Suboptimality of the current believed optimum $\hat{\mathbf{x}}$ is measured by the simple regret $g(\mathbf{x}^*_{obs}) - g(\hat{\mathbf{x}})$, where $\mathbf{x}^*_{obs}$ is the highest scoring parameters found by any of our considered routines on that problem.

Considered BO routines We consider standard BO using three well-known acquisition functions: expected improvement (EI) (Mockus et al., 1978), max-value entropy search (MES) (Wang and Jegelka, 2017), and knowledge-gradient (KG) (Frazier et al., 2008). Expected improvement is widely regarded as the base-line for BO, max-value entropy search is often seen as the *state-of-the-art* computationally-light acquisition function, and, although incurring significant computational overheads, KG has high performance and theoretical guarantees. We also consider the FASTCV of Swersky et al. (2013) (as presented in Section 6.3) as an approach to speed up optimisation under a fixed evaluation strategy. For our hyper-parameter tuning experiments we further consider the hyper-parameter specific BO routine of FABOLAS (Klein et al., 2017a) (with code provided in (Klein et al., 2017b)). As FASTCV and FABOLAS do not support Batch decisions, we present their performance only for experiments where $B = 1$.

For experiments where $B > 1$, we must disentangle the benefits of considering an adaptive evaluation strategy with the efficiency improvements naturally provided by allowing batch recommendations. To this end, we compare each run of batch BOSH with popular BO heuristics allocating batches of size B across a fixed evaluation strategy of size one. Batches are allocated using the popular heuristic of locally penalised (LP) EI (González et al., 2016a), as-well as the GIBBON acquisition function of chapter 5 (referred to in the subsequent experiments as Batch DPP). Unfortunately, Pearce et al. (2019) have yet to provide code for their batch KG approach, so we have been unable to provide direct comparisons. However, standard BO under the KG acquisition

function had very similar performance but larger overheads to standard BO with MES (upon which BOSH is based). We do not consider scenarios where we simultaneously deploy both batch BO and full evaluation strategies (e.g. a batch of 5 different x values, each evaluated using 5-fold CV). Fitting 25 models in parallel is beyond the resources of most ML researchers, necessitating a choice between batch BO or an evaluation strategy of size larger than 1.

GP Kernels All our GPs use Matérn 5/2 kernels (Matérn, 1960), resulting in $d+2$ unknown kernel parameters for standard BO and $d+2+B^2$ for FASTCV (which requires an additional $B \times B$ between-seed correlation matrix). For BOSH, rather than using separate lower and upper kernels for our HGP, we found that tying length-scales between each kernel greatly improved the stability of the HGP. Moreover, adding a bias term to the lower kernel sped up early-stage optimisation by conveniently modelling realisations of the objective function differing significantly in value but not in shape (i.e approximate translations). Therefore, our HGP has $d+4$ kernel parameters. We fit kernel parameters after each BO step to maximise the model marginal likelihood across all presented approaches except for KG, where the only available implementation (Balandat et al., 2019) follows the arguments of Snoek et al. (2012) and integrates kernel parameters over specially chosen hyper-priors. Although parameter integration can stabilise the early stages of BO for some tasks, it incurs significant overheads and would have harmed the light computational nature of our proposed acquisition function.

Initialisation costs Before beginning any BO routine, we must collect an initialisation of points to fit the surrogate model. To allow stable maximisation of the marginal likelihood, it is common to initialise with at least as many evaluations as unknown kernel parameters (to guarantee identifiability). For standard BO, this corresponds to $d+3$ evaluations of the chosen evaluation strategy (i.e requiring $B*(d+3)$ individual function evaluations). Similarly, we allowed BOSH $d+5$ evaluations for each of the seeds in an initial seed pool with two elements (i.e $2*(d+5)$ evaluations in total). Reliable initialisation of FASTCV's $B \times B$ correlation matrix (of which its performance was very sensitive) required at least $d+3$ evaluations for each of its B

considered seeds. Therefore, as well as providing improved efficiency and precision once optimisation begins, BOSH's ability to model only as many individual seeds as required allows significantly lower initialisation costs.

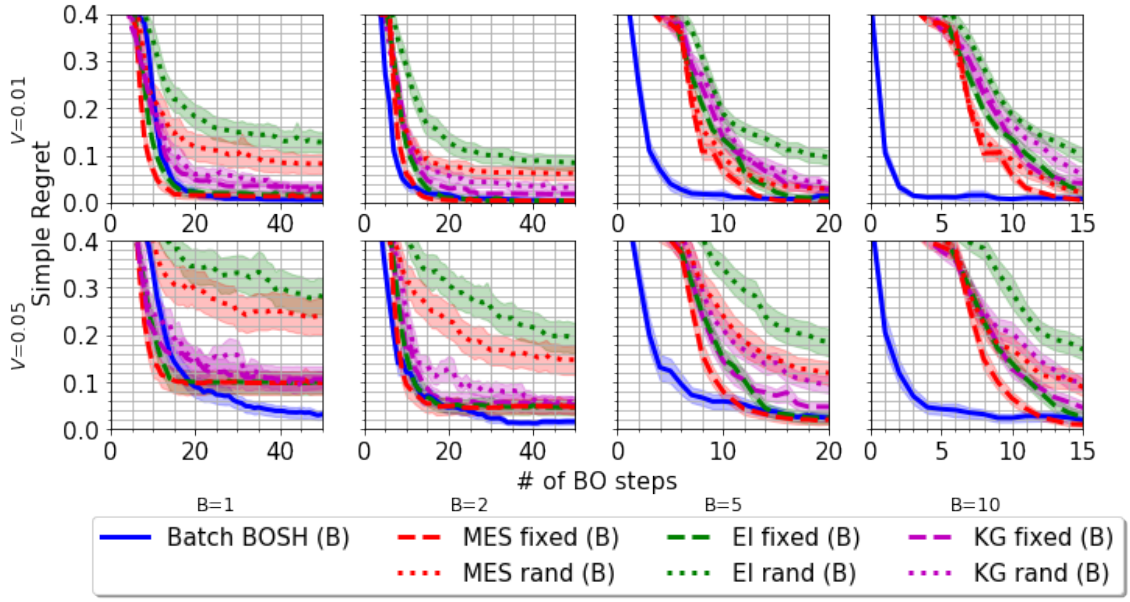


Figure 6.6.1: Optimisation of the upper function of the two HGPs presented in Figure 6.5.1.

6.6.1 Optimisation of Synthetic Objective

First, we simulate data directly from an HGP to investigate exactly when BOSH provides more efficient and reliable optimisation than standard BO. We seek to find the maximum of $g(x)$ (as plotted in Figure 6.5.1) by querying the perturbed curves f_s generated from two different lower kernels, one with a small lower kernel variance (denoted as V) causing low between-realisation variability, and another with a larger variance causing high between-realisation variability.

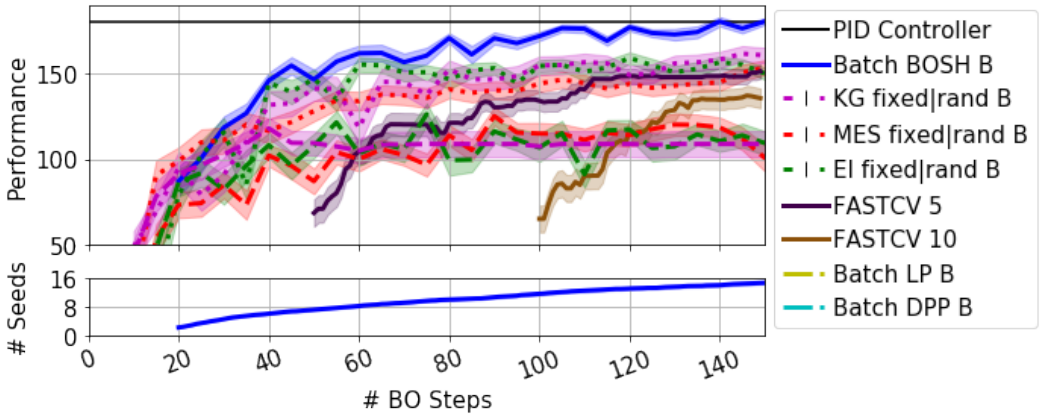
Figure 6.6.1 demonstrates the general behaviour that we see across all our experiments: using fixed evaluation strategies can provide either precise or efficient optimisation of stochastic objective functions, not both. BOSH's adaptive evaluation strategy is able to provide both efficient and precise optimisation. Although standard BO under large evaluation strategies can be as precise optimisation as BOSH, the

reliance on expensive evaluations early in the optimisation and the substantial initialisation costs mean that they provide significantly slower optimisation (a performance gap growing with the size of evaluation variance). Reassuringly, using fixed evaluation strategies does provide improved performance over making completely random queries.

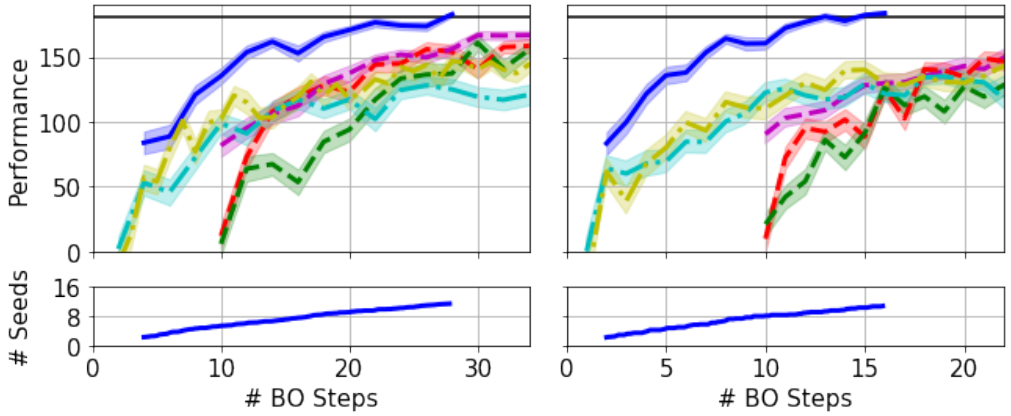
6.6.2 Reinforcement Learning

We now consider a challenging seven-dimensional stochastic optimisation test-case for BOSH. We wish to fine-tune a controller for a well-studied reinforcement learning problem, where we must guide a lunar lander across a randomly initialised space to its landing zone by controlling its thrusters (as provided in the OpenAI Gym[1]). Our controller is parameterised by seven unknown constants and a particular configuration can be tested by running a single (or B) randomly generated scenarios. To construct a more challenging optimisation task, we randomly vary the initial location and velocities of the lander as well as the location of the landing site across scenarios (all controlled by the same random seed). We seek to outperform OpenAI's hard-coded controller (denoted as the PID controller) according to a 'true' performance measured over a set of 100 fixed initial conditions, using as few simulation runs as possible (Figure 6.6.2). In this task there is substantial variation in performance across different random seeds meaning that optimising the controller over a small collection of initialisation fails to provide good 'true' performance. In fact, basing optimisation on a single fixed initialisation is comfortably outperformed by using no evaluation strategy at all. Although batch BO on single seeds provides fast initial optimisation, achieving reasonable precision requires much larger evaluation strategies. By adaptively considering up to 15 different seeds, BOSH is able to provide fast and precise optimisation to quickly match the performance of the PID controller. In contrast, FASTCV's need to initialise and then update the large between-seed correlation matrix severely hampers optimisation efficiency.

[1] *https://gym.openai.com/envs/LunarLander-v2/*



(a) B=1



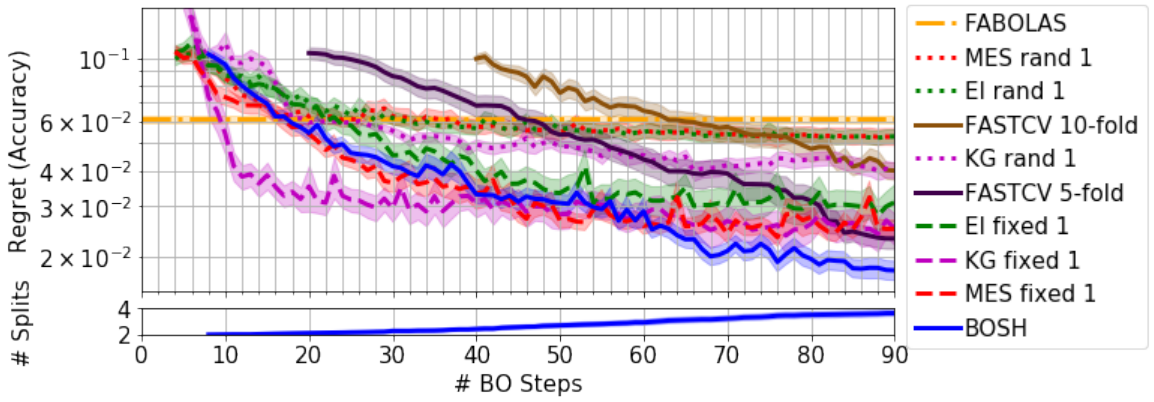
(b) B=5 (c) B=10

Figure 6.6.2: Optimising 7 parameters of a Lunar Lander controller.

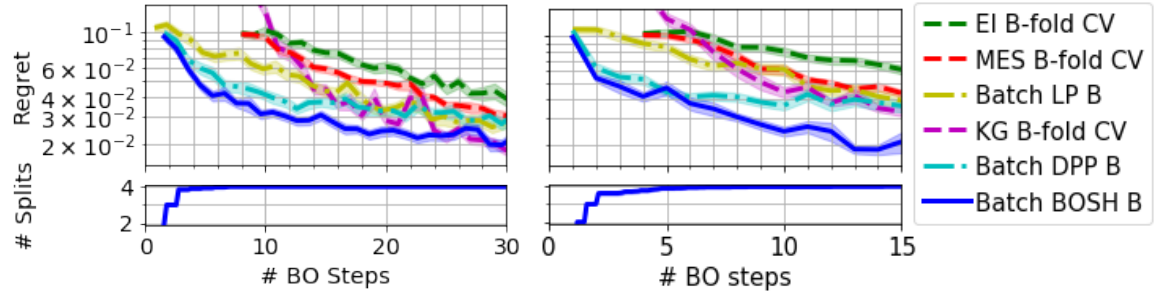
6.6.3 Hyper-parameter Tuning

We now test the performance of BOSH on two well-known ML hyper-parameter tuning tasks : using a support vector machine (SVM) to classify the sentiment in IMDB movie reviews (Maas et al., 2011) and using probabilistic matrix factorisation (PMF) (Mnih and Salakhutdinov, 2008) to recommend movies on the Movie-lens-100k data set (Hoffman et al., 2010). Here, we seek hyper-parameter values that provide the highest model performance. As already argued, the model scores based on a particular evaluation strategy do not necessarily correspond to the true performance and so true model performance is calculated for IMDB data on a large held-out test set and using expensive but reliable performance estimates based on 20 train-test splits for the PMF. We stress that these high-cost estimates are only performed retrospectively, after stopping the optimisation, and during the actual tuning our individual performance estimates are generated using a pool of randomly generated train-test splits for BOSH or single train-test splits and K-fold CV as fixed evaluation strategies for standard BO. As FABOLAS is able to query models using only small proportions of the available data, it is able to find reasonably well performing hyper-parameter configurations in a fraction of the computation used by standard BO and BOSH. However, we will see that a shortcoming of FABOLAS's reliance on low-fidelity performance estimates mean that even if allowed a significantly longer run-time, it fails to improve upon this chosen configuration (which we plot as a horizontal line).

These tuning tasks form two very different challenges for BOSH. Firstly our IMDB data-set is relatively small and, as it consists of textual data, is highly heterogeneous, meaning that there is high-variability between performance estimates made on different partitions of the data. Figure 6.6.3a shows that BOSH adaptively considers up to four seeds as the optimisation progresses, providing higher-precision tuning than standard BO based on single train-test splits. KG's initial fast optimisation is due to integration of its kernel parameters, however, even this computationally expensive BO routine is unable to reach the final precision of BOSH. In contrast, the much larger Movie-lens dataset has low variability between performance estimates and so should be able to be optimised without the need for fixed evaluation strategies. Unlike FASTCV, which



(a) B=1



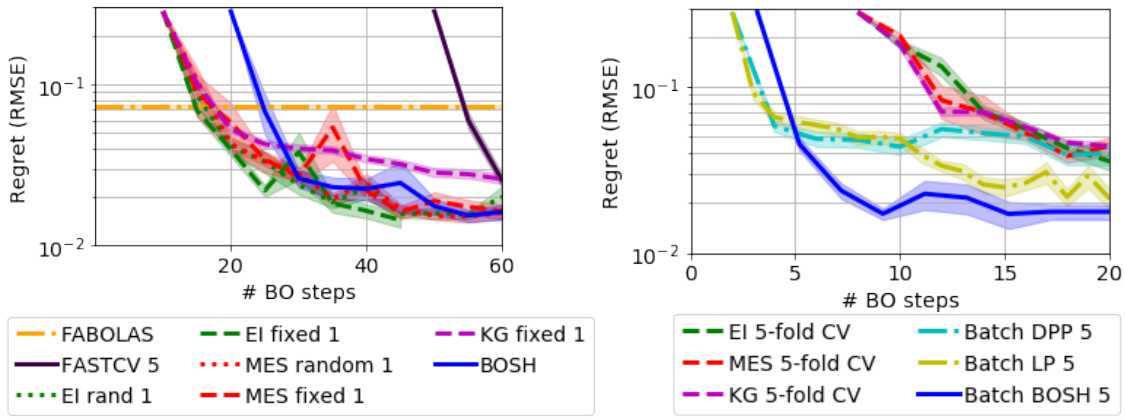
(b) B=5 (c) B=10

Figure 6.6.3: Tuning two SVM hyper-parameters for IMDB movie review classification.

incurs unnecessary costs for this task, BOSH's adaptive evaluation strategy allows it to closely match the performance of standard BO (Figure 6.6.4a). Across both tasks, FABOLAS fails to provide high-precision optimisation, although we do note that it identifies reasonably well-performing configuration using only as much computation as fitting a single model on all the data. When parallel resources are available, BOSH provides substantially faster tuning than BO under cross-validation and, for tasks with significant variability, more precise tuning than batch BO on single splits (Figures 6.6.3b,6.6.3c and 6.6.4b).

6.6.4 Simulation Optimisation

For our final experiment we consider a simulation optimisation problem from the set of benchmark tasks presented at *http://simopt.org/*. Here we wish to decide (x, y)



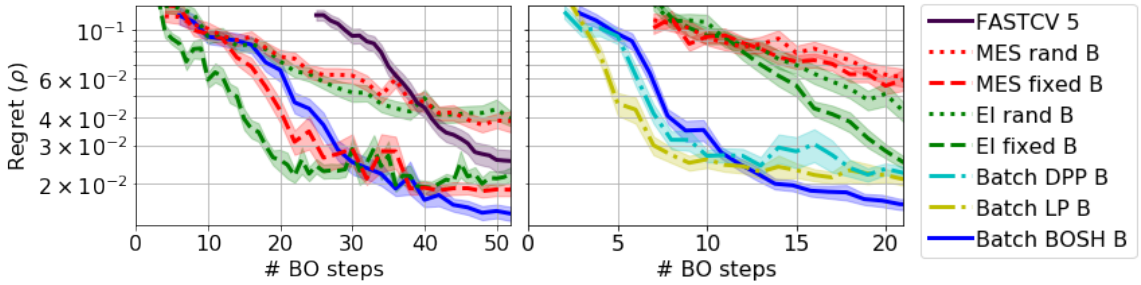
(a) B=1 (b) B=5

Figure 6.6.4: Tuning four PMF hyper-parameters to minimise root mean reconstruction error for movie recommendations.

locations of two warehousing facilities on a unit square. Orders arise throughout the square according to a pre-specified non-homogeneous Poisson process and each order is served by one of the ten trucks belonging to the closest warehouse (or queued if all trucks are busy). It is our goal to position the warehouses such that we maximise the proportion ρ of orders delivered within 60 minutes. Our base estimate of ρ comes from simulating demand for a single day according to a single random seed. We can calculate more reliable estimates by simulating demand for B independent days and we retrospectively estimate the true ρ with an expensive but reliable 100 day simulation. We see that although standard BO based on single day simulations is able to provide fast rough optimisation (Figure 6.6.5), efficient high-precision optimisation requires an adaptive evaluation strategy.

6.7 Conclusions

Optimising stochastic functions using Bayesian optimisation with a fixed evaluation strategy does not achieve the high precision optimisation that is commonly claimed, since it simply results in over-fitting to the evaluation strategy. We instead propose BOSH, an extension to Bayesian optimisation that instead deploys an adaptive



(a) B=1 (b) B=5

Figure 6.6.5: Allocating warehouses to cope with simulated demand.

evaluation strategy as the optimisation progresses using a novel principled information-theoretical framework.

Chapter 7

BOFFIN TTS: Few-shot Speaker Adaptation by Bayesian Optimisation

Status: Published as Moss H. B., Aggarwal V., Prateek N., Gonzalez J. & Barra-Chicote R., BOFFIN TTS: Few-shot Speaker Adaptation by Bayesian Optimisation, *The International Conference on Acoustics, Speech and Signal Processing*, 2020.

7.1 Preface

Our final chapter presents BOFFIN TTS (Bayesian Optimisation For FIne-tuning Neural Text To Speech), a real-world deployment of BO implemented while seconding at Amazon Research. Here, the task is to fine-tune a pre-trained TTS model to mimic a new speaker using a small corpus of target utterances. Contrary to the current practice in speaker adaptation, this chapter demonstrates that there does not exist a *one-size-fits-all* adaptation strategy, with convincing synthesis requiring a corpus-specific configuration of the hyper-parameters that control fine-tuning. By using Bayesian optimisation to efficiently optimise these hyper-parameter values for a target speaker, we are able to perform adaptation with an average 30% improvement in speaker similarity over standard techniques. Results indicate, across multiple corpora, that BOFFIN TTS can learn to synthesise new speakers using less than ten minutes of audio, achieving the same naturalness as produced for the speakers used to train

the base model.

7.2 Introduction

Given enough data, text to speech (TTS) systems can learn to convincingly mimic speakers across a wide range of acoustic and phonetic styles. However, training systems from scratch requires tens of hours of high-quality audio and reliable transcriptions, either from a single speaker to create speaker-specific models or spread across several speakers when training multi-speaker models (Latorre et al., 2019; van den Oord et al., 2016; Gibiansky et al., 2017; Ping et al., 2018). Training models on less data sacrifices quality and reliability Chung et al. (2019).

To scale TTS catalogues across speakers for whom we have limited data, we adapt existing multi-speaker systems to generate new speakers - a well-studied form of transfer learning known as **speaker adaptation** (Yamagishi et al., 2009). Adaptation is possible in scenarios where we have just minutes of target audio and partial phoneme coverage, as the robust representation of text and subsequent mappings to coherent speech are shared between the speakers (Latorre et al., 2019). Only a small proportion of our network's capacity encodes speaker-specific information. We, therefore, need only enough utterances to learn **speaker identity** (the characteristics defining a target speaker's voice).

Existing strategies for speaker adaptation fall into two broad categories. Many approaches use pre-trained auxiliary encoding networks to extract speaker characteristics to be combined with linguistic features as inputs to a TTS model (Li et al., 2017; Taigman et al., 2017; Nachmani et al., 2018; Jia et al., 2018). In contrast, alternative approaches fine-tune the weights of existing multi-speaker models to synthesise new speakers (Arik et al., 2018; Chen et al., 2018a). As fine-tuning provides the most natural adaptation across multiple TTS models (Arik et al., 2018; Chen et al., 2018a), and is applicable to any existing system (without the need for training additional encoding networks), it is the focus of this report.

Our primary contribution is to demonstrate that successful speaker adaptation

requires fine-tuning of adaptation hyper-parameters (henceforth referred to as the **adaptation strategy**) for each target speaker. We carefully tune the hyper-parameters governing adaptation and introduce two additional parameters not previously used for speaker adaption, demonstrating that the optimal hyper-parameter configuration depends subtly on the acoustic and phonetic properties of the target speaker alongside attributes of the target corpus (like audio-quality and size). For example, the amount of regularisation required to prevent over-fitting (of which few-shot speaker adaptation is particularly susceptible (Chen et al., 2018a)), depends on the quality and quantity of adaptation utterances.

In this work, we formulate few-shot speaker-adaptation as an optimisation problem - the task of finding appropriate hyper-parameter values for any given speaker. Our proposed BOFFIN[1] TTS system automatically and efficiently solves this optimisation problem through the hyper-parameter tuning framework of **Bayesian optimisation** (BO), providing a fully automatic speaker-adaptation system suitable for general target speakers. BO has been shown to find high-performing hyper-parameters in competitively few model fits for many machine learning tasks (Snoek et al., 2012), surpassing the performance of human tuners for problems in computer vision (Bergstra et al., 2013), natural language processing (Wang et al., 2015), and recently for reinforcement learning in AlphaGo (Chen et al., 2018b). However, BO has yet to see wide-spread use in TTS, where grid-based and random searches are still commonplace for hyper-parameter optimisation. We hope that our successes with BO for speaker adaptation will encourage its more wide-spread use across TTS.

We evaluate BOFFIN TTS across three distinct scenarios, varying both the number of speakers in the base multi-speaker model and corpora audio-quality.

[1]Boffin: British slang for a scientific expert.

7.3 System Description

7.3.1 Base Multi-Speaker Model

Our **base model** (the model we adapt to target speakers) is a Tacotron2 (Wang et al., 2017a) style multi-speaker system explained in detail by Latorre et al. (2019) and Prateek et al. (2019), consisting of an acoustic context-generation model and neural vocoder. Our acoustic model relies on an attention-based sequence-to-sequence network to generate context sequences (represented as mel-spectrograms) from input texts (see Figure 7.3.1). Unlike Tacotron2 which models raw graphemes, we pre-process input text with a grapheme-to-phoneme module. To condition on individual speakers, we learn a speaker-embedding from a one-hot-encoding of speaker IDs (following van den Oord et al. (2016)). This dense representation of speaker characteristics is presented to the attention module alongside encoded input text, to be decoded as a speaker-specific mel-spectrogram. Model weights are tuned with an ADAM optimiser to minimise the teacher-forced L1 loss between predicted and extracted mel-spectrograms. To complete the TTS pipeline, we convert mel-spectograms to waveforms using the multi-speaker neural vocoder of Lorenzo-Trueba et al. (2018). This vocoder is trained across 74 speakers and suitable for generating natural speech for our wide-range of adaptation speakers.

7.3.2 Base-line Speaker Adaptation System

Existing approaches for speaker adaptation by fine-tuning, although targeting different TTS architectures (Arik et al., 2018; Chen et al., 2018a), all share the same approach which we apply to our chosen model to form a **base-line** adaptation system. To synthesise speakers not present in the training corpus, we continue the same learning process used to train the base model, but replace the training data with utterances of only the target speaker to allow the fine-tuning of weights and the learning of a new speaker embedding with respect to this new data. To avoid over-fitting to small collections of target utterances, we hold-out 20% of adaptation data to form a

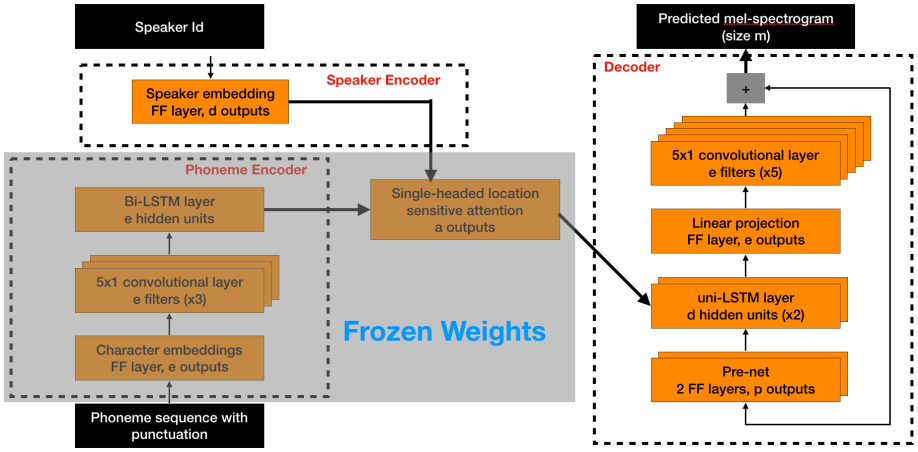


Figure 7.3.1: Multi-speaker acoustic model architecture.

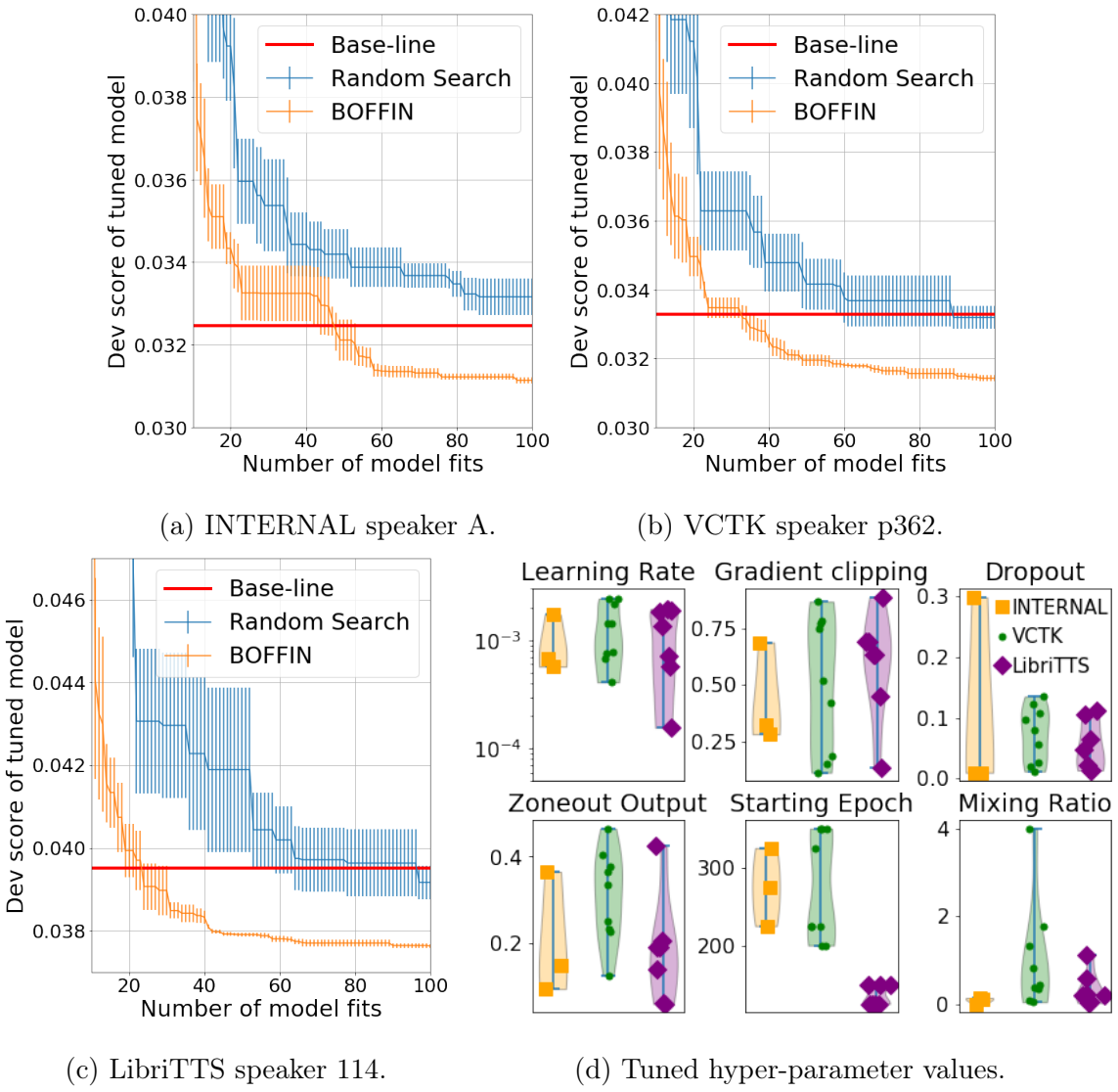
validation set for early-stopping. From extensive human tuning, we know that the hyper-parameter configuration chosen for our base-model is capable of producing high-quality synthesis (achieving higher than four MOS naturalness scores for several speakers). We, therefore, expect this hyper-parameter configuration to form a competitive base-line for adaptation. Nevertheless, we later demonstrate that we can achieve a substantial improvement in adaptation quality using BOFFIN TTS.

7.4 BOFFIN TTS

There are two key differences between BOFFIN TTS and the base-line adaptation system. We allow the hyper-parameters controlling our adaptation to change to suit the target-speaker and, crucially, propose a framework for finding their optimal configuration in an efficient and automatic manner.

7.4.1 How Does BOFFIN TTS Control Adaptation?

The key to effective adaptation is to learn characteristics of the target speaker without losing the generalisability of the base model (a phenomenon known as catastrophic forgetting). To this end, we believe there are nine key hyper-parameters that determine



(a) INTERNAL speaker A.

(b) VCTK speaker p362.

(c) LibriTTS speaker 114.

(d) Tuned hyper-parameter values.

Figure 7.4.1: (a, b, c): Loss of the current best hyper-parameter configuration found by each system as we adapt to three randomly selected speakers from each corpora. We plot means and standard error for BOFFIN TTS and RS based on 5 runs with different random seeds, alongside the loss achieved by the base-line adaptation system. (d): Hyper-parameter values chosen by BOFFIN TTS for multiple target speakers across three different data-sets. Each point represents a single speaker. We plot the six hyper-parameters whose optimal values show the largest variation across speakers.

the success of adaptation. These include seven parameters already widely used in machine learning to control learning dynamics (learning rate, batch size, decay-factor and gradient-clipping threshold) and to perform regularisation (dropout and two zoneout parameters), alongside two parameters unique to BOFFIN TTS.

Although, tuning these seven standard hyper-parameters allowed us to learn the identity of the target speaker, the resulting models often show poor generalisation capabilities. Therefore, we propose two additional hyper-parameters. Firstly, we supplement our adaptation corpus, forming a tune-able ratio of target speakers to speakers already seen by the model (a simple approach to mitigate catastrophic forgetting known as a rehearsal method). Finally, we also tune which epoch of our trained base-model from which we begin adaptation. A base model before full convergence to the base speakers can provide a model more amenable for adaptation.

In addition to hyper-parameter tuning, we also exploit the specific architecture of our chosen base-model. Rather than allowing our fine-tuning to update all model weights (as in Chen et al. (2018a)), we only allow fine-tuning of the weights in our speaker embedding and decoder modules (i.e those containing speaker-specific information, see Figure 7.3.1). We know that our encoder and attention modules are already able to facilitate synthesis across multiple speakers and we found that freezing their weights during adaptation led to more robust synthesis.

7.4.2 How Does BOFFIN TTS Optimise Adaptation?

Learning an optimal adaptation strategy for a target speaker is a difficult high-dimensional hyper-parameter optimisation (HPO) problem. As is common in HPO, this optimisation task is characterised by expensive evaluations (requiring a full adaptation to evaluate each single hyper-parameter configuration), a mixture of discrete and continuous variables, and a lack of analytical gradients for our objective function (the performance of adaptation) with respect to all our hyper-parameters. Consequently, we cannot apply gradient-based optimisers and the high-dimension of our tuning task makes a simple grid-search computationally infeasible (and likely ineffective (Bergstra and Bengio, 2012)). We, therefore, use Bayesian optimisation.

In a nutshell, BO is able to provide highly efficient HPO by using information from already evaluated hyper-parameter configurations to predict which untested configurations are 'likely' to perform well and therefore should be next evaluated. In particular, to choose the $t+1^{th}$ hyper-parameter for evaluation, we fit a Gaussian process model Rasmussen (2004a) to our t collected configuration-evaluation pairs $\mathcal{D}_t = \{\mathbf{x}_i, y_i\}_{i=1,..,t}$ across the hyper-parameter space $\mathcal{X}$, producing Gaussian predictions of performance at each configuration $\mathbf{x} \in \mathcal{X}$ of $y(\mathbf{x})|\mathcal{D}_t$. We then evaluate the configuration that we expect (according to our model) will provide the largest improvement over the best current best evaluation (with score $y'_t = \min_{i=1,..,t} y_i$), i.e we next evaluate configuration

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmax}}\, \mathbb{E}_{y(\mathbf{x})|\mathcal{D}_t}\left[\max(y'_t - y(\mathbf{x}), 0)|\mathcal{D}_t\right]. \tag{7.4.1}$$

For Gaussian processes, the inner expression of (7.4.1) and its gradients have convenient analytical forms (see Shahriari et al. (2016) for a comprehensive review of BO). Therefore, $\mathbf{x}_{t+1}$ can be efficiently found using a standard gradient-based optimiser.

We consider the performance of BOFFIN TTS when seeking to minimise L1 mel-spectogram loss across a held-out validation set of target speaker utterances. Although L1 loss does not necessarily correlate exactly with the perceptual quality of synthesised samples (as is the case for all objective TTS metrics), we found it informative enough to find hyper-parameters with high perceptual scores (Section 7.5). Adaptation to speakers from three different corpora is presented in Figure 7.4.1 (experimental details are discussed in Section 7.5). Our plots start after an initialisation stage of 10 random hyper-parameters, as this is required to provide a meaningful initial model across $\mathcal{X}$. Note that replacing BOFFIN TTS's BO component with random search (RS) fails to substantially improve upon our baseline (not speaker-specific) adaptation system. We need a sophisticated tuner like BO to find speaker-specific adaptation strategies. In addition, Figure 7.4.1d shows that not only does the optimal hyper-parameter configuration vary between data-sets, but also across each individual speaker within each corpora. For example, our proposed *Mixing Ratio* hyper-parameter requires larger values in general across the VCTK corpus than for our other corpora, however, the optimal *Mixing Ratio* still varies substantially across just the VCTK speakers.

System	INTERNAL	VCTK	LibriTTS
base-synth	3.45 ± 0.08	3.76 ± 0.10	3.10 ± 0.10
base-truth	3.84 ± 0.08	4.05 ± 0.08	4.10 ± 0.08
adapt-synth	3.43 ± 0.10	3.6 ± 0.10	2.90 ± 0.10
adapt-truth	4.05 ± 0.08	4.09 ± 0.08	3.97 ± 0.08

Table 7.4.1: Comparing the mean naturalness scores achieved by BOFFIN TTS on target speakers (adapt-synth), by the base multi-speaker model on base speakers(base-synth), and by true audio for both target (adapt-truth) and base-model speakers (base-truth). We present each listener with samples across multiple base and adapted speakers and ask for a 5 point score from 'completely unnatural' to 'completely natural'. We print mean responses alongside 95% confidence bounds.

7.5 Results

We have demonstrated that BOFFIN outperforms the base-line speaker adaptation system with respect to L1 loss. However, to investigate whether this lower score corresponds to an improvement in perceptual quality at inference time, we collected the perceptual evaluations of human listeners.

7.5.1 Experimental Protocol

To thoroughly test the performance of BOFFIN TTS, we consider three distinct corpora: (i) multi-speaker corpus with studio-quality recordings (referred to as INTERNAL[2]), (ii) the open-source VCTK corpus Veaux et al. (2017), and (iii) the LibriTTS audio-book corpus Zen et al. (2019). By considering a range of recording qualities and base-models with differing numbers of base speakers, we can understand the limitations of using BOFFIN TTS in a variety of practical settings. The architecture of our base-model remains fixed except for the more challenging LibriTTS task, where we double the size of our speaker embedding to accommodate a larger collection of base speakers. BO is performed with the Python library Emukit (Paleyes et al., 2019).

[2]The internal corpus contains no customer voice recordings.

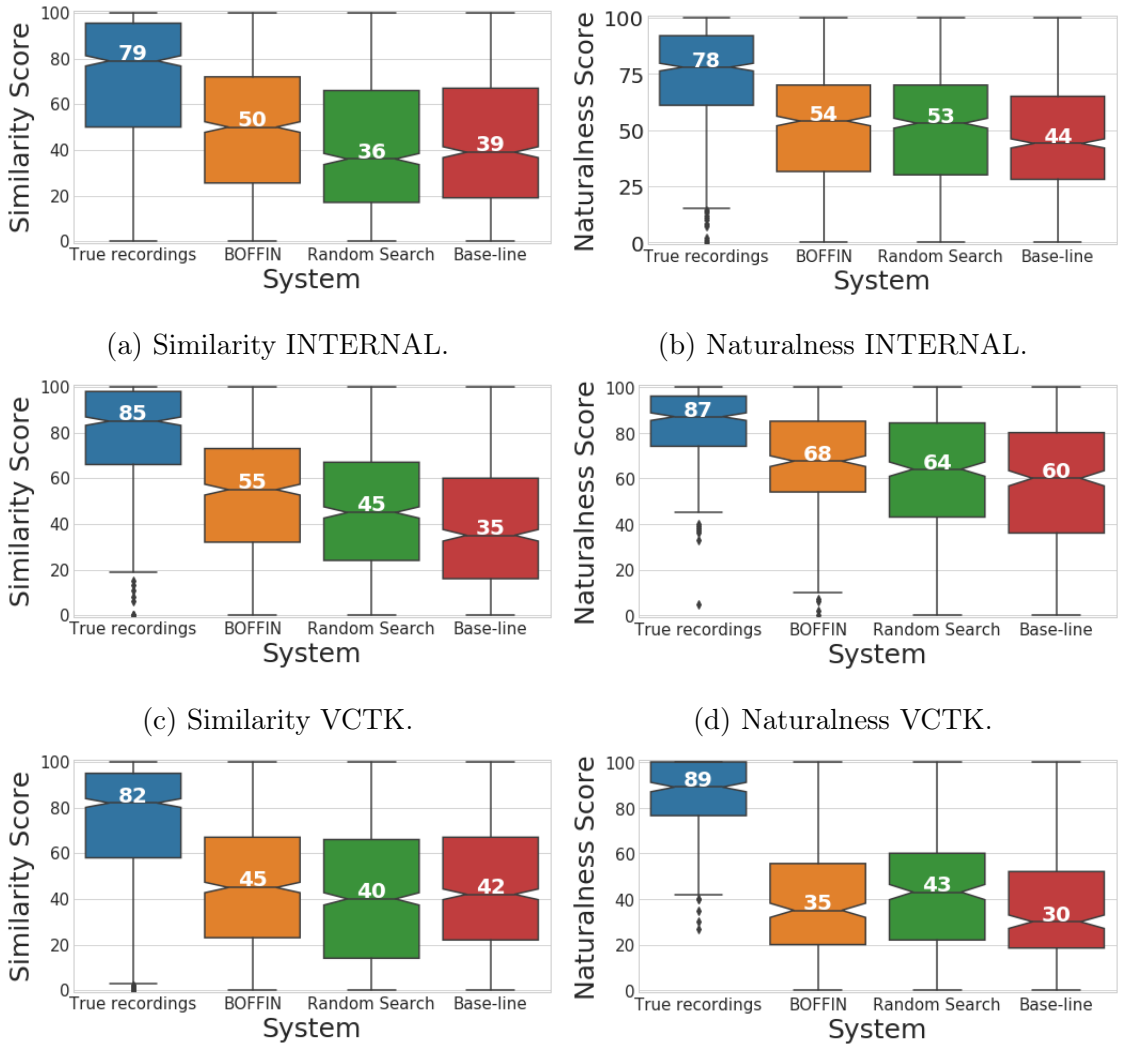
For each experiment, we adapt to 4 unseen speakers (from the same corpora used to train the base-model) using a random sample of 100 utterances (representing between 5 and 10 minutes of audio depending on the corpus), with 20% retained as a validation set. To evaluate each system, we compare naturalness and achieved similarity to the target speaker using a MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) test (Recommendation, 2001). We also compare the naturalness achieved by BOFFIN TTS on target speakers with that achieved by the base multi-speaker model on its original speakers using a Mean Objective Score (MOS) test for naturalness. Each evaluation is presented to 25 native US listeners using Amazon Mechanical Turk. Statistical significance tests are performed at the p=0.01 level with Bonferroni-Holm corrections, using paired t and Mann-Whitney U tests for the MUSHRA and MOS evaluations respectively.

7.5.2 Adaptation from a base-model with few speakers

For our first experiment, we train a base-model on 4 male and 4 female proprietary speakers (each with 2.5k utterances) and adapt to 2 female and 2 child held-out speakers. Figure 7.5.1a show that BOFFIN TTS is able to achieve significant improvements in speaker similarity, with an improvement of 28% over the base-line and 39% over RS. Crucially, Figure 7.5.1b shows BOFFIN TTS' improvement in similarity does not sacrifice perceptual quality, achieving a small but statistically significant improvement in naturalness over the base-line speaker adaptation system. Moreover, Table 7.4.1 demonstrates that BOFFIN TTS is able to adapt to target speakers without a significant drop in perceptual quality from the base-model's speakers (learnt with 250 times more data).

7.5.3 Adaptation from a moderately rich base-model

We now consider a harder adaptation task; adapting to VCTK speakers with much higher variation in expressiveness, prosody and audio-quality than those in INTERNAL. Our base-model is trained on 22 speakers: 14 from VCTK (with 400 utterances each)



(a) Similarity INTERNAL.

(b) Naturalness INTERNAL.

(c) Similarity VCTK.

(d) Naturalness VCTK.

(e) Similarity LibriTTS.

(f) Naturalness LibriTTS.

Figure 7.5.1: MUSHRA tests for speaker similarity and naturalness. For similarity, we presented the same utterance synthesised by each system alongside a reference recording of the target speaker on another utterance and requested a rating of each system between 'definitely a different person' (0) and 'definitely the same person' (100). For naturalness, we repeat without a reference recording and instead asked for ratings between 'completely unnatural' and 'completely natural'

supplemented with the 8 already considered in our first experiment (added to provide a more robust base-model). We adapt to 4 unseen VCTK speakers. This challenging adaptation scenario necessitates target speaker-specific adaptation strategies, with BOFFIN TTS providing significant improvements of 57% in similarity and 13% in naturalness over the base-line (Figures 7.5.1c and 7.5.1d). Moreover, Table 7.4.1 shows that BOFFIN TTS is once again able synthesise target speakers without a significant drop in naturalness than achieved for speakers already present in the base multi-speaker model.

7.5.4 Adaptation from a rich base-model

To understand the limitations of BOFFIN TTS, our final experiment considers an even larger base-model containing 200 speakers (each with 200 utterances) from LibriTTS. We adapt to 4 additional unseen libriTTS speakers. LibriTTS is derived from audio-books and so contain noise, artefacts, and highly expressive voices. Consequently, although BOFFIN TTS was able to adapt to target speakers without a statistically significant drop in naturalness over the speakers used to train the base-system (Table 7.4.1) (as is consistent with our other experiments), our base-model itself was of much lower quality than our other base-models, making it difficult for our MUSHRA listeners to make a statistically significant preference in similarity across all three systems (Figures 7.5.1e and 7.5.1f).

7.6 Conclusion

We propose the few-shot speaker-adaptation framework of BOFFIN TTS. By learning adaptation strategies custom to each target speaker, BOFFIN TTS can achieve higher speaker similarity than using a *one-size-fits-all* adaptation strategy, particularly when adapting to challenging target speakers from high-performance multi-speaker models.

A direction for future work is to provide an information-theoretic extension of BOFFIN. For example, we could use the GIBBON acquisition function of Chapter 5 to further improve speaker adaptation efficiency and to build a framework that

supports parallel computing resources. Unfortunately, we have been unable to test this hypothesis ourselves as the experiments ran in this paper require substantial computational resources, proprietary data, and a private code-base only accessible to Amazon employees.

Chapter 8

Conclusions

8.1 Final Remarks and Contributions

This thesis addresses challenging open problems in Bayesian optimisation. Novel approximation strategies have been proposed that greatly increase the applicability of information-theoretic BO. These approximations are analysed theoretically and empirically, with the resulting acquisition functions tested in exotic Bayesian optimisation frameworks developed in the remainder of the thesis. A primary focus is on developing BO frameworks for highly-structured input spaces.

In Chapter 3, this thesis proposed a novel computationally light information-theoretic approach for multi-task Bayesian optimisation. MUMBO reduces uncertainty in the optimal value of the objective function with each subsequent query, and provides principled decision-making across general multi-task structures at a cost which scales only linearly with the dimension of the search space. Consequently, MUMBO substantially outperforms current acquisitions across a range of optimisation and hyper-parameter tuning tasks.

Chapter 4 revolutionises the way in which Bayesian optimisation is performed for high-cost string design problems like molecular search and synthetic gene design. By departing from fixed-length representations of strings, BOSS is the first Bayesian optimisation method that acts directly over raw strings. BOSS is able to provide highly effective optimisation even for spaces obeying complicated syntactical constraints.

Chapter 6 provides BOSH, a challenging multi-task and batch Bayesian optimisation setting in which to test GIBBON. BOSH is inspired by optimisation problems with controllable observation noise, for example when choosing train-test splits for hyper-parameter optimisation or initial conditions for simulation optimisation. In particular, we combine GIBBON with hierarchical Gaussian processes to build an optimisation routine that adaptively increases the number of considered objective function realisations as the optimisation progresses, yielding more efficient and precise optimisation than standard BO methods based on fixed collections of realisations.

Our final chapter presents a real-world application of BO. Chapter 7 considers the task of few-shot speaker-adaptation, a well-established natural language processing pipeline which adapts an existing text-to-speech system to mimic a new speaker. Using BO, we are able to build a framework that can learn adaptation strategies custom to each target speaker, achieving a higher level of speaker similarity than existing methods.

8.2 Future Work and Possible Extensions

During this thesis, we have demonstrated the efficacy of performing information-theoretic search for many popular BO extensions. However, there are many other variants of BO that were not tackled in this thesis. Prominent examples already performed by information-theoretic BO include multi-objective (Hernández-Lobato et al., 2016; Belakaria et al., 2019) and constrained BO (Garrido-Merchán and Hernández-Lobato, 2019; Belakaria et al., 2020). Our GIBBON acquisition function could be used within these frameworks to provide batch extensions and reduce their computational overhead. Similarly, GIBBON could also be used to improve the performance of any framework relying on batch BO heuristics, for example in non-myopic BO (González et al., 2016b; Jiang et al., 2020).

BO for structured optimisation is still an open area of research. Although there is now a high-performance and computationally light-weight acquisition function suitable for these tasks (courtesy of GIBBON), future work is required to build frameworks

to expand the ideas of BOSS to other types of discrete structures beyond strings. Promising work has already begun on BO for neural network architecture design (Kandasamy et al., 2018b) and BO for combinatorial structures (Deshwal et al., 2020), however, many structures highly prevalent in machine learning, like trees and graphs remain largely unsupported by BO. To solve these remaining problems, our BOSS framework could be extended to support other convolution kernels such as tree (Collins and Duffy, 2002) and graph kernels (Vishwanathan et al., 2010).

Appendix A

Supplementary Material for MUMBO

A.1 Calculation of the MUMBO acquisition function

We now provide a thorough description of our proposed approach to calculate the MUMBO acquisition function for any choice of $\mathbf{x}$ and $\mathbf{z}$:

$$\alpha_n^{MUMBO}(\mathbf{x}, \mathbf{z}) = H(y(\mathbf{x}, \mathbf{z}) \,|\, D_n) - \mathbb{E}_{g^*}[H(y(\mathbf{x}, \mathbf{z}) \,|\, g^*, D_n)]\,. \tag{A.1.1}$$

For ease of notation we drop the dependence on $\mathbf{x}$ and $\mathbf{z}$, so that g denotes the target function value at $\mathbf{x}$, f denotes the evaluation of $\mathbf{x}$ at fidelity $\mathbf{z}$, and y denotes the (noisy) observed value of $f(\mathbf{x}, \mathbf{z})$, and seek to calculate the respective acquisition value α_n^{MUMBO}. From our underlying GP model we can extract our current beliefs about g and f as following a bi-variate Gaussian distribution:

$$\begin{pmatrix} g \\ f \end{pmatrix} \sim N\left[\begin{pmatrix} \mu_g \\ \mu_f \end{pmatrix}, \begin{pmatrix} \sigma_g^2 & \Sigma \\ \Sigma & \sigma_f^2 \end{pmatrix}\right].$$

Then, noting that $Cov(y, g) = \Sigma$, we can write a similar expression for our current beliefs about g and noisy observations y as

$$\begin{pmatrix} g \\ y \end{pmatrix} \sim N\left[\begin{pmatrix} \mu_g \\ \mu_f \end{pmatrix}, \begin{pmatrix} \sigma_g^2 & \Sigma \\ \Sigma & \sigma_f^2 + \sigma^2 \end{pmatrix}\right].$$

We now derive analytical expressions for these predictive distributions from our underlying GP model. We denote our chosen kernel (defined over $\mathcal{X} \times \mathcal{Z}$)

as k, so that $k((\mathbf{x}, \mathbf{z}), (\mathbf{x}', \mathbf{z}'))$ represents our prior co-variance between the evaluation of $\mathbf{x}$ on fidelity $\mathbf{z}$ and the evaluation of $\mathbf{x}'$ on fidelity $\mathbf{z}'$. Denote the location in the fidelity space that corresponds to the true objective function as $\mathbf{z}_0$ (i.e. $f(\mathbf{x}, \mathbf{z}_0) = g(\mathbf{x})$). For observations D_n, let $\mathbf{y}_n$ be the observed y values, define the kernel matrix $\mathbf{K}_n = [k((\mathbf{x}_i, \mathbf{z}_i), (\mathbf{x}_j, \mathbf{z}_j))]_{(\mathbf{x}_i, \mathbf{z}_i), (\mathbf{x}_j, \mathbf{z}_j) \in D_n}$ and kernel vectors $\mathbf{k}_n((\mathbf{x}, \mathbf{z})) = [k((\mathbf{x}_i, \mathbf{z}_i), (\mathbf{x}, \mathbf{z}))]_{(\mathbf{x}_i, \mathbf{z}_i) \in D_n}$. Then, following Rasmussen (2004a), the terms of our bi-variate Gaussian after observations D_n are:

$$
\begin{aligned}
\mu_g =& \mathbf{k}_n((\mathbf{x}, \mathbf{z}_0))^T (\mathbf{K}_n + \sigma^2 I)^{-1} \mathbf{y}_n \\
\mu_f =& \mathbf{k}_n((\mathbf{x}, \mathbf{z}))^T (\mathbf{K}_n + \sigma^2 I)^{-1} \mathbf{y}_n \\
\sigma_g^2 =& k((\mathbf{x}, \mathbf{z}_0), (\mathbf{x}, \mathbf{z}_0)) - \mathbf{k}_n((\mathbf{x}, \mathbf{z}_0))^T (\mathbf{K}_n + \sigma^2 I)^{-1} \mathbf{k}_n((\mathbf{x}, \mathbf{z}_0)) \\
\sigma_f^2 =& k((\mathbf{x}, \mathbf{z}), (\mathbf{x}, \mathbf{z})) - \mathbf{k}_n((\mathbf{x}, \mathbf{z}))^T (\mathbf{K}_n + \sigma^2 I)^{-1} \mathbf{k}_n((\mathbf{x}, \mathbf{z})) \\
\Sigma =& k((\mathbf{x}, \mathbf{z}), (\mathbf{x}, \mathbf{z}_0)) - \mathbf{k}_n((\mathbf{x}, \mathbf{z}))^T (\mathbf{K}_n + \sigma^2 I)^{-1} \mathbf{k}_n((\mathbf{x}, \mathbf{z}_0))).
\end{aligned}
$$

Following the advice of Snoek et al. (2012) we consistently use a Matérn 5/2 kernel to model performance surfaces over the hyper-parameter space.

The first term of (A.1.1) is simply the differential entropy of a Gaussian distribution and so can be calculated analytically as $\frac{1}{2} \log(2\pi e(\sigma_f^2 + \sigma^2))$. The second term of (3.4.1) is an expectation over the maximum value of the true objective g^*, which can be approximated using a Monte Carlo approach; we use Wang and Jegelka (2017)'s method to approximately sample from $g^* \mid D_n$ using a mean-field approximation and extreme value theory, generating a set of N values $G = \{g_1, \ldots, g_N\}$. For each d-dimensional example in Section 3.5, we base our mean-field approximation on a grid of GP predictions at $10,000d$ random locations and any already evaluated locations. Note that we generate only one set of N samples of g^* for each BO step and all the required acquisition queries in that step are calculated with respect to this sample.

All that remains is to calculate the quantity inside the expectation for a given value of g^*, i.e the differential entropy of the random variable $y|g < g^*$ with a distribution that we now derive.

A.1.1 Derivation of the Extended Skew Normal Distribution

To simplify notation, rather than manipulating the co-variance Σ directly, we derive MUMBO in terms of the predictive correlation between y and g:

$$\rho = \frac{\Sigma}{\sigma_g \sqrt{\sigma_f^2 + \sigma^2}}.$$

Then using the well-known result for the conditional distribution of a bi-variate normal, we know that $g \,|\, y$ is also normally distributed with mean $\mu_g + \frac{\sigma_g}{\sqrt{\sigma_f^2+\sigma^2}}\rho(y - \mu_f)$ and variance $\sigma_g^2(1-\rho^2)$. We can therefore write the cumulative distribution function for $y|g \leq g^*$ as

$$\begin{aligned}\mathbb{P}(y \leq \theta | g \leq g*) =& \frac{\mathbb{P}(y \leq \theta, g \leq g*)}{\mathbb{P}(g \leq g*)} \\ =& \frac{\int_{-\infty}^{\theta} \phi(\frac{u-\mu_f}{\sqrt{\sigma_f^2+\sigma^2}}) \Phi\left(\frac{g*-\mu_g - \frac{\sigma_g}{\sqrt{\sigma_f^2+\sigma^2}}\rho(u-\mu_f)}{\sqrt{\sigma_g^2(1-\rho^2)}}\right) du}{\sqrt{\sigma_f + \sigma^2}\Phi(\frac{g*-\mu_g}{\sigma_g})}.\end{aligned}$$

After differentiating with respect to θ and defining $\gamma_{g^*} = \frac{g^*-\mu_g}{\sigma_g}$, we can write down the probability density function for the standardized variable $Z_{g^*} = \frac{y-\mu_f}{\sqrt{\sigma_f^2+\sigma^2}} | g < g^*$ as;

$$p(\theta) = \frac{1}{\Phi(\gamma_{g^*})}\phi(\theta)\Phi\left(\frac{\gamma_{g^*} - \rho\theta}{\sqrt{1-\rho^2}}\right),$$

which we recognize as the density of an extended skew normal distribution (ESG) (Azzalini, 1985), with moments

$$\mathbb{E}(Z_{g^*}) = \rho\frac{\phi(\gamma_{g^*})}{\Phi(\gamma_{g^*})}, \quad \mathrm{Var}(Z_{g^*}) = 1 - \rho^2\frac{\phi(\gamma_{g^*})}{\Phi(\gamma_{g^*})}\left[\gamma_{g^*} + \frac{\phi(\gamma_{g^*})}{\Phi(\gamma_{g^*})}\right]. \tag{A.1.2}$$

As Arellano-Valle et al. (2013) show that the differential entropy of an ESG is non-analytical, so too must be the final term in our MUMBO acquisition (A.1.1). We therefore perform numerical integration using Simpson's rule across eight standard deviations around the mean of Z_{g^*} (quantities provided by (A.1.2)). Note that the equivalent quantity in the original implementation of MES (without fidelity considerations) has a truncated normal distribution, with a closed form expression for its entropy.

A.1.2 Derivation of the full MUMBO acquisition function

We can now derive the simplified form (3.4.2) of the MUMBO acquisition function presented in Section 3.4, starting from the information-theoretic definition (A.1.1). Noting that $H(y|g^*, D_n) = H(Z_{g^*}) + \frac{1}{2}\log(\sigma_f^2 + \sigma^2)$ and that $H(y|D_n) = \frac{1}{2}\log(2\pi e(\sigma_f^2 + \sigma^2))$, we can rewrite (A.1.1) for a fixed choice of $\mathbf{x}$ and $\mathbf{z}$ as

$$\alpha_n^{MUMBO} = \frac{1}{2}\log(2\pi e) - \mathbb{E}_{g^*}\left[H(Z_{g^*})\right].$$

The differential entropy $H(Z_{g^*})$ for a fixed sample g^* can be decomposed into three terms

$$H(Z_{g^*}) = \mathbb{E}_{\theta\sim Z_{g^*}}\left[-\log(\phi(\theta)) + \log(\Phi(\gamma_{g^*})) - \log\left(\Phi\left(\frac{\gamma_{g^*} - \rho\theta}{\sqrt{1-\rho^2}}\right)\right)\right]$$

After expanding the first of these terms as

$$\mathbb{E}_{\theta\sim Z_{g^*}}\left[-\log(\phi(\theta))\right] = \tfrac{1}{2}\mathbb{E}_{\theta\sim Z_{g^*}}\left[\theta^2\right] + \tfrac{1}{2}\log(2\pi),$$

and further expanding using our expressions for the moments of Z_{g^*}, we now have

$$\alpha_n^{MUMBO} = \mathbb{E}_{g^*}\left[\rho^2\frac{\gamma_{g^*}\phi(\gamma_{g^*})}{2\Phi(\gamma_{g^*})} - \log(\Phi(\gamma_{g^*})) + \mathbb{E}_{\theta\sim Z_{g^*}}\left[\log\left(\Phi\Big\{\frac{\gamma_{g^*} - \rho\theta}{\sqrt{1-\rho^2}}\Big\}\right)\right]\right].$$

Therefore, after reintroducing dependence on $\mathbf{x}$ and $\mathbf{z}$ and replacing the expectation over g^* with a Monte-Carlo approximation across our set of N samples G, we see that MUMBO can be expressed as

$$\begin{aligned}\alpha_n^{MUMBO}(\mathbf{x},\mathbf{z}) \approx \frac{1}{N}\sum_{g^*\in G}\Bigg[&\rho(\mathbf{x},\mathbf{z})^2\frac{\gamma_{g^*}(\mathbf{x})\phi(\gamma_{g^*}(\mathbf{x}))}{2\Phi(\gamma_{g^*}(\mathbf{x}))} - \log(\Phi(\gamma_{g^*}(\mathbf{x})))\\ &+ \mathbb{E}_{\theta\sim Z_{g^*}(\mathbf{x},\mathbf{z})}\left[\log\left(\Phi\Big\{\frac{\gamma_{g^*}(\mathbf{x}) - \rho(\mathbf{x},\mathbf{z})\theta}{\sqrt{1-\rho^2(\mathbf{x},\mathbf{z})}}\Big\}\right)\right]\Bigg].\end{aligned}$$

A.2 Experimental Details

We now provide implementation details for our all our experiments.

A.2.1 Discrete Multi-fidelity BO

Figure 3.5.1 shows the performance of MUMBO over the standard MF benchmark functions used by Xiong et al. (2013) and Kandasamy et al. (2016). These problems have an objective function and a discrete hierarchy of low-fidelity approximations that can be queried at reduced cost. We measure the performance of the MF approaches in terms of the total resources spent on query costs after random initializations. We wish to find high-performing incumbents after spending few resources. We generate starting points for the optimization by querying twice as many random points as the problem dimension and evaluate these across all fidelities. For the information-theoretic approaches we also provide the time spent deciding where to make each successive evaluation as this is an important practical consideration.

In Figure 3.5.1 we present the performance of the MF-GP-UCB algorithm of Kandasamy et al. (2017) using their published code. Unfortunately we were unable to achieve performance on these functions even close to the level claimed in their work. However, our approaches outperform even the results claimed in their paper. This performance discrepancy is likely due to our different initialization scheme and that we do not tune their algorithm's hyper-parameters (illustrating the benefit of using a parameter-free approach like MUMBO). Also note that MF-GP-UCB models fidelities as separate GPs, whereas MUMBO and MTBO use the more sophisticated coregionaliazation model.

We now provide detailed information about each of our synthetic functions.

Forrester Function. A single dimensional function (Forrester et al., 2008) defined on $\mathcal{X} = [0, 1]$ with three fidelitlies with query costs 10, 5 and 2:

$$f(x_1, 0) = (6x_1 - 2)^2 \sin(12x_1 - 4)$$
$$f(x_1, 1) = 0.75f(x_1, 0) + 3(x_1 - 0.5) + 2$$
$$f(x_1, 2) = 0.5f(x_1, 0) + 5(x_1 - 0.5) + 2$$

Currin exponential function (discete fidelity space). A two-dimensional

function defined on $\mathcal{X} = [0,1]^2$ with two fidelities queried with costs 10 and 1:

$$\begin{aligned} f(x_1, x_2, 0) =& \left(1 - \exp(-\frac{1}{2x_2})\right) \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20} \\ f(x_1, x_2, 1) =& \frac{1}{4} f(x_1 + 0.05, x_2 + 0.05, 0) \\ &+ \frac{1}{4} f(x_1 + 0.05, max(0, x_2 - 0.05), 0) \\ &+ \frac{1}{4} f(x_1 - 0.05, x_2 + 0.05, 0) \\ &+ \frac{1}{4} f(x_1 - 0.05, max(0, x_2 - 0.05), 0). \end{aligned}$$

Hartmann 3 function. A three-dimensional function with 4 local extrema defined on $\mathcal{X} = [0,1]^3$ with three fidelities ($m = 0, 1, 2$) queried at costs $100, 10$ and 1:

$$f(x_1, x_2, x_3, m) = -\sum_{i=1}^{4} \alpha_{i,m+1} \exp\left(-\sum_{j=1}^{3} A_{i,j}(x_j - P_{i,j})^2\right),$$

where

$$A = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}, \quad \alpha = \begin{pmatrix} 1 & 1.01 & 1.02 \\ 1.2 & 1.19 & 1.18 \\ 3 & 2.9 & 2.8 \\ 3.2 & 3.3 & 3.4 \end{pmatrix}, \quad P = \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}.$$

Hartmann 6 function. A six-dimensional function defined on $\mathcal{X} = [0,1]^6$ with four fidelities ($m = 0, 1, 2, 3$) queried at costs $1000, 100, 10$ and 1:

$$f(x_1, x_2, x_3, x_4, x_5, x_6, m) = -\sum_{i=1}^{4} \alpha_{i,m+1} \exp\left(-\sum_{j=1}^{6} A_{i,j}(x_j - P_{i,j})^2\right),$$

where

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \qquad \alpha = \begin{pmatrix} 1 & 1.01 & 1.02 & 1.03 \\ 1.2 & 1.19 & 1.18 & 1.17 \\ 3 & 2.9 & 2.8 & 2.7 \\ 3.2 & 3.3 & 3.4 & 3.5 \end{pmatrix},$$

$$P = \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}.$$

Borehole function. An eight-dimensional function defined on

$$\begin{aligned} \mathcal{X} = [&0.05, 0.15; 100, 50,000; 63070, 115600; 990, \\ &1110; 63.1, 116; 700, 820; 1120, 1680; 9855, 12055] \end{aligned}$$

with two fidelities queried with costs 10 and 1:

$$\begin{aligned} f(\mathbf{x}, 0) =& \frac{2\pi x_3(x_4 - x_6)}{\log(x_2/x_1)\left(1 + \frac{2x_7x_3}{log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right)}, \\ f(\mathbf{x}, 1) =& \frac{5x_3(x_4 - x_6)}{\log(x_2/x_1)\left(1.5 + \frac{2x_7x_3}{log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right)}. \end{aligned}$$

A.2.2 Continuous Multi-fidelity BO: FABOLAS

For our second set of experiments, we consider the MF hyper-parameter tuning framework of Klein et al. (2017a), which dynamically chooses the amount of training data used for hyper-parameter evaluations. Their FABOLAS algorithm is widely regarded as state-of-the-art, achieving hyper-parameter tuning with orders of magnitude less computation that standard BO and other competing MF tuning routines. We use the code provided for FABOLAS within the ROBO package (Klein et al., 2017b) by the same authors. We use their implementation exactly, only swapping out their original MTBO acquisition function for our proposed MUMBO acquisition. A good

hyper-parameter tuner finds hyper-parameter configurations that will perform well on new data after using as little computational resource as possible, including effort spent fitting models and deciding the hyper-parameter configuration and fidelity to query. By splitting our data into train, validation and test sets, we are able to report total wall-clock time against the performance (in accuracy) of incumbents on this test set (calculated retrospectively at the end of the optimization). During the optimization, models are trained on random subsets of chosen proportions of the training set and tested on the full validation set.

We consider the same examples as Klein et al. (2017a), using the same data-sets downloaded from the HPOlib BO benchmark repository (Eggensperger et al., 2013) of MNIST (Deng, 2012) and Vehicle Registrations (Siebert, 1987) - we refer the reader to their work for specific details. As a result of limited computational resources and wishing to repeat each experiment over multiple random seeds, we had to halve the training data (to $25,000$ for both MNIST and Vehicle) throughout the experiment (including testing the incumbents). We do, however, use the full test and validation sets. For each replication, we start with 10 random hyper-parameter initializations each evaluated on $\frac{1}{64}$, $\frac{1}{32}$, $\frac{1}{16}$ and $\frac{1}{8}$ of the training data.

A.2.3 Multi-task BO: FASTCV

In Section 3.5.4, we test MUMBO in a multi-task framework by providing the first information-theoretic implementation of FASTCV (Swersky et al., 2013), where we sequentially make evaluations on a single K-fold CV folds with the aim of optimizing the evaluations based on all K folds. As discussed in Section 3.5.4, the original implementation of FASTCV chooses hyper-parameters to evaluate and then the fold upon which to make the evaluation as a two-stage heuristic based on the expected improvement acquisition. In Figure 3.5.3, we investigate the change in performance of replacing this acquisition function with the principled MT decision-making provided by our MUMBO acquisition function. We also present the performance of standard BO routines that have to evaluate all K CV folds for each hyper-parameter query. For ML models, the acquisition function overheads are insignificant compared to the costs of

fitting the model on large proportions of the training data (unlike the small proportions chosen by FABOLAS), and so we measure the performance of our algorithms by the number of individual model fits required to reach a certain incumbent performance. To allow the fair comparison of the computational resources used by each algorithm, we consider a single optimization step as the evaluation of a single model on a single fold and so each hyper-parameter evaluation using K-fold CV counts as K steps.

We consider two well-known ML tasks: using a support vector machine (SVM) to classify the sentiment in IMDB movie reviews (Maas et al., 2011) and using probabilistic matrix factorization (PMF) (Mnih and Salakhutdinov, 2008) to recommend movies on the Movie-lens-100k data set (Hoffman et al., 2010). We tune the regularization strength across $[e^{-5}, e^{25}]$ and kernel coefficient across $[e^{-25}, e^{5}]$ for the SVM and the learning rate across $[0, 0.01]$, regularization strength across $[0, 0.1]$, matrix rank across $[50, .., 150]$ and number of model epochs across $[10, .., 50]$ for the PMF. To create a difficult MT optimization problem, we use only a small subset of the IMDB data (a random subset of $1,000$ reviews split into 10 folds) as this increases the between-fold variability of a K-fold CV estimate (Bengio and Grandvalet, 2004) and so limits the similarity of evaluations on different folds that is exploited by FASTCV. Despite this challenging MT set-up, both the original FASTCV and MUMBO are able to provide significantly faster tuning than standard approaches, with MUMBO providing an additional increase in test performance over FASTCV (as based on the reliable performance estimates calculated on the $49,000$ reviews not used for training). In addition, we also consider the whole of the large Movelens-100k dataset split into 5 folds. Despite the stochastic nature of PMF meaning that our tuning algorithms have deal with high levels of observation noise for each hyper-parameter evaluation, we once again we see that the principled decision-making of MUMBO allows much faster optimization than all the other approaches - achieving lower 5-fold CV estimated mean squared error (a standard measurement of performance for recommendation systems).

A.2.4 Wider Comparison With Existing Methods

We now present the functions used for final experiments.

Currin exponential function (continuous fidelity space). A two-dimensional function defined on $\mathcal{X} = [0,1]^2$ with fidelity space $z \in [0,1]$. The cost of querying fidelity z is given by $\lambda(z) = 0.1 + z^2$ with the objective lying at fidelity $z = 1$.

$$f(x_1, x_2, z) = \left(1 - 0.1(1-z)\exp(-\frac{1}{2x_2})\right) \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20}.$$

Rosenbrock function. A two-dimensional function defined on $\mathcal{X} = [-2,2]^2$ with two fidelities ($m = 0, 1$) queried at costs 1000 and 1. Observations are contaminated with Gaussian noise with variance 0.001 and $1e-6$ for each fidelity respectively :

$$\begin{aligned} f(x_1, x_2, 0) =& (1 - x_1)^2 + 100(x_2 - x_1^2)^2 \\ f(x_1, x_2, 1) =& f(x_1, x_2, 0) + 0.1\sin(10x_1 + 5x_2). \end{aligned}$$

Appendix B

Supplementary Material for BOSS

B.1 Dynamic Programs For SSK Evaluations and Gradients

We now detail recursive calculation strategies for calculating $k_n(\mathbf{a}, \mathbf{b})$ and its gradients with $O(nl^3)$ complexity. A recursive strategy is able to efficiently calculate the contributions of particular sub-string, pre-calculating contributions of the smaller sub-strings contained within the target string.

Adapting the recursion and notation of Beck and Cohn (2017) to our chosen contribution function, $k_n(\mathbf{a}, \mathbf{b})$ can be calculated by following for $i = 1, ..n-1$:

$$\begin{aligned}
\mathbf{K}'_0 &= \mathbf{1} \\
\mathbf{K}'_i &= \mathbf{D}^T_{|\mathbf{a}|}\mathbf{K}''_i\mathbf{D}_{|\mathbf{b}|} \\
\mathbf{K}''_i &= \lambda_m^2(\mathbf{M} \odot \mathbf{K}'_{i-1}) \\
k_i &= \lambda_m^2 \sum_{j,k}(\mathbf{M} \odot \mathbf{K}'_i)_{j,k},
\end{aligned}$$

producing the kernel evaluation $k_n(\mathbf{a}, \mathbf{b}) = \sum k_i$. Here, $\odot$ is the Hadamard product, $\mathbf{M}$ is the $|\mathbf{a}| \times |\mathbf{b}|$ matrix of character matches between the two strings ($M_{ij} = \mathbb{1}_{a_i}(b_j)$), and $\mathbf{D}_\ell$ is the $\ell \times \ell$ matrix

$$\mathbf{D}_\ell = \begin{bmatrix} 0 & 1 & \lambda_g & \cdots & \lambda_g^{\ell-2} \\ 0 & 0 & 1 & \cdots & \lambda_g^{\ell-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

The gradients of k_n with respect to the kernel parameters λ_m and λ_g can also be calculated recursively. For the kernel gradients with respect to match decay we calculate

$$\begin{aligned} \frac{\partial \mathbf{K}'_0}{\partial \lambda_m} &= \mathbf{0} \\ \frac{\partial \mathbf{K}'_i}{\partial \lambda_m} &= \mathbf{D}_{|\mathbf{a}|}^T \frac{\partial \mathbf{K}''_i}{\partial \lambda_m} \mathbf{D}_{|\mathbf{b}|} \\ \frac{\partial \mathbf{K}''_i}{\partial \lambda_m} &= 2\lambda_m (\mathbf{M} \odot \mathbf{K}'_{i-1}) + \lambda_m^2 \left(\mathbf{M} \odot \frac{\partial \mathbf{K}'_{i-1}}{\partial \lambda_m} \right) \\ \frac{\partial k_i}{\partial \lambda_m} &= \sum_{j,k} \left[2\lambda_m (\mathbf{M} \odot \mathbf{K}'_{ijk}) + \lambda_m^2 \left(\mathbf{M} \odot \frac{\partial \mathbf{K}'_{ijk}}{\partial \lambda_m} \right) \right], \end{aligned}$$

producing the gradient $\frac{\partial k_n(\mathbf{a},\mathbf{b})}{\partial \lambda_m} = \sum \frac{\partial k_i}{\partial \lambda_m}$.

Similarly, kernel gradients with respect to gap decay are calculated by

$$\begin{aligned} \frac{\partial \mathbf{K}'_0}{\partial \lambda_g} &= \mathbf{0} \\ \frac{\partial \mathbf{K}'_i}{\partial \lambda_g} &= \frac{\partial \mathbf{D}_{|\mathbf{a}|}^T}{\partial \lambda_g} \mathbf{K}''_i \mathbf{D}_{|\mathbf{b}|} + \mathbf{D}_{|\mathbf{a}|}^T \frac{\partial \mathbf{K}''_i}{\partial \lambda_g} \mathbf{D}_{|\mathbf{b}|} + \mathbf{D}_{|\mathbf{a}|}^T \mathbf{K}''_i \frac{\partial \mathbf{D}_{|\mathbf{b}|}}{\partial \lambda_g} \\ \frac{\partial \mathbf{K}''_i}{\partial \lambda_g} &= \lambda_m^2 \left(\mathbf{M} \odot \frac{\partial \mathbf{K}'_{i-1}}{\partial \lambda_g} \right) \\ \frac{\partial k_i}{\partial \lambda_g} &= \lambda_m^2 \sum_{j,k} \left(\mathbf{M} \odot \frac{\partial \mathbf{K}'_{ijk}}{\partial \lambda_g} \right), \end{aligned}$$

producing the gradient $\frac{\partial k_n(\mathbf{a},\mathbf{b})}{\partial \lambda_g} = \sum \frac{\partial k_i}{\partial \lambda_g}$, where $\frac{\partial \mathbf{D}_\ell}{\partial \lambda_g}$ is the $\ell \times \ell$ matrix

$$\frac{\partial \mathbf{D}_\ell}{\partial \lambda_g} = \begin{bmatrix} 0 & 0 & 1 & 2\lambda_g & 3\lambda_g^2 & \cdots & (\ell-2)\lambda_g^{\ell-3} \\ 0 & 0 & 0 & 1 & 2\lambda_g & \cdots & (\ell-3)\lambda_g^{\ell-4} \\ 0 & 0 & 0 & 0 & 1 & \cdots & (\ell-4)\lambda_g^{\ell-5} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots 1 & \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

B.2 Context-free Grammars

Context-free grammars (CFG) are 4-tuples $G = (V, \Sigma, R, S)$, consisting of:

- a set of non-terminal symbols V,
- a set of terminal symbols Σ (also known as an alphabet),
- a set of production rules R,
- a non-terminal starting symbol S from which all strings are generated.

Production rules are simple maps permitting the swapping of non-terminals with other non-terminals or terminals. All strings generated by the CFG can be broken down into a (non-unique) tree of production rules with the non-terminal starting symbol S at its head. These are known as the parse trees and are demonstrated in Figure 4.5.1 in the main paper.

The CFG for the symbolic regression task of Section 4.6.3 is given by the following rules:

$S \to S$ '+' T

$S \to S$ '$*$' T

$S \to S$ '/' T

$S \to T$

$T \to$ '(' S ')'

$T \to$ 'sin(' S ')'

$T \to$ 'exp(' S ')'

$T \to$ 'x'

$T \to$ '1'

$T \to$ '2'

$T \to$ '3',

where $V = \{S, T\}$ and $\Sigma = \{+, *, /, x, 1, 2, 3\}$. Although each individual production rule is a simple replacement operation, the combination of many such rules can specific a string space with complex syntactical constraints. For example, these 11 rules are able to specify that the string '(sin(2*x)+3(x*(2+exp(x))))+1/2' is valid but that '(sin(2*x)+3(x*(2+exp(x)))+1/2' (with invalid bracket closing) is not.

Sampling from the CFG. One of the advantages of CFGs is that it is easy (and cheap) to generate large collections of valid strings by recursively sampling production rules. However, when sampling strings from the grammar, we found this simple sampling strategy to produce long and repetitive strings. For our BO applications, where sample diversity is key, we instead employed a sampling strategy that down-weights the probability of selecting a particular rule based on the number of times it has already occurred in the parse tree. In particular, the probability of applying a particular rule to a non-terminal is proportional to c^n, where n is the number of occurrences of that rule in the current branch and c is a discount factor (set to 0.1 in all our experiments). The construction of this sampler ensures that a wide range of production rules are used when generating from the CFG.

B.3 Genetic Algorithms

We now provide implementation details for our GA acquisition function optimisers. During each GA step, populations are refined through stochastic biologically-inspired operations, providing a population achieving (on average) higher scores. The GA begins with a randomly sampled population and ends once the best string in the population stops improving between iterations (Algorithm 4). The N strings of the $i + 1^{th}$ population are perturbations of the i^{th} population. To evolve a population (Algorithm 5), a *tournament* process first selects n candidate strings (with replacement) attaining the highest evaluations across random sub-samples of a proportion p_t of the current population. To create the next population, these candidate strings undergo stochastic perturbations: a *mutation* operation producing a new offspring string from a single parent, and a *crossover* operation combining attributes of two parent strings to produce two new offspring. These operations occur with probability p_c and p_m respectively, which, alongside p_t, control the level of diversity maintained across populations. To highlight the robustness of our genetic algorithm acquisition optimiser, we do not tune the evolution parameters to each task, using populations of 100 candidate strings and $(p_t, p_c, p_m) = (0.5, 0.75, 0.1)$ for all our experiments. The exact crossover and mutation operators chosen to traverse string spaces under different syntactical constraints are discussed in the main paper.

B.4 Experimental Details

We now provide implementation details for our all our experiments.

B.4.1 Synthetic String Optimisation Experiments

Although seemingly simple tasks, our synthetic string optimisation tasks of Section 4.6.1 are deceptively challenging, as only a very small proportion of valid strings produce high scores. In fact, these tasks are considerably more challenging than the common benchmarks used to test standard BO frameworks. Figure B.4.1, shows the

Algorithm 4 Genetic Algorithms for Acquisition Function Maximisation

function GA(p_t, p_c, p_m, N)
  $n \leftarrow 0$
  Sample N strings for initial population P_0
  Evaluate acquisition function $A_0 \leftarrow \alpha(P_0)$
  Store current best value $\alpha_{best} \leftarrow \max(A_0)$
  while $\alpha_{best} = \max(A_n)$ **do**
    Begin new iteration $n \leftarrow n + 1$
    Evolve population $P_n \leftarrow$ **EVOLVE**(P_{n-1}, p_t, p_c, p_m)
    Evaluate acquisition function $A_n \leftarrow \alpha(P_n)$
    Store current best value $\alpha_{best} \leftarrow \max(\max(A_{n-1}), \alpha_{best})$
  return String achieving score α_{best}

performance attained by random search over our synthetic string tasks and standard benchmarks [1]. All objective functions are standardised ($\in [0, 1]$) and we run 1000 optimisation steps, plotting the mean and standard error across 25 replications. We see that our easiest synthetic string optimisation tasks are among the hardest of the standard benchmark problems to solve with random search, and we expect this to hold similarly for BO.

We now provide comprehensive experimental results across the synthetic string optimisation tasks. In Figures B.4.2,B.4.3,B.4.4,B.4.5,B.4.6,B.4.7 and B.4.8, we show the performance and computational overhead of our string kernels, extending the analysis from the main paper to include a variety of sub-sequence lengths considered by the string and feature-based kernels. We see that the string kernels always provide superior optimisation over existing kernels, with the string kernel based on sub-sequences of maximum length 5 consistently among the best. The string kernel is particularly effective for the most complicated objective functions (Figures B.4.3 and B.4.7) and when observations are contaminated by observation noise (Figure B.4.6). For problems with larger alphabets (and so significantly larger search spaces), our

[1] *https://www.sfu.ca/ ssurjano/index.html*

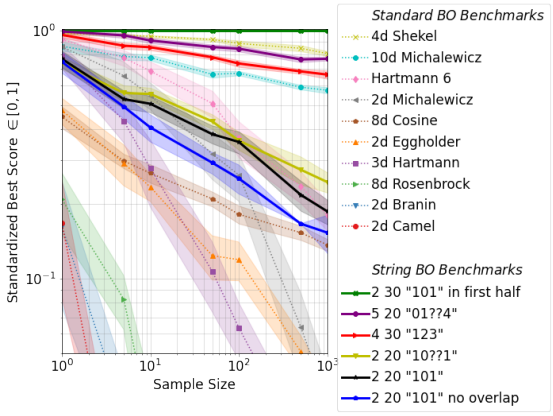


Figure B.4.1: Comparing random search across standard BO benchmarks (faint) and our synthetic string experiments (bold). For the string tasks, the legend *ALS* denoted the task with an alphabet of size A, strings of length L and counting the occurrences of the pattern S.

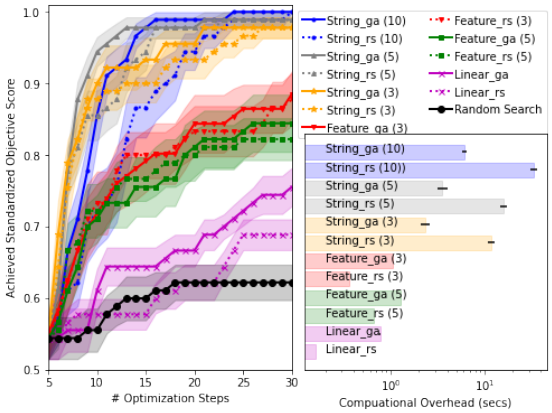


Figure B.4.2: Optimising the number of non-overlapping occurrences of "101" in a string of length 20 and alphabet ["0","1"]

genetic algorithm acquisition optimiser dramatically outperforms a larger budget random search optimiser (Figure B.4.5 and B.4.7).

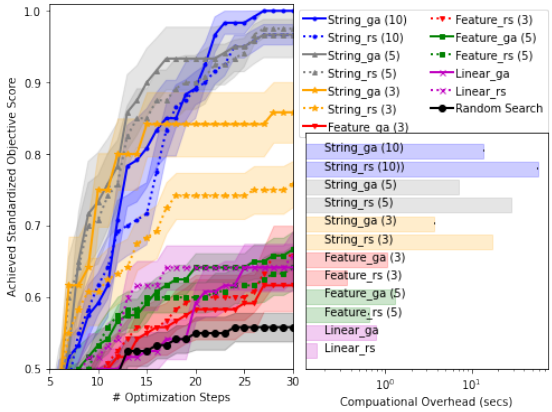


Figure B.4.3: Optimising the number of occurrences of "10??1" in a string of length 20 and alphabet ["0","1"]

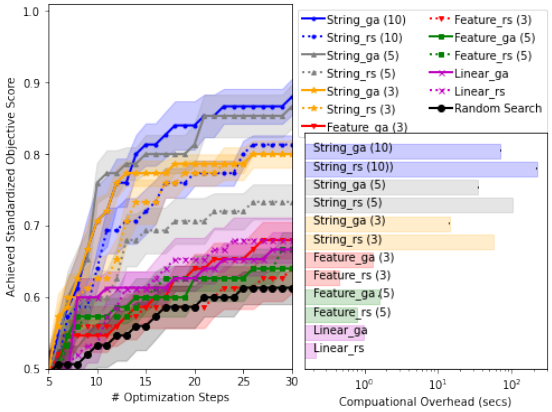


Figure B.4.4: Optimising the number of occurrences of "101" in the first half of a string of length 30 and alphabet ["0","1"].

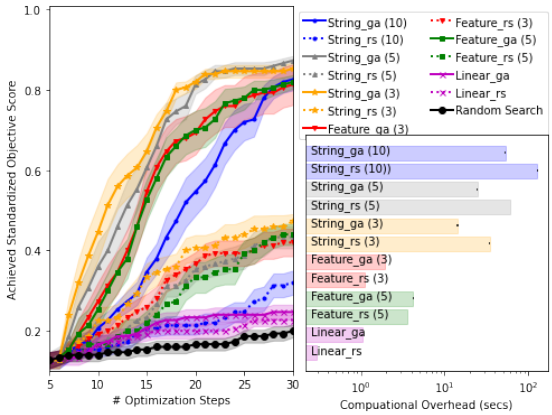


Figure B.4.5: Optimising the number of occurrences of "123" of a string with length 30 and an alphabet of ["0","1","2","3"].

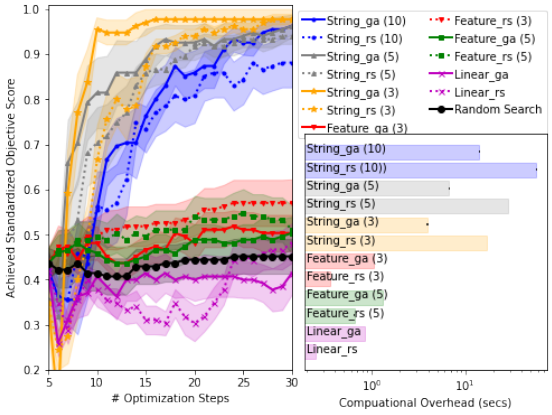


Figure B.4.6: Optimising the number of occurrences of "101" with observations contaminated by Gaussian noise (with a variance of 2) of a binary string of length 20.

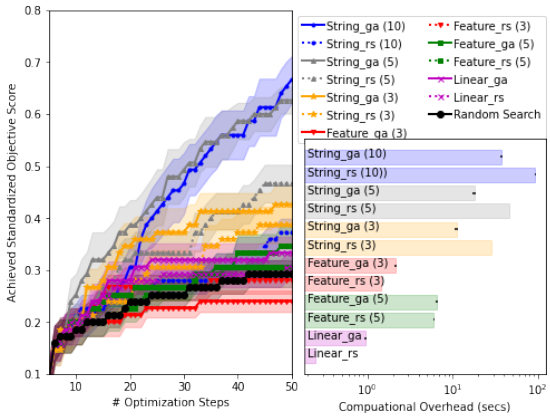


Figure B.4.7: Optimising the number of occurrences of "01??4" in a string of length 20 and alphabet ["0","1","2,"3","4"]

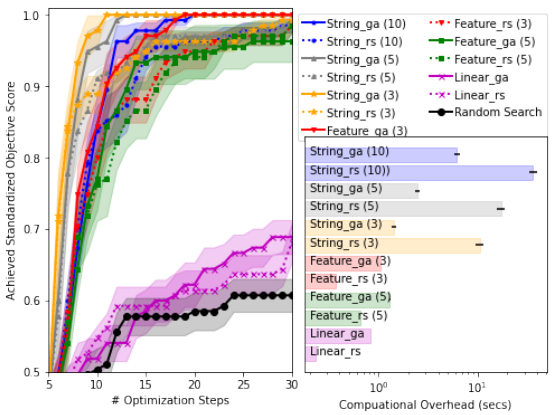


Figure B.4.8: Optimising the number of occurrences of "101" in a string of length 20 and alphabet ["0","1"]

B.4.2 Protein Optimisation

We now provide additional details for our four protein optimisation experiments, each targeting one of the following proteins.

1. Cystic fibrosis transmembrane conductance regulator:

   ```
   TIKENIFGVS.
   ```

2. Invertebrate iridescent virus 6 (IIV-6) (Chilo iridescent virus):

   ```
   MTSRGHLRRAPCCYAFKSATSHQRTRTSLCLASPPAPHCLLLYSHRCLTYFTVDYELSFFCL.
   ```

3. Anaphase-promoting complex subunit 15B:

   ```
   MSTLFPSLLPQVTDSLWFNLDRPCVDENELQQQEQQHQAWLLSIAEKDSSLVPIGKPASEPY
   DEEEEEDDEDDEDSEEDSEDDEDMQDMDEMNDYNESPDDGEIEADMEGAEQDQDQWMI.
   ```

4. Tyrosine-protein kinase abl-1:

   ```
   MGHSHSTGKEINDNELFTCEDPVFDQPVASPKSEISSKLAEEIERSKSPLILEVSPRTPDSV
   QMFRPTFDTFRPPNSDSSTFRGSQSREDLVACSSMNSVNNVHDMNTVSSSSSSSAPLFVALY
   DFHGVGEEQLSLRKGDQVRILGYNKNNEWCEARLYSTRKNDASNQRRLGEIGWVPSNFIAPY
   NSLDKYTWYHGKISRSDSEAILGSGITGSFLVRESETSIGQYTISVRHDGRVFHYRINVDNT
   EKMFITQEVKFRTLGELVHHHSVHADGLICLLMYPASKKDKGRGLFSLSPNAPDEWELDRSE
   ```

```
IIMHNKLGGGQYGDVYEGYWKRHDCTIAVKALKEDAMPLHEFLAEAAIMKDLHHKNLVRLLG
VCTHEAPFYIITEFMCNGNLLEYLRRTDKSLLPPIILVQMASQIASGMSYLEARHFIHRDLA
ARNCLVSEHNIVKIADFGLARFMKEDTYTAHAGAKFPIKWTAPEGLAFNTFSSKSDVWAFGV
LLWEIATYGMAPYPGVELSNVYGLLENGFRMDGPQGCPPSVYRLMLQCWNWSPSDRPRFRDI
HFNLENLISSNSLNDEVQKQLKKNNDKKLESDKRRSNVRERSDSKSRHSSHHDRDRDRESLH
SRNSNPEIPNRSFIRTDDSVSFFNPSTTSKVTSFRAQGPPFPPPPQQNTKPKLLKSVLNSNA
RHASEEFERNEQDDVVPLAEKNVRKAVTRLGGTMPKGQRIDAYLDSMRRVDSWKESTDADNE
GAGSSSLSRTVSNDSLDTLPLPDSMNSSTYVKMHPASGENVFLRQIRSKLKKRSETPELDHI
DSDTADETTKSEKSPFGSLNKSSIKYPIKNAPEFSENHSRVSPVPVPPSRNASVSVRPDSKA
EDSSDETTKDVGMWGPKHAVTRKIEIVKNDSYPNVEGELKAKIRNLRHVPKEESNTSSQEDL
PLDATDNTNDSIIVIPRDEKAKVRQLVTQKVSPLQHHRPFSLQCPNNSTSSAISHSEHADSS
ETSSLSGVYEERMKPELPRKRSNGDTKVVPVTWIINGEKEPNGMARTKSLRDITSKFEQLGT
ASTIESKIEEAVPYREHALEKKGTSKRFSMLEGSNELKHVVPPRKNRNQDESGSIDEEPVSK
DMIVSLLKVIQKEFVNLFNLASSEITDEKLQQFVIMADNVQKLHSTCSVYAEQISPHSKFRF
KELLSQLEIYNRQIKFSHNPRAKPVDDKLKMAFQDCFDQIMRLVDR.
```

As each amino acid in these protein sequences can be represented as one of a set of possible codons (triples of bases), the string spaces for these problems are incredibly large, with each space containing 5.53e+4, 9.48e+33, 4.81e+49 and 1.22e+614 unique strings, respectively. The permitted mappings from amino acids to valid codons are as follows:

$F \rightarrow ttt\,|\,ttc$ $\quad$ $I \rightarrow att\,|\,atc\,|\,ata$

$L \rightarrow tta\,|\,ttg\,|\,ctt\,|\,ctc\,|\,cta,\ \ ctg$ $\quad$ $M \rightarrow atg$

$S \rightarrow tct\,|\,tcc\,|\,tca\,|\,tcg\,|\,agt\,|\,agc$ $\quad$ $T \rightarrow act|acc|aca|acg$

$Y \rightarrow tat\,|\,tac$ $\quad$ $N \rightarrow aat|aac$

$C \rightarrow tgt\,|\,tgc$ $\quad$ $K \rightarrow aaa|aag$

$W \rightarrow tgg$ $\quad$ $V \rightarrow gtt\,|\,gtc\,|\,gta\,|\,gtg$

$P \rightarrow cct|ccc|cca|ccg$ $\quad$ $A \rightarrow gct|gcc|gca|gcg$

$H \rightarrow cat|cac$ $\quad$ $D \rightarrow gat|gac$

$Q \rightarrow caa|cag$ $\quad$ $E \rightarrow gaa|gag$

$R \rightarrow cgt|cgc|cga|cgg|aga|agg$ $\quad$ $G \rightarrow ggt|ggc|gga|ggg.$

Figure B.4.9 extends the analysis of our protein optimisation tasks to include the computational overheads incurred by each each BO routine (as measured on a single processor). The high evaluation costs of our SSK means that its overhead is substantially greater than the other approaches. However, in real gene design loops, this additional computational cost (hours) is negligible compared to the cost and time saved in wet-lab experiments (days). Moreover, the acquisition function calculations

can be trivially parallelised across up to 100 cores (the size of the populations used in the GA acquisition function optimiser) as well as across the m partial SSK calculations. If GPUs are available, these can also be used to efficiently calculate SSKs (Beck and Cohn, 2017).

B.4.3 BO in a VAE's Latent Space

To perform BO in the latent space of a VAE, we follow the set-up of Kusner et al. (2017), fitting a GP with an SE kernel and using a multi-start gradient descent acquisition function optimiser. We tried SE kernels with both individual and tied length scales across latent dimensions, however, this did not have a significant effect on performance, possibly due to difficulties in estimating many kernel parameters in these low-data BO problems. In order to perform BO, a compact area of the latent space must be chosen for the search space. Unfortunately, Kusner et al. (2017) do not provide details about how this should be determined. We chose the space containing the most central 75% of representations from the set of strings used to train the VAE ($100,000$ arithmetic expressions). We also tried using the space containing all representations from the training data, however, this led to a drop in optimisation performance, likely due to less reliable encoding/decoding learned by the VAE in these more sparsely supported parts of the latent space.

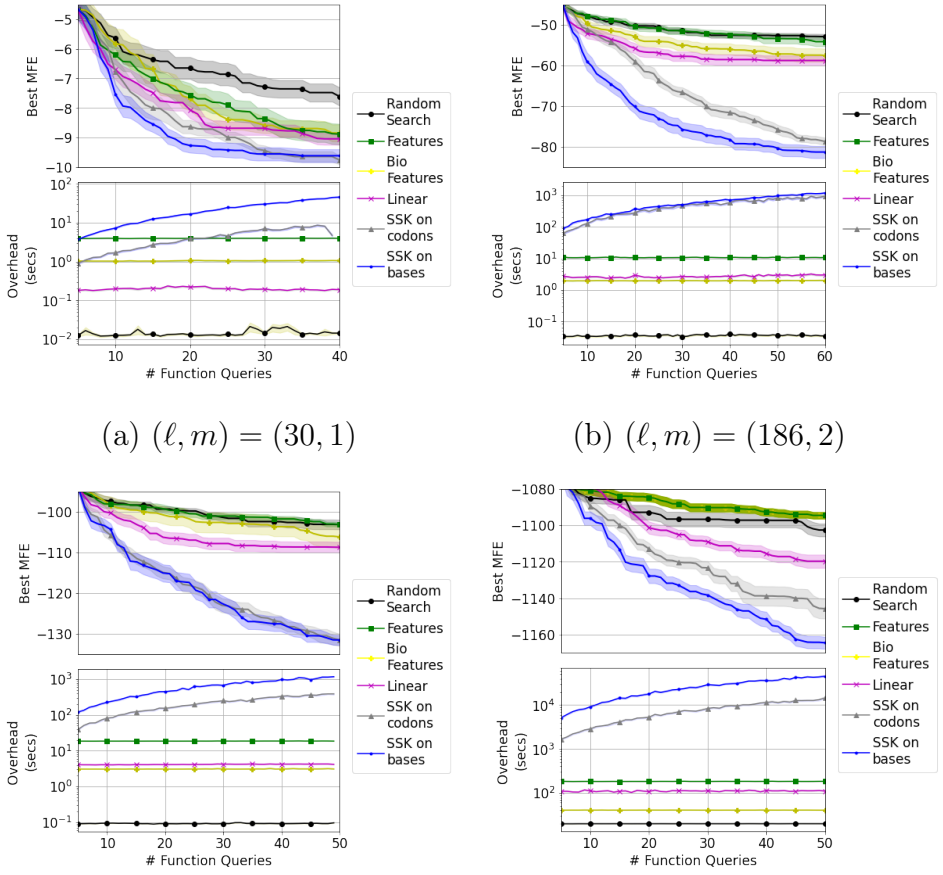
B.4.4 Visualising BO Surrogate Models

In Section 4.6.4, we present a kernel principal component analysis (KPCA) visualisation of the feature space induced by our SSK. We now extend this analysis to include the VAE competitors. In particular, we perform KPCA on the SE kernel used to define a surrogate model over each VAE's latent representations (Figure B.4.10). All figures show the representations of the same sampled $4,000$ SMILES strings, color-coded to represent their molecule scores (a linear combination of their water-octanol partition coefficient, ring-size and synthetic accessibility). We see that the GP with an SSK produces a significantly smoother KPCA space that the GPs fit in VAE latent space,

with the *CVAE* showing slightly more structure than the *GVAE*. This ranking matches the relative performance of the BO routines based on these surrogate models (Figure 4.6.3). So although the latent spaces of these VAE have been shown to exhibit some smoothness (Kusner et al., 2017), this is not captured by the GP model. Figure 4.6.3.d visualises the intrinsic representation of an SSK when kernel parameters are purposely chosen to provide a bad fit. We choose very low λ_m and high λ_g to heavily penalise the long contiguous sub-sequences we know to be informative for this task. The stark difference in smoothness between the visualisations of the tuned and badly-tuned SSKs demonstrates their flexibility as well as the importance of using a representation supervised to the the specific objective function of interest.

Algorithm 5 Evolution of Genetic Algorithm Populations

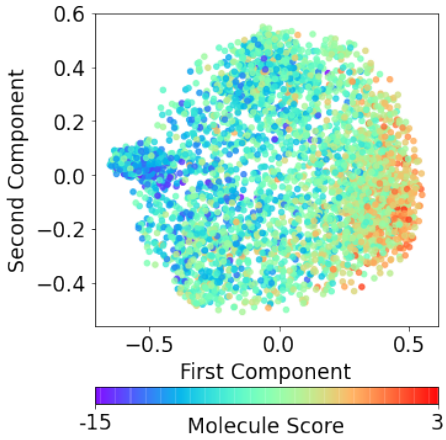
function EVOLVE(P,p_t, p_c, p_m)
  Initialise new population $P_{new} \leftarrow \emptyset$
  while $|P_{new}| < |P|$ **do**
    Collect a candidate string $s_1 \leftarrow$ **TOURNAMENT**(P, p_t)
    Sample $r \sim U[0, 1]$
    if $r < p_c$ **then**
      Sample another candidate string $s_2 \leftarrow$ **TOURNAMENT**(P, p_t)
      Perform crossover $s_1, s_2 \leftarrow$ **CROSSOVER**(s_1, s_2)
      Sample $r_1, r_2 \sim U[0, 1]$
      if $r_1 < p_m$ **then**
        Perform mutation $s_1 \leftarrow$ **MUTATION**(s_1)
      if $r_2 < p_m$ **then**
        Perform mutation $s_2 \leftarrow$ **MUTATION**(s_2)
      Add two strings to new population $P_{new} \leftarrow P_{new} \bigcup \{s_1, s_2\}$
    else
      Sample $r \sim U[0, 1]$
      if $r_1 < p_m$ **then**
        Perform mutation $s_1 \leftarrow$ **MUTATION**(s_1)
      Add string to new population $P_{new} \leftarrow P_{new} \bigcup \{s_1\}$
  return New population P_{new}



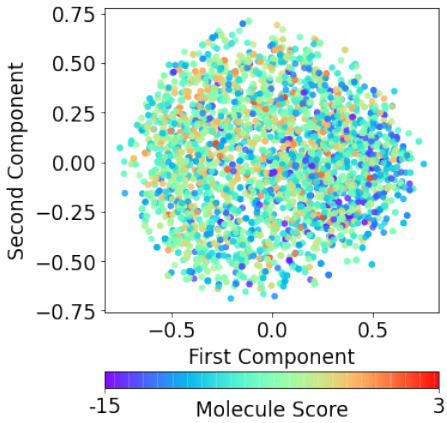
(a) $(\ell, m) = (30, 1)$ (b) $(\ell, m) = (186, 2)$

(c) $(\ell, m) = (360, 8)$ (d) $(\ell, m) = (3672, 64)$

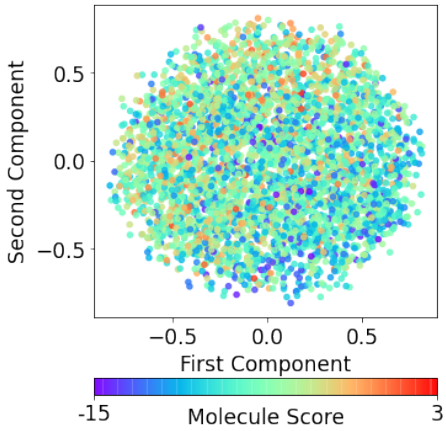
Figure B.4.9: Optimisation performance and computational overhead when finding the representation with minimal *minimum free-folding energy* (MFE) of a protein of length ℓ. SSKs are applied to codon or base representations split into m or $3m$ parts, respectively.



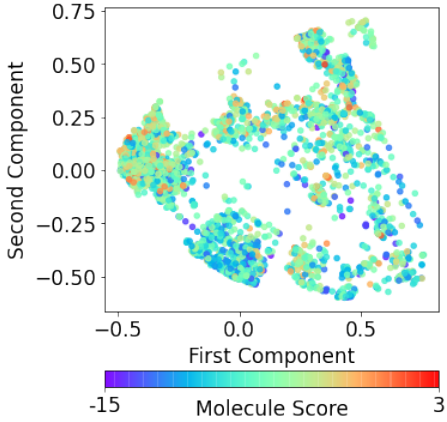
(a) SSK on raw SMILES strings.



(b) SE kernel in the *CVAE* latent space.



(c) SE kernel in the *GVAE* latent space.



(d) SSK with poor choices of kernel parameters.

Figure B.4.10: Top two KPCA components visualising the intrinsic representations of the surrogate models used to predict molecule scores from SMILES strings. Aside from (d), kernel parameters are tuned to maximise GP likelihood over 10 evaluated molecules.

Appendix C

Supplementary Material for GIBBON

C.1 Extracting The Required Predictive Quantities from a Gaussian Process Surrogate Model

We now demonstrate how the distributional quantities required to calculate GIBBON can easily be extracted from a GP surrogate model. For observations D_n, let $\mathbf{y}_n$ be the already observed evaluations y , and define the kernel matrix $\mathbf{K}_n = [k(\mathbf{z}_i, \mathbf{z}_j)]_{\mathbf{z}_i, \mathbf{z}_j \in D_n}$ and kernel vectors $\mathbf{k}_n(\mathbf{z}) = [k(\mathbf{z}_i, \mathbf{z})]_{\mathbf{z}_i \in D_n}$ for any valid kernel defined over the combined search space $\mathcal{Z} = \mathcal{X} \times \mathcal{F}$. Finally, denote the location in the fidelity space corresponding to the true objective function as $\mathbf{s}_0$ (i.e $f_{\mathbf{s}_0}(\mathbf{x}) = g(\mathbf{x})$). Here, as is standard in multi-fidelity optimisation, we have assumed the ability to query (at least nosily) the true objective function. Then, following Rasmussen (2004a) our GP surrogate model provides the following:

$$
\begin{aligned}
\mu_i^C =& \mathbf{k}_n((\mathbf{x}_i, \mathbf{s}_0))^T (\mathbf{K}_n + \operatorname{diag}(\boldsymbol{\sigma}_n))^{-1} \mathbf{y}_n \\
\mu_i^A =& \mathbf{k}_n(\mathbf{z}_i)^T (\mathbf{K}_n + \operatorname{diag}(\boldsymbol{\sigma}_n))^{-1} \mathbf{y}_n \\
\Sigma_{i,j}^C =& k((\mathbf{x}_i, \mathbf{s}_0), (\mathbf{x}_j, \mathbf{s}_0)) - \mathbf{k}_n((\mathbf{x}_i, \mathbf{s}_0))^T (\mathbf{K}_n + \operatorname{diag}(\boldsymbol{\sigma}_n))^{-1} \mathbf{k}_n((\mathbf{x}_j, \mathbf{s}_0)) \\
\Sigma_{i,j}^A =& k(\mathbf{z}_i, \mathbf{z}_j) - \mathbf{k}_n(\mathbf{z}_i)^T (\mathbf{K}_n + \operatorname{diag}(\boldsymbol{\sigma}_n))^{-1} \mathbf{k}_n(\mathbf{z}_j) \\
\rho_i =& \frac{k(\mathbf{z}_i, (\mathbf{x}_i, \mathbf{s}_0)) - \mathbf{k}_n(\mathbf{z}_i)^T (\mathbf{K}_n + \operatorname{diag}(\boldsymbol{\sigma}_n))^{-1} \mathbf{k}_n((\mathbf{x}_i, \mathbf{s}_0)))}{\sqrt{\Sigma_{i,i}^g \Sigma_{i,i}^y}},
\end{aligned}
$$

where $\text{diag}(\boldsymbol{\sigma_n})$ is the $|D_n| \times |D_n|$ diagonal matrix of observation noises in the evaluations D_n.

C.2 Proof of Theorem 5.4.1

Theorem 5.4.1 (Distribution of $\mathbf{A}$ given $C^* < m$)**.** *Consider two b-dimensional multivariate Gaussian random variables $\boldsymbol{A}$ and $\boldsymbol{C}$ where $\boldsymbol{C} \sim N(\boldsymbol{\mu}^C, \Sigma^C)$ and each individual component of $\boldsymbol{A}$ is distributed as $A_j \sim N(\mu_j^A, \Sigma_{j,j}^A)$. Suppose further that each pair $\{A_j, C_j\}$ are jointly Gaussian with correlation ρ_j, and that each A_j is conditionally independent of $\{C_i\}_{i \neq j}$ given C_j. Define $C^* = \max \boldsymbol{C}$. Then the conditional density of $\boldsymbol{A}$ given that $C^* < m$ is given by*

$$\frac{1}{\mathbb{P}(C^* < m)} \phi_{\boldsymbol{Z}_1}(\boldsymbol{a}) \Phi_{\boldsymbol{Z}_2}(\boldsymbol{m}),$$

where $\boldsymbol{m} = (m, .., m) \in \mathbb{R}^B$ and $\phi_{\boldsymbol{Z}_1}$ and $\Phi_{\boldsymbol{Z}_2}$ are the probability density and cumulative density functions for the multivariate Gaussian random variables

$$\boldsymbol{Z}_1 \sim \boldsymbol{N}\left(\boldsymbol{\mu}^A, S + D\Sigma^C D\right) \quad \text{and} \quad \boldsymbol{Z}_2 \sim \boldsymbol{N}\left(\boldsymbol{\mu}^C + \Sigma^{-1} D S^{-1}(\boldsymbol{a} - \boldsymbol{\mu}^A), \Sigma^{-1}\right),$$

where $\Sigma^A = D\Sigma^C D + S$ for D and S, diagonal matrices with elements $D_{j,j} = \rho_j \sqrt{\frac{\Sigma_j^A}{\Sigma_{j,j}^C}}$ and $S_{j,j} = (1 - \rho_j^2)\Sigma_j^A$, and $\Sigma = \left(\left(\Sigma^C\right)^{-1} + DS^{-1}D\right)$.

Proof. As detailed in the main body of this report, we have that

$$\mathbf{C} \sim N(\boldsymbol{\mu}^C, \Sigma^C) \quad \text{and} \quad A_j \sim N_1\left(\mu_j^A, \Sigma_j^A\right),$$

for some known mean vectors $\boldsymbol{\mu}^C, \boldsymbol{\mu}^A \in \mathbb{R}^B$, a variance vector $\boldsymbol{\Sigma}^A \in \mathbb{R}^B$ and a covariance matrix $\Sigma^C \in \mathbb{R}^{B \times B}$, as well as a vector $\boldsymbol{\rho} \in \mathbb{R}^B$ of the correlation between each pair $\{A_j, C_j\}$. In this section we use $f_{\mathbf{X}}$ to denote the probability density function for the random variable $\mathbf{X}$ and $f_{\mathbf{X},\mathbf{Y}}$ to denote the joint probability density function for the random variables $\mathbf{X}$ and $\mathbf{Y}$.

Now, consider the probability distribution function of random variable of interest:

$$\begin{aligned} f_{\mathbf{A}|C^*\leq m}(\mathbf{a}) =& \frac{1}{\mathbb{P}(C^* \leq m)} \int^{\mathbf{m}} f_{\mathbf{A},\mathbf{C}}(\mathbf{a},\mathbf{b})\, d\mathbf{b} \\ =& \frac{1}{\mathbb{P}(C^* \leq m)} \int^{\mathbf{m}} f_{\mathbf{A}|\mathbf{C}=\mathbf{b}}(\mathbf{a}) f_{\mathbf{C}}(\mathbf{b})\, d\mathbf{b} \\ =& \frac{1}{\mathbb{P}(C^* \leq m)} \int^{\mathbf{m}} \prod_{i=1}^{B} \left[f_{A_i|C_i=b_i}(a_i)\right] f_{\mathbf{C}}(\mathbf{b})\, d\mathbf{b}, \end{aligned} \tag{C.2.1}$$

where $\mathbf{b} \in \mathbb{R}^B$ and $\mathbf{m} = (m, .., m) \in \mathbb{R}^B$. The factorisation of $f_{\mathbf{A}|\mathbf{C}=\mathbf{b}}$ is due to the conditional independence of $A_j|C_j$ from $\{C_i\}_{i\neq j}$.

A well-known result for the conditional distribution from a bi-variate Gaussian gives us that for each $i \in \{1, .., B\}$

$$A_i = a_i | C_i = b_i \sim N_1\left(\mu_i^A + \rho_i \sqrt{\frac{\Sigma_i^A}{\Sigma_{i,i}^C}}(b_i - \mu_i^C), (1-\rho_i^2)\Sigma_i^A\right),$$

i.e. we have that

$$\mathbf{A}|\mathbf{C} = \mathbf{b} \sim N\left(\boldsymbol{\mu}^A + D(\mathbf{b} - \boldsymbol{\mu}^C), S\right), \tag{C.2.2}$$

for diagonal matrices $D, S \in \mathbb{R}^B$ with elements $D_{i,i} = \rho_i \sqrt{\frac{\Sigma_i^A}{\Sigma_{i,i}^C}}$ and $S_{i,i} = (1-\rho_i^2)\Sigma_i^A$.

Using (C.2.2), the integrand of (C.2.1) can now be regarded as the product of two b-dimensional Gaussian densities

$$\begin{aligned} \left[\prod_{i=1}^{b} f_{A_i|C_i=b_i}(a_i)\right] f_{\mathbf{C}}(\mathbf{b}) &= N\left(\mathbf{a}; \boldsymbol{\mu}^A + D(\mathbf{b} - \boldsymbol{\mu}^C), S\right) * N(\mathbf{b}; \boldsymbol{\mu}^C, \Sigma^C) \\ &= |D| N\left(\mathbf{b}; \boldsymbol{\mu}^C + D^{-1}(\mathbf{a} - \boldsymbol{\mu}^A), D^{-1} S D^{-1}\right) * N(\mathbf{b}; \boldsymbol{\mu}^C, \Sigma^C), \end{aligned}$$

which, using the following standard formula for the product of Gaussians densities

$$\begin{aligned} N(\mathbf{x}; \mathbf{m}_1, \Sigma_1) * N(\mathbf{x}; \mathbf{m}_2, \Sigma_2) =& N(\mathbf{m}_1; \mathbf{m}_2, \Sigma_1 + \Sigma_2) \\ & * N(\mathbf{x}; \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1} \left(\sigma_1^{-1}\mathbf{m}_1 + \Sigma_2^{-1}\mathbf{m}_2\right), \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1}), \end{aligned}$$

can be re-expressed as

$$\begin{aligned}\left[\prod_{i=1}^{b} f_{A_i|C_i=b_i}(a_i)\right] f_{\mathbf{C}}(\mathbf{b}) &= |D| N\left(\boldsymbol{\mu}^C; \boldsymbol{\mu}^C + D^{-1}(\mathbf{a}-\boldsymbol{\mu}^A), D^{-1}SD^{-1} + \Sigma^C\right) \\ &\quad * N\left(\mathbf{b}; \boldsymbol{\mu}^C + \Sigma^{-1}DS^{-1}(\mathbf{a}-\boldsymbol{\mu}^A), \Sigma^{-1}\right) \\ &= N\left(\mathbf{a}; \boldsymbol{\mu}^A, S + D\Sigma^C D\right) \\ &\quad * N\left(\mathbf{b}; \boldsymbol{\mu}^C + \Sigma^{-1}DS^{-1}(\mathbf{a}-\boldsymbol{\mu}^A), \Sigma^{-1}\right)\end{aligned}$$

where $\Sigma = \left(\left(\Sigma^C\right)^{-1} + DS^{-1}D\right)$.

Therefore, we have rewritten the integrand of (C.2.1) as a product of two Gaussian densities, where only one depend on $\mathbf{b}$. Consequently, the first Gaussian term can be taken outside the integral, yielding the claimed expression

$$f_{\mathbf{A}|C^*<m}(\mathbf{a}) = \frac{1}{\mathbb{P}(C^* < m)} \phi_{\mathbf{Z}_1}(\mathbf{a}) \Phi_{\mathbf{Z}_2}(\mathbf{m}), \tag{C.2.3}$$

where $\phi_{\mathbf{Z}_1}$ and $\Phi_{\mathbf{Z}_2}$ are the probability density and cumulative density functions for the multivariate Gaussian variables

$$\mathbf{Z}_1 \sim \mathbf{N}_b\left(\boldsymbol{\mu}^A, S + D\Sigma^C D\right) \quad \text{and} \quad \mathbf{Z}_2 \sim \mathbf{N}_b\left(\boldsymbol{\mu}^C + \Sigma^{-1}DS^{-1}(\mathbf{a}-\boldsymbol{\mu}^A), \Sigma^{-1}\right).$$

□

C.3 Experimental Details for Synthetic Benchmarks.

We now provide detailed information about each of our synthetic benchmarks.

C.3.1 Standard BO benchmarks

Shekel function. A four-dimensional function with ten local and one global minima defined on $\mathcal{X} \in [0, 10]^4$:

$$f(\mathbf{x}) = -\sum_{i=1}^{10}\left(\sum_{j=1}^{4}(x_j - A_{j,i})^2 + \beta_i\right)^{-1},$$

where

$$\beta = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 4 \\ 4 \\ 6 \\ 3 \\ 7 \\ 5 \\ 5 \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3.6 \end{pmatrix}.$$

Ackley function. A four-dimensional function with many local minima and a nearly flat outer region surrounding a single global minima defined on $\mathcal{X} \in [-32.768, 32.768]^4$:

$$f(\mathbf{x}) = -20 \exp\left(-0.2 * \sqrt{\frac{1}{4}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{4}\sum_{i=1}^{4} \cos(2\pi x_i)\right) + 20 + \exp(1).$$

Hartmann 6 function. A six-dimensional function with six local minima and a single global minima defined on $\mathcal{X} \in [0, 1]^6$:

$$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} A_{i,j}(x_j - P_{i,j})^2\right),$$

where

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \qquad \alpha = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix},$$

$$P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}.$$

C.3.2 Multi-fidelity benchmarks

Currin exponential function (discrete fidelity space). A two-dimensional function defined on $\mathcal{X} = [0, 1]^2$ with two fidelities queried with costs 10 and 1:

$$\begin{aligned} f(x_1, x_2, 0) =& \left(1 - \exp(-\frac{1}{2x_2})\right) \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_1^2 + 4x_1 + 20} \\ f(x_1, x_2, 1) =& \frac{1}{4} f(x_1 + 0.05, x_2 + 0.05, 0) \\ &+ \frac{1}{4} f(x_1 + 0.05, max(0, x_2 - 0.05), 0) \\ &+ \frac{1}{4} f(x_1 - 0.05, x_2 + 0.05, 0) \\ &+ \frac{1}{4} f(x_1 - 0.05, max(0, x_2 - 0.05), 0). \end{aligned}$$

Hartmann 3 function. A three-dimensional function with 4 local extrema defined on $\mathcal{X} = [0, 1]^3$ with three fidelities ($m = 0, 1, 2$) queried at costs $100, 10$ and 1:

$$f(x_1, x_2, x_3, m) = -\sum_{i=1}^{4} \alpha_{i,m+1} \exp\left(-\sum_{j=1}^{3} A_{i,j}(x_j - P_{i,j})^2\right),$$

where

$$A = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}, \quad \alpha = \begin{pmatrix} 1 & 1.01 & 1.02 \\ 1.2 & 1.19 & 1.18 \\ 3 & 2.9 & 2.8 \\ 3.2 & 3.3 & 3.4 \end{pmatrix}, \quad P = \begin{pmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{pmatrix}.$$

Hartmann 6 function. A six-dimensional function defined on $\mathcal{X} = [0,1]^6$ with four fidelities ($m = 0, 1, 2, 3$) queried at costs $1000, 100, 10$ and 1:

$$f(x_1, x_2, x_3, x_4, x_5, x_6, m) = -\sum_{i=1}^{4} \alpha_{i,m+1} \exp\left(-\sum_{j=1}^{6} A_{i,j}(x_j - P_{i,j})^2\right),$$

where

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, \qquad \alpha = \begin{pmatrix} 1 & 1.01 & 1.02 & 1.03 \\ 1.2 & 1.19 & 1.18 & 1.17 \\ 3 & 2.9 & 2.8 & 2.7 \\ 3.2 & 3.3 & 3.4 & 3.5 \end{pmatrix},$$

$$P = 10^{-4} \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}.$$

Borehole function. An eight-dimensional function defined on

$$\mathcal{X} = [0.05, 0.15; 100, 50,000; 63070, 115600; 990,$$
$$1110; 63.1, 116; 700, 820; 1120, 1680; 9855, 12055]$$

with two fidelities queried with costs 10 and 1:

$$f(\mathbf{x}, 0) = \frac{2\pi x_3(x_4 - x_6)}{\log(x_2/x_1)\left(1 + \frac{2x_7x_3}{log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right)},$$
$$f(\mathbf{x}, 1) = \frac{5x_3(x_4 - x_6)}{\log(x_2/x_1)\left(1.5 + \frac{2x_7x_3}{log(x_2/x_1)x_1^2x_8} + \frac{x_3}{x_5}\right)}.$$

C.4 Comparing GIBBON with MES

In our synthetic experiments of Section 5.7, we were surprised to see that GIBBON was able to outperform MES even in the noiseless standard BO case for which MES provides an exact calculation of entropy reductions. As GIBBON approximates MES, we actually expected GIBBON to perform strictly worse than MES in this particular setting. We posit that the high-performance of GIBBON is due to it enjoying an easier inner-loop maximisation, permitting higher-precision maximisation under the same acquisition maximisation budget. We now explore this hypothesis.

To gain further intuition about why GIBBON's acquisition function is easier to optimise, we analyse the so-called degenerate forms of MES and GIBBON where the acquisition functions are built using only a single max-value sample. By defining the function $u(\mathbf{x}) = \frac{m^* - \mu_n^g(\mathbf{x})}{\sqrt{\Sigma^g(\mathbf{x})}}$, degenerate GIBBON and MES can be expressed as

$$\alpha_n^{\text{GIBBON}}(\mathbf{x}) = -\log\left(1 - \frac{\phi(u(\mathbf{x}))}{\Phi(u(\mathbf{x}))}\left(u(\mathbf{x}) + \frac{\phi(u(\mathbf{x}))}{\Phi(u(\mathbf{x}))}\right)\right)$$
$$\alpha_n^{\text{MES}}(\mathbf{x}) = \frac{u(\mathbf{x})\phi(u(\mathbf{x}))}{2\Phi(u(\mathbf{x}))} - \log\Phi(u(\mathbf{x})).$$

Although taking very different analytical forms, these two acquisition functions are strictly decreasing in u (as shown in Figure C.4.1), with GIBBON taking larger values in promising locations of the space. So although (in this degenerate and noiseless setting) GIBBON and MES would choose exactly the same points under given exact inner-loop maximisation, GIBBON's larger values around its local maxima make high-precision inner-loop maximisation easier.

Note that in this limited setting, Wang and Jegelka (2017) provide a bound on the simple regret of degenerate MES. As degenerate GIBBON and degenerate MES choose the same query points, this regret bound is also inherited by degenerate GIBBON. Although this result does not hold for practical implementations of GIBBON, where we typically use 5 or 10 samples of g^*, or when we perform batch or multi-fidelity BO, the existence of this theoretical guarantee provides reassuring evidence for the validity of our approach.

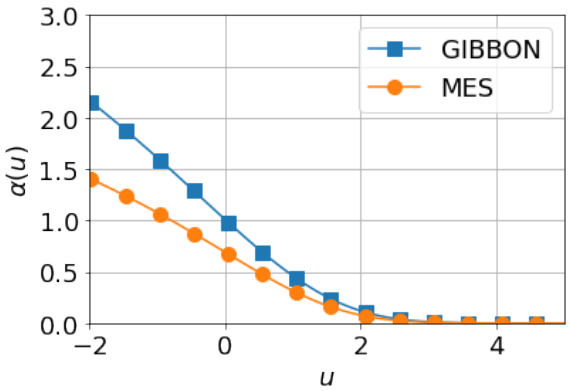


Figure C.4.1: Degenerate GIBBON and MES as functions of u. The two acquisition functions are monotonically decreasing, with GIBBON taking much larger values.

Appendix D

Supplementary Material for BOSH

D.1 Suboptimality of Tuning a Fixed Evaluation Strategy

We now derive the expected suboptimality of optimising a fixed evaluation strategy instead of the true objective function, following the notation defined in Section 6.2. For ease of understanding, we just consider a single dimensional optimisation problem. However, a similar (but less intuitive) expression can be derived for optimisation over multi-dimensional search spaces.

We wish to compare x^*, the optimiser of true model performance, with x^*_S, the optimiser of the evaluation strategy based on a collection of K random train-test splits $S = \{s_1, .., s_K\}$ (a random variable).

Estimators of this form are well-studied in the robust statistics literature, and are known as M-estimators (see Hampel et al., 2011, for a summary). Under some non-restrictive assumptions, which we state shortly, these estimators are known to be approximately normally distributed

$$x^*_S \sim \mathcal{N}(x^*, \frac{\sigma^2}{K}),$$

where $\sigma^2 = \mathrm{Var_s}(x^*_{\{s\}})$ represents the variability in optimisers chosen according to evaluation strategies based on single random seeds. We have assumed that x^* is unique,

that the space of feasible hyper-parameter values is a compact set and that $f_{s_i}(\mathbf{x})$ are continuous, twice differential and uniformly bounded.

Now, after performing a first order Taylor expansion, we have

$$\mathbb{E}_S\left[g(x_S^*)\right] \approx g(x^*) + \frac{g''(x^*)}{2}\frac{\sigma^2}{K},$$

providing

$$\mathbb{E}_S\left[g(x^*) - g(x_S^*)\right] \approx \frac{|g''(x^*)|}{2K}\sigma^2, \tag{D.1.1}$$

where $g''(x^*)$ is the second derivative of the true objective function at its maximum (measuring the stability of our objective function around its optimum).

(D.1.1) represents the stability of the objective function as we move away from its optimiser. Therefore this expression explicitly relates the reliability of our optimisation with our chosen evaluation strategy (through the choice of K) and shows that there is a minimum threshold for K that ensures reliable optimisation to any chosen level of precision. Optimising an evaluation strategy based on fewer seeds than this threshold will likely just over-fit to our evaluation strategy rather than providing the desired precision. In other words, we do not want to waste resources finding the location $\mathbf{x}_S^*$ to a higher precision than $|\mathbf{x}^* - \mathbf{x}_S^*|$.

D.2 Predictive Distribution of an HGP

In Section 6.5.1 we defined our HGP model

$$\begin{aligned} g(\mathbf{x}) &\sim \mathcal{GP}(0, k_g) \\ f_s(\mathbf{x}) &\sim \mathcal{GP}(g(\mathbf{x}), k_f) \\ y_s(\mathbf{x}) &= f_s(\mathbf{x}) + \epsilon. \end{aligned}$$

As demonstrated by Hensman et al. (2013), this formulation is equivalent to assuming a prior co-variance of

$$\begin{aligned} Cov(f_s(\mathbf{x}), f_{s'}(\mathbf{x}'))) &= k_g(\mathbf{x}, \mathbf{x}') + \mathbb{I}_{s=s'} k_f(\mathbf{x}, \mathbf{x}') \\ Cov(f_s(\mathbf{x}), g(\mathbf{x}')) &= k_g(\mathbf{x}, \mathbf{x}') \end{aligned} \tag{D.2.1}$$

We will now provide closed form expressions for the joint predictive distributions of $g(\mathbf{x})$ and $y_s(\mathbf{x})$ given a set of collected evaluations $D_n = \{(\mathbf{x}_i, s_i, y_i)\}_{i=1}^n$, where $y_i = f_{s_i}(\mathbf{x}_i) + \epsilon$ under Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

Defining a compound kernel $\tilde{k}$ (defined over $\mathcal{X} \times S$) as $\tilde{k}((\mathbf{x}, s), (\mathbf{x}', s')) = k_g(\mathbf{x}, \mathbf{x}') + \mathbb{I}_{s=s'} k_f(\mathbf{x}, \mathbf{x}')$ and following Rasmussen (2004a) and Hensman et al. (2013), our joint posterior distribution can be written as

$$\begin{aligned} &\begin{pmatrix} g(\mathbf{x}) \\ y_s(\mathbf{x}) \end{pmatrix} \Bigg| D_n \\ &\sim \quad N\left[\begin{pmatrix} \mu_n^g(\mathbf{x}) \\ \mu_n(\mathbf{x}, s) \end{pmatrix}, \begin{pmatrix} \sigma_n^{g2}(\mathbf{x}) & \Sigma_n(\mathbf{x}, s) \\ \Sigma_n(\mathbf{x}, s) & \sigma_n^2(\mathbf{x}, s) + \sigma^2 \end{pmatrix} \right], \end{aligned} \tag{D.2.2}$$

where

$$\begin{aligned} \mu_n(\mathbf{x}, s) =& \tilde{\mathbf{k}}_n((\mathbf{x}, s))^T \left(\tilde{\mathbf{K}}_n + \sigma^2 I_n\right)^{-1} \mathbf{y}_n \\ \mu_n^g(\mathbf{x}) =& \mathbf{k}_n^g((\mathbf{x}, s))^T \left(\tilde{\mathbf{K}}_n + \sigma^2 I_n\right)^{-1} \mathbf{y}_n \\ \sigma_n^2(\mathbf{x}, s) =& \tilde{k}\left((\mathbf{x}, s), (\mathbf{x}, s)\right) \\ & - \tilde{\mathbf{k}}_n((\mathbf{x}, s))^T \left(\tilde{\mathbf{K}}_n + \sigma^2 I_n\right)^{-1} \tilde{\mathbf{k}}_n((\mathbf{x}, s)) \\ \sigma_n^{g2}(\mathbf{x}) =& k^g\left(\mathbf{x}, \mathbf{x}\right) - \mathbf{k}_n^g(\mathbf{x})^T \left(\tilde{\mathbf{K}}_n + \sigma^2 I_n\right)^{-1} \mathbf{k}_n^g(\mathbf{x}) \\ \Sigma_n(\mathbf{x}, s) =& k^g\left((\mathbf{x}, s), (\mathbf{x}, s)\right) \\ & - \tilde{\mathbf{k}}_n((\mathbf{x}, s))^T \left(\tilde{\mathbf{K}}_n + \sigma^2 I_n\right)^{-1} \mathbf{k}_n^g(\mathbf{x}), \end{aligned}$$

for $\tilde{\mathbf{K}}_n = \left[\tilde{k}((\mathbf{x}_i, s_i), (\mathbf{x}_j, s_j))\right]_{i,j=1,..,n}$, $\tilde{\mathbf{k}}_n((\mathbf{x}, s)) = \left[\tilde{k}((\mathbf{x}_i, s_i), (\mathbf{x}, s))\right]_{i=1,..,n}$,$\mathbf{k}_n^g(\mathbf{x}) = [k_g(\mathbf{x}_i, \mathbf{x})]_{i=1,..n}$ and $\mathbf{y} = [y_i]_{i=1,..,n}$.

Similarly, we also have a predictive covariance between evaluations on different seeds as

$$\begin{aligned} V_n((\mathbf{x}, s), (\mathbf{x}', s')) = & \tilde{k}\left((\mathbf{x}, s), (\mathbf{x}', s')\right) \\ & - \tilde{\mathbf{k}}_n((\mathbf{x}, s))^T \left(\tilde{\mathbf{K}}_n + \sigma^2 I_n\right)^{-1} \tilde{\mathbf{k}}_n(\mathbf{x}', s'), \end{aligned} \tag{D.2.3}$$

Note that predicting from our HGP requires the inversion of the $n \times n$ matrix $\tilde{\mathbf{K}}_n + \sigma^2 I_n$ and so has comparable cost to predictions from standard GPs.

D.3 Experimental Details

We now provide additional details about our implementation of BOSH and the exact set-ups of our experiments. Experimental code implementing BOSH through the Emukit [1] Python package for these examples is available at *redacted for review.*

HGP Kernel. Our implementation of BOSH uses the following structure for the upper and lower kernels:

$$k_g(\mathbf{x}, \mathbf{x}') = k_{\alpha_g,\beta}(\mathbf{x}, \mathbf{x}')$$
$$k_f(\mathbf{x}, \mathbf{x}') = k_{\alpha_f,\beta}(\mathbf{x}, \mathbf{x}') + \sigma_f^2,$$

where $k_{\alpha,\beta}$ denotes the Matérn 5/2 (Matérn, 1960) kernel with variance $\alpha \in \mathbb{R}$ term and length scales $\beta \in \mathbb{R}^d$ hyper-parameters, i.e

$$k_{\alpha,\beta}(\mathbf{x}, \mathbf{x}') = \alpha(1 + \sqrt{5}d_\beta(\mathbf{x}, \mathbf{x}') + \frac{5}{3}d_\beta(\mathbf{x}, \mathbf{x}')^2)e^{-\sqrt{5}d_\beta(\mathbf{x},\mathbf{x}')},$$

for a weighted distance measure $d_\beta(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T diag(\beta)(\mathbf{x} - \mathbf{x}')$.

As the length-scales are shared between the lower and upper kernels, the total number of kernel parameters for BOSH (including the scale of observation noise σ^2in our Gaussian likelihood) is $d + 4$, only two more than a standard GP with a Matérn 5/2 kernel.

D.3.1 Reinforcement Learning: Lunar Lander

The Lunar Lander problem is a well-known reinforcement learning task, where we must control three engines (left, main and right) to successfully land a rocket. The learning environment and a hard-coded PID controller is provided in the OpenAI gym [2]. We seek to optimise the 7 thresholds present in the description of the controller to provide the largest average reward over 100 random initial conditions. Our RL environment is exactly as provided by OpenAI, with the small modification of randomly initialising the initial lander location (as-well as random initial velocities and terrain) to make

[1] *https://github.com/amzn/emukit*

[2] *https://gym.openai.com/*

a more challenging stochastic optimisation problem. We lose 0.3 points per second of fuel use and 100 if we crash. We gain 10 points each time a leg makes contact with the ground, 100 points for any successful landing, and 200 points for a successful landing in the specified landing zone· Each individual run of the environment allows the testing of a controller on a specific random seed.

D.3.2 Hyper-parameter Tuning: IMDB SVM

We tested the performance of BOSH on a real ML problem: tuning a sentiment classification model on the collection of $25,000$ positive and $25,000$ negative IMDB movie reviews used by Maas et al. (2011), seeking the hyper-parameter values that provide the model with the highest accuracy. We tune the flexibility of the decision boundary (C) and the RBF kernel coefficient (gamma) for an SVM (Cortes and Vapnik, 1995), a standard model for binary text classification. As is common in the natural language processing literature, we train our classifier on a bag-of-words representation of the data (Jurafsky and Martin, 2014), using tf-idf weightings (Salton and Buckley, 1988). In order to measure the true performance of tuned hyper-parameters, we must use the available data in an unconventional way. By restricting our model fitting and tuning to a randomly sub-sampled $1,000$ review subset to act as our training set for all our experiments, we provide a large held-out collection of $49,000$ movie reviews, upon which we can calculate the 'true' performance of the hyper-parameter configurations chosen by our tuning algorithms. We then randomly draw our train-test splits from this fixed training set, with test sets of 10%. As already argued, the model scores based on a particular evaluation strategy do not necessarily correspond to the true performance and so, although we acknowledge that this contrived use of the data is not standard, this set-up is necessary to measure the improved efficiency and reliability provided by BOSH.

D.3.3 Hyper-parameter Tuning: Movie-lens PMF

For our second hyper-parameter tuning task we consider tuning the learning rate across $[0, 0.01]$, regularisation strength across $[0, 0.1]$, matrix rank across $[50, .., 150]$ and number of model epochs across $[10, .., 50]$ for a probabilistic matrix factorisation (MPF) (Mnih and Salakhutdinov, 2008) recommendation system on the well-known Movie-lens-100k Hoffman et al. (2010) data (using the Surprise[3] Python library). Unlike the IMDB data-set, we consider the whole of the $100,000$ data points for training (to be used for train-test splits and K-fold CV). As we do not have any held-out data to calculate true performance, we instead use the average performance estimates across 20 train-test splits to reliably (albeit expensively) measure true performance in terms of mean reconstruction error (a standard metric for recommendation systems).

D.3.4 Simulation Optimisation: Facility Allocation

Our final example considers the problem of optimally allocating two warehouses on a unit square (each edge corresponds to 30km), with a search space consisting of the x and y locations for each warehouse $(x_1, y_1, x_2, y_2) \in [0, 1]^4$. We are interested in maximising the proportion of orders delivered in under 60 minutes. The only way to estimate the performance of a particular facility allocation is to simulate demand for a particular day and seeing how each configuration copes. More reliable performance estimates can be obtained by running multiple days of simulations. For most real-world simulation optimisation problems, the cost of running a single simulation is substantial and so there there is a need to find the optimum allocation whilst running as few simulations as possible. In our experiments we run the exact implementation provided as part of the SimOpt[4] test-bed of simulation optimisation problems, originally proposed by Pasupathy and Ghosh (2013).

Orders are assumed to arise during working hours (8AM to 5PM) at a rate of 0.3 per minute with the exact delivery location (x, y) controlled by a density function proportional to $1.6 - (|x - 0.8| + |y - 0.8|)$. Each warehouse has 10 trucks that can

[3] *http://surpriselib.com/*

[4] *http://simopt.org/*

carry and deliver a single order. Any orders for which there are no available trucks are queued and those in a queue at the end of the day must still be delivered. Time taken to load and unload the trucks are assumed to be distributed exponentially with means 5 and 10. Trucks can travel only in the x or y direction at a speed of $30kmh$.

Bibliography

Mohammed AlQuraishi. Alphafold at CASP13. *Bioinformatics*, 2019.

Ahsan S Alvi, Binxin Ru, Jan Calliess, Stephen J Roberts, and Michael A Osborne. Asynchronous batch bayesian optimisation with improved local penalisation. In *International Conference on Machine Learning*, 2019.

Eric Anderson, Gilman D Veith, and David Weininger. SMILES, a line notation and computerized interpreter for chemical structures. *Environmental Research Laboratory*, 1987.

Reinaldo B Arellano-Valle, Javier E Contrera-Reyes, and Marc G Genton. Shannon entropy and mutual information for multivariate skew-elliptical distributions. *Scandinavian Journal of Statistics*, 2013.

Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. In *Advances in Neural Information Processing Systems*, 2018.

Barry C Arnold, Robert J Beaver, Richard A Groeneveld, and William Q Meeker. The nontruncated marginal of a truncated bivariate normal distribution. *Psychometrika*, 1993.

Adelchi Azzalini. A class of distributions which includes the normal ones. *Scandinavian Journal of Statistics*, 1985.

Adelchi Azzalini and A Dalla Valle. The multivariate skew-normal distribution. *Biometrika*, 1996.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: programmable Bayesian optimization in PyTorch. In *Neural Information Processing Systems*, 2020.

Daniel Beck and Trevor Cohn. Learning kernels over strings using Gaussian processes. In *International Joint Conference on Natural Language Processing*, 2017.

Daniel Beck, Trevor Cohn, Christian Hardmeier, and Lucia Specia. Learning structural kernels for natural language processing. *Transactions of the Association for Computational Linguistics*, 2015.

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2019.

Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Max-value entropy search for multi-objective Bayesian optimization with constraints. *arXiv preprint arXiv:2009.01721*, 2020.

Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 2004.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 2012.

James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *Journal of Machine Learning Research*, 2013.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2017.

Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task Gaussian process prediction. In *Neural Information Processing Systems*, 2008.

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. Word-sequence kernels. *Journal of Machine Learning Research*, 2003.

D-S Cao, J-C Zhao, Y-N Yang, C-X Zhao, J Yan, S Liu, Q-N Hu, Q-S Xu, and Y-Z Liang. In silico toxicity prediction by support vector machine and SMILES representation-based string kernel. *SAR and QSAR in Environmental Research*, 2012.

Paruchuri Chaitanya and Pratibha Vellanki. Bayesian optimisation for low-noise aerofoil design with aerodynamic constraints. *International Journal of Aeroacoustics*, 2020.

Yutian Chen, Yannis Assael, Brendan Shillingford, David Budden, Scott Reed, Heiga Zen, Quan Wang, Luis C Cobo, Andrew Trask, Ben Laurie, et al. Sample efficient adaptive text-to-speech. *arXiv preprint arXiv:1809.10460*, 2018a.

Yutian Chen, Aja Huang, Ziyu Wang, Ioannis Antonoglou, Julian Schrittwieser, David Silver, and Nando de Freitas. Bayesian optimization in alphago. *arXiv preprint arXiv:1812.06855*, 2018b.

Clément Chevalier and David Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In *International Conference on Learning and Intelligent Optimization*, 2013.

Yu-An Chung, Yuxuan Wang, Wei-Ning Hsu, Yu Zhang, and RJ Skerry-Ryan. Semi-supervised training for improving data efficiency in end-to-end speech synthesis. In *Internal Conference on Acoustics, Speech and Signal Processing*, 2019.

Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems*, 2002.

Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 1995.

Thomas M Cover and Joy A Thomas. *Elements of information theory*, 2012.

Kurt Cutajar, Mark Pullin, Andreas Damianou, Neil Lawrence, and Javier González. Deep Gaussian processes for multi-fidelity modeling. *arXiv preprint arXiv:1903.07320*, 2019.

Steven J Daniels, Alma AM Rahat, Richard M Everson, Gavin R Tabor, and Jonathan E Fieldsend. A suite of computationally expensive shape optimisation problems using computational fluid dynamics. In *International Conference on Parallel Problem Solving from Nature*, 2018.

Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 2012.

Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Mercer features for efficient combinatorial Bayesian optimization. *arXiv preprint arXiv:2012.07762*, 2020.

Jesse Dodge, Kevin Jamieson, and Noah A Smith. Open loop hyperparameter optimization and determinantal point processes. *arXiv preprint arXiv:1706.01566*, 2017.

Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *Neural Information Processing Systems: Workshop on Bayesian Optimization*, 2013.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machhine Learning*, 2018.

Alexander Forrester, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*, 2008.

Peter I Frazier and Jialei Wang. Bayesian optimization for materials design. In *Information Science for Materials Discovery and Design*, 2016.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 2008.

Eduardo C Garrido-Merchán and Daniel Hernández-Lobato. Predictive entropy search for multi-objective Bayesian optimization with constraints. *Neurocomputing*, 2019.

Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech. In *Advances in Neural Information Processing Systems*, 2017.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems*, 2012.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 2018.

Chengyue Gong, Jian Peng, and Qiang Liu. Quantile stein variational gradient descent for batch Bayesian optimization. In *International Conference on Machine Learning*, 2019.

Javier González, Joseph Longworth, David C James, and Neil D Lawrence. Bayesian optimization for synthetic gene design. In *Advances in Neural Information Processing Systems: Bayesian Optimization Workshop*, 2014.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch Bayesian optimization via local penalization. In *Artificial intelligence and statistics*, 2016a.

Javier González, Michael Osborne, and Neil Lawrence. Glasses: Relieving the myopia of Bayesian optimisation. In *Artificial Intelligence and Statistics*, 2016b.

Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design. *Chem. Sci.*, 2020.

László Györfi, Michael Kohler, Adam Krzyzak, and Harro Walk. *A distribution-free theory of nonparametric regression*, 2006.

Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust Statistics: The Approach Based on Influence Functions*, 2011.

David Haussler. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz, 1999.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 2012.

James Hensman, Neil D Lawrence, and Magnus Rattray. Hierarchical Bayesian modelling of gene expression time series across irregularly sampled replicates and clusters. *BMC Bioinformatics*, 2013.

Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective Bayesian optimization. In *International Conference on Machine Learning*, 2016.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Neural Information Processing Systems*, 2014.

José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017.

William J Hill and William G Hunter. A review of response surface methodology: a literature survey. *Technometrics*, 1966.

Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, 2010.

Matthew W Hoffman and Zoubin Ghahramani. Output-space predictive entropy search for flexible global optimization. In *Advances in Neural Information Processing Systems: Bayesian Optimization Workshop*, 2015.

John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 2001.

Frédéric Hourdin, Thorsten Mauritsen, Andrew Gettelman, Jean-Christophe Golaz, Venkatramani Balaji, Qingyun Duan, Doris Folini, Duoying Ji, Daniel Klocke, Yun Qian, et al. The art and science of climate model tuning. *Bulletin of the American Meteorological Society*, 2017.

Deng Huang, Theodore T Allen, William I Notz, and Ning Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, 2006.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, 2011.

Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Advances in Neural Information Processing Systems*, 2018.

Shali Jiang, Henry Chai, Javier Gonzalez, and Roman Garnett. Binoculars for efficient, nonmyopic sequential experimental design. In *International Conference on Machine Learning*, 2020.

Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 1993.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 1998.

Kai Junge, Josie Hughes, Thomas George Thuruthel, and Fumiya Iida. Improving robotic cooking using batch bayesian optimization. *IEEE Robotics and Automation Letters*, 2020.

Dan Jurafsky and James H Martin. *Speech and Language Processing*, 2014.

Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. In *The International Conference on Machine Learning*, 2019.

Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems*, 2016.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In *The International Conference in Machine Learning*, 2017.

Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabás Póczos. Parallelised Bbayesian optimisation via Thompson sampling. In *The International Conference on Artificial Intelligence and Statistics*, 2018a.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with Bayesian optimization and optimal transport. In *Advances in Neural Information Processing Systems*, 2018b.

Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched Gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, 2016.

Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 2000.

Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic Gaussian process regression. In *The International Conference on Machine Learning*, 2007.

Sujin Kim, Raghu Pasupathy, and Shane G Henderson. A guide to sample average approximation. In *Handbook of simulation optimization*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

Jack PC Kleijnen. Kriging metamodeling in simulation: A review. *European journal of operational research*, 2009.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *The International Conference on Artificial Intelligence and Statistics*, 2017a.

Aaron Klein, Stefan Falkner, Numair Mansur, and Frank Hutter. RoBo: A flexible and robust Bayesian optimization framework in Python. In *Advances in Neural Information Processing Systems: Bayesian Optimization Workshop*, 2017b.

Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*,2002.

Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 1995.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, 1995.

Egor Kraev. Grammars and reinforcement learning for molecule optimization. *arXiv preprint arXiv:1811.11222*, 2018.

Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 2012.

Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *The International Conference on Machine Learning*, 2017.

Rémi Lam, Douglas L Allaire, and Karen E Willcox. Multifidelity optimization using statistical surrogate modeling for non-hierarchical information sources. In *Structures, Structural Dynamics, and Materials*, 2015.

Javier Latorre, Jakub Lachowicz, Jaime Lorenzo-Trueba, Thomas Merritt, Thomas Drugman, Srikanth Ronanki, and Viacheslav Klimkov. Effect of data reduction on sequence-to-sequence neural tts. In *The International Conference on Acoustics, Speech and Signal Processing*, 2019.

Loic Le Gratiet and Josselin Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 2014.

Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu. Deep speaker: an end-to-end neural speaker embedding system. *arXiv:1705.02304*, 2017.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2002.

Ronny Lorenz, Stephan H Bernhart, Christian Höner Zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. Viennarna package 2.0. *Algorithms for Molecular Biology*,2011.

Jaime Lorenzo-Trueba, Thomas Drugman, Javier Latorre, Thomas Merritt, Bartosz Putrycz, and Roberto Barra-Chicote. Robust universal neural vocoding. *arXiv:1811.06292*, 2018.

Xiaoyu Lu, Javier González, Zhenwen Dai, and Neil Lawrence. Structured variationally auto-encoded optimization. In *The International Conference on Machine Learning*, 2018.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Association for computational linguistics*, 2011.

MacKay, David JC. Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. In *Network: computation in neural systems*, 1995.

Benjamin P MacLeod, Fraser GL Parlane, Thomas D Morrissey, Florian Häse, Loïc M Roch, Kevan E Dettelbach, Raphaell Moreira, Lars PE Yunker, Michael B Rooney, Joseph R Deeth, et al. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 2020.

Sébastien Marmin, Clément Chevalier, and David Ginsbourger. Differentiating the multipoint expected improvement for optimal batch design. In *The International Workshop on Machine Learning, Optimization and Big Data*, 2015.

Bertil Matérn. Spatial variation, *Lecture Notes in Statistics*, 1960.

Robert I Mckay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'neill. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 2010.

Mark McLeod, Michael A Osborne, and Stephen J Roberts. Practical Bayesian optimization for variable cost objectives. *arXiv preprint arXiv:1703.04335*, 2017.

Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, 2008.

Jonas Mockus. *Bayesian approach to global optimization: theory and applications*, 2012.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards global optimization*,1978.

Henry Moss, David Leslie, and Paul Rayson. Using J-K-fold cross validation to reduce variance when tuning NLP models. In *The International Conference on Computational Linguistics*, 2018.

Henry Moss, Andrew Moore, David Leslie, and Paul Rayson. FIESTA: Fast identification of state-of-the-art models using adaptive bandit algorithms. In *The Annual Meeting of the Association for Computational Linguistics*, 2019.

Henry B Moss and Ryan-Rhys Griffiths. Gaussian process molecule property prediction with flowmo. *Advances in Neural Information Processing: Machine Learning for Molecules Workshop*, 2020.

Henry B Moss, Vatsal Aggarwal, Nishant Prateek, Javier González, and Roberto Barra-Chicote. Boffin tts: Few-shot speaker adaptation by Bayesian optimization. In *The International Conference on Acoustics, Speech and Signal Processing*, 2020a.

Henry B. Moss, Daniel Beck, Javier Gonzalez, David L. Leslie, and Paul. Rayson. Boss: Bayesian optimisation over string spaces. In *Advances in neural information processing systems*, 2020b.

Henry B Moss, David S Leslie, and Paul Rayson. Bosh: Bayesian optimization by sampling hierarchically. *The International Conference on Machine Learning: Workshop on Real World Experimental Design and Active Learning*, 2020c.

Henry B Moss, David S Leslie, and Paul Rayson. Mumbo: Multi-task max-value Bayesian optimization. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2020d.

Eliya Nachmani, Adam Polyak, Yaniv Taigman, and Lior Wolf. Fitting new speakers based on a short untranscribed sample. *arXiv:1802.06984*, 2018.

Nhu-Van Nguyen, Seok-Min Choi, Wan-Sub Kim, Jae-Woo Lee, Sangho Kim, Daniel Neufeld, and Yung-Hwan Byun. Multidisciplinary unmanned combat air vehicle system design using multi-fidelity model. *Aerospace Science and Technology*, 2013.

Anthony O'Hagan. A Markov property for covariance structures. *Statistics Research Report*, 1998.

Andrei Paleyes, Mark Pullin, Maren Mahsereci, Neil Lawrence, and Javier González. Emulation of physical processes with emukit. In *Advances in Neural Processing Systems: Workshop on Machine Learning and the Physical Sciences*, 2019.

Seongeon Park, Jonggeol Na, Minjun Kim, and Jong Min Lee. Multi-objective Bayesian optimization of chemical reactor design using computational fluid dynamics. *Computers & Chemical Engineering*, 2018.

Raghu Pasupathy and Soumyadip Ghosh. Simulation optimization: A concise overview and implementation guide. In *Theory Driven by Influential Applications*, 2013.

Raghu Pasupathy and Shane G Henderson. Simopt: A library of simulation optimization problems. In *The Winter Simulation Conference*, 2011.

Michael Pearce, Matthias Poloczek, and Juergen Branke. Bayesian optimization allowing for common random numbers. In *The Winter Simulation Conference*, 2019.

Paris Perdikaris, Maziar Raissi, Andreas Damianou, Neil D Lawrence, and George Em Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2017.

Victor Picheny, David Ginsbourger, and Yann Richet. Noisy expected improvement and on-line computation time allocation for the optimization of simulators with tunable fidelity. In *The International Conference on Engineering Optimization* 2010.

Victor Picheny, David Ginsbourger, Yann Richet, and Gregory Caplin. Quantile-based optimization of noisy computer experiments with tunable precision. *Technometrics*, 2013.

Ghanshyam Pilania, James E Gubernatis, and Turab Lookman. Multi-fidelity machine learning models for accurate bandgap predictions of solids. *Computational Materials Science*, 2017.

Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: 2000-speaker neural text-to-speech. In *The International Conference on Learning Representations*, 2018.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In *Advances in Neural Information Processing Systems*, 2017.

Nishant Prateek, Mateusz Lajszczak, Roberto Barra-Chicote, Thomas Drugman, Jaime Lorenzo-Trueba, Thomas Merritt, Srikanth Ronanki, and Trevor Wood. In other news: A bi-style text-to-speech model for synthesizing newscaster voice with limited data. In *The North American Chapter of the Association for Computational Linguistics*, 2019.

Malte Prieß, Slawomir Koziel, and Thomas Slawig. Surrogate-based optimization of climate model parameters using response correction. *Journal of Computational Science*, 2011.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, 2008.

Laura Elena Raileanu and Kilian Stoffel. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 2004.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, 2004.

ITUR Recommendation. Method for the subjective assessment of intermediate sound quality (mushra). *ITU, BS*, 2001.

Binxin Ru, Michael A Osborne, Mark McLeod, and Diego Granziol. Fast information-theoretic Bayesian optimisation. In *The International Conference on Machine Learning*, 2018.

Binxin Ru, Ahsan S Alvi, Vu Nguyen, Michael A Osborne, and Stephen J Roberts. Bayesian optimisation over multiple continuous and categorical inputs. In *The International Conference on Machine Learning*, 2020.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*,1988.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *The International conference on artificial neural networks*, 1997.

Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, 2015.

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.

J Siebert. Vehicle recognition using rule based methods. *Turing Institute Report*, 1987.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *The International conference on machine learning*, 2015.

Jialin Song, Yury S Tokpanov, Yuxin Chen, Dagny Fleischman, Kate T Fountaine, Harry A Atwater, and Yisong Yue. Optimizing photonic nanostructures via multi-fidelity Gaussian processes. *arXiv preprint arXiv:1811.07707*, 2018.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *The International Confernce on Machine Learning*, 2009.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *The Annual Meeting of the Association for Computational Linguistics*, 2019.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2013.

Kevin Swersky, Yulia Rubanova, David Dohan, and Kevin Murphy. Amortized Bayesian optimization over discrete spaces. In *Uncertainty in Artificial Intelligence*, 2020.

Yaniv Taigman, Lior Wolf, Adam Polyak, and Eliya Nachmani. Voiceloop: Voice fitting and synthesis via a phonological loop. *arXiv:1707.06588*, 2017.

Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity Bayesian optimization with max-value entropy search. *arXiv preprint arXiv:1901.08275*, 2019.

Ryokei Tanaka and Hiroyoshi Iwata. Bayesian optimization for genomic selection: a method for discovering the best genotype among a large number of candidates. *Theoretical and Applied Genetics*, 2018.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The International Speech Communication Association: Speech Synthesis Workshop*, 2016.

Jarno Vanhatalo, Pasi Jylänki, and Aki Vehtari. Gaussian process regression with student-t likelihood. *Advances in Neural Information Processing Systems*, 2009.

Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. *Centre for Speech Technology Technical Report*, 2017.

Jean-Philippe Vert. Kernel methods in genomics and computational biology. *Kernel methods in Bioengineering, Signal and Image Processing*, 2007.

S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 2010.

Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang, and Sridhar Mahadevan. Efficient hyper-parameter optimization for nlp applications. In *The Conference on Empirical Methods in Natural Language Processing*, 2015.

Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *Interspeech*, 2017a.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *The International Conference on Machine Learning*, 2017.

Zi Wang, Bolei Zhou, and Stefanie Jegelka. Optimization as estimation with Gaussian processes in bandit settings. In *Artificial Intelligence and Statistics*, 2016.

Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional Bayesian optimization via structural kernel learning. In *The International Conference on Machine Learning*, 2017b.

Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 1994.

Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems*, 2016.

Jian Wu and Peter I Frazier. In *Neural Information Processing Systems: Bayesian Optimisation Workshop*, 2017.

Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. *arXiv preprint arXiv:1903.04703*, 2019.

Shifeng Xiong, Peter ZG Qian, and CF Jeff Wu. Sequential design and analysis of high-accuracy and low-accuracy computer codes. *Technometrics*, 2013.

Jie Xu, Si Zhang, Edward Huang, Chun-Hung Chen, Loo Hay Lee, and Nurcin Celik. Mo2tos: Multi-fidelity optimization with ordinal transformation and optimal sampling. *Asia-Pacific Journal of Operational Research*, 2016.

Junichi Yamagishi, Takao Kobayashi, Yuji Nakano, Katsumi Ogata, and Juri Isogai. Analysis of speaker adaptation algorithms for hmm-based speech synthesis and a constrained smaple adaptation algorithm. *IEEE Transactions on Audio, Speech, and Language Processing*, 2009.

Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *The International Conference on Machine Learning*, 1997.

Hau Kit Yong, Leran Wang, David JJ Toal, Andy J Keane, and Felix Stanley. Multi-fidelity kriging-assisted structural optimization of whole engine models employing medial meshes. *Structural and Multidisciplinary Optimization*, 2019.

Ping Yu, Yuan Yan, Qing Gu, and Xiangyang Wang. Codon optimisation improves the expression of trichoderma viride sp. endochitinase in pichia pastoris. *Scientific reports*, 2013.

Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv:1904.02882*, 2019.

Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs. In *Advances in Neural Information Processing Systems*, 2019.

Yehong Zhang, Trong Nghia Hoang, Bryan Kian Hsiang Low, and Mohan Kankanhalli. Information-based multi-fidelity Bayesian optimization. In *Advances in Neural Information Processing Systems: Workshop on Bayesian Optimization*, 2017.

Lingxiao Zheng, Tyson L Hedrick, and Rajat Mittal. A multi-fidelity modelling approach for evaluation and optimization of wing stroke aerodynamics in flapping flight. *Journal of Fluid Mechanics*, 2013.

Lavi R Zuhal, Cahya Amalinadhi, Yohanes B Dwianto, Pramudita S Palar, and Koji Shimoyama. Benchmarking multi-objective Bayesian global optimization strategies for aerodynamic design. In *Structures, Structural Dynamics, and Materials Conference*, 2018.