Taking the Human Out of the Loop: A Review of Bayesian Optimization

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams and Nando de Freitas

***Abstract*—Big data applications are typically associated with systems involving large numbers of users, massive complex software systems, and large-scale heterogeneous computing and storage architectures. The construction of such systems involves many distributed design choices. The end products (e.g., recommendation systems, medical analysis tools, real-time game engines, speech recognizers) thus involves many tunable configuration parameters. These parameters are often specified and hard-coded into the software by various developers or teams. If optimized jointly, these parameters can result in significant improvements. Bayesian optimization is a powerful tool for the joint optimization of design choices that is gaining great popularity in recent years. It promises greater automation so as to increase both product quality and human productivity. This review paper introduces Bayesian optimization, highlights some of its methodological aspects, and showcases a wide range of applications.**

I. Introduction

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

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

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

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

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

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

Bayesian optimization has emerged as a powerful solution for these varied design problems. In academia, it is impacting a wide range of areas, including interactive user-interfaces [26], robotics [101], [110], environmental monitoring [106], information extraction [158], combinatorial optimisation [79], [159], automatic machine learning [16], [143], [148], [151], [72], sensor networks [55], [146], adaptive Monte Carlo [105], experimental design [11] and reinforcement learning [27].

When software engineers develop programs, they are often faced with myriad choices. By making these choices explicit, Bayesian optimization can be used to construct optimal programs [74]: that is to say, programs that run faster or compute better solutions. Furthermore, since different components of software are typically integrated to build larger systems, this framework offers the opportunity to automate integrated products consisting of many parametrized software modules.

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

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{1}$$

where $\mathcal{X}$ is some design space of interest; in global optimization, $\mathcal{X}$ is often a compact subset of $\mathbb{R}^d$ but the Bayesian optimization framework can be applied to more unusual search spaces that involve categorical or conditional inputs, or even combinatorial search spaces with multiple

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

categorical inputs. Furthermore, we will assume the *black-box* function f has no simple closed form, but can be evaluated at any arbitrary query point $\mathbf{x}$ in the domain. This evaluation produces noise-corrupted (stochastic) outputs $y \in \mathbb{R}$ such that $\mathbb{E}[y \mid f(\mathbf{x})] = f(\mathbf{x})$. In other words, we can only observe the function f through unbiased noisy point-wise observations y. Although this is the minimum requirement for Bayesian optimization, when gradients are available, they can be incorporated in the algorithm as well; see for example Sections 4.2.1 and 5.2.4 of [99]. In this setting, we consider a sequential search algorithm which, at iteration n, selects a location $\mathbf{x}_{n+1}$ at which to query f and observe y_{n+1}. After N queries, the algorithm makes a final recommendation $\bar{\mathbf{x}}_N$, which represents the algorithm's best estimate of the optimizer.

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

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

In summary, the Bayesian optimization framework has two key ingredients. The first ingredient is a probabilistic surrogate model, which consists of a prior distribution that captures our beliefs about the behavior of the unknown objective function and an observation model that describes the data generation mechanism. The second ingredient is a loss function that describes how optimal a sequence of queries are; in practice, these loss functions often take the form of regret, either simple or cumulative. Ideally, the expected loss is then minimized to select an optimal sequence of queries. After observing the output of each query of the objective, the prior is updated to produce a more informative posterior distribution over the space of objective functions; see Figure 1 and Algorithm 1 for an illustration and pseudo-code of this framework. See Section 4 of [64] for another introduction.

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

Algorithm 1 Bayesian optimization

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

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

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

A. Paper overview

In this paper, we introduce the ingredients of Bayesian optimization in depth. Our presentation is unique in that we aim to disentangle the multiple components that determine the success of Bayesian optimization implementations. In particular, we focus on statistical modelling as this leads to general algorithms to solve a broad range tasks. We also provide an extensive comparison among popular acquisition functions. We will see that the careful choice of statistical model is often far more important than the choice of acquisition function heuristic.

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

B. Applications of Bayesian optimization

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

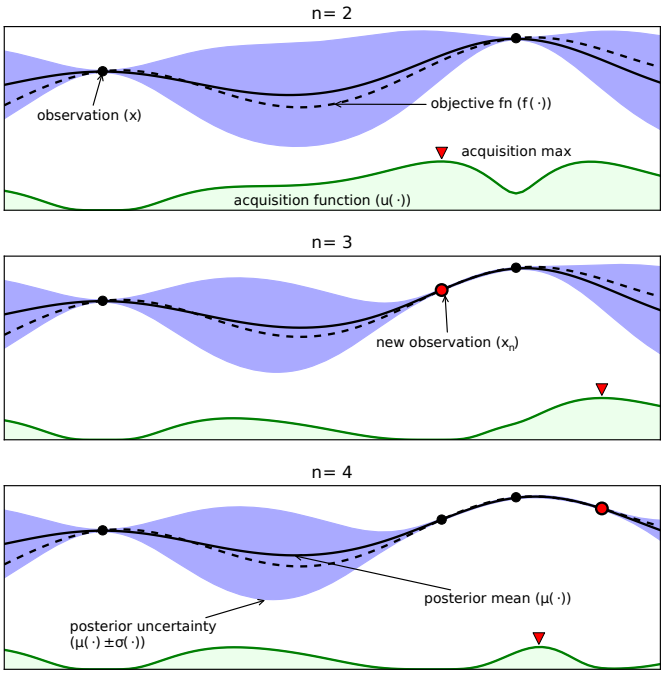

Fig. 1. Illustration of the Bayesian optimization procedure over three iterations. The plots show the mean and confidence intervals estimated with a probabilistic model of the objective function. Although the objective function is shown, in practice it is unknown. The plots also show the acquisition functions in the lower shaded plots. The acquisition is high where the model predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration). Note that the area on the far left remains unsampled, as while it has high uncertainty, it is correctly predicted to offer little improvement over the highest observation [27].

1) A/B testing: Though the idea of A/B testing dates back to the early days of advertising in the form of so-called focus groups, the advent of the internet and smartphones has given web and app developers a new forum for implementing these tests at unprecedented scales. By redirecting small fractions of user traffic to experimental designs of an ad, app, game, or website, the developers can utilize noisy feedback to optimize any observable metric with respect to the product's configuration. In fact, depending on the particular phase of a product's life, new subscriptions may be more valuable than revenue or user retention, or vice versa; the click-through rate might be the relevant objective to optimize for an ad, whereas for a game it may be some measure of user engagement.

The crucial problem is how to optimally query these subsets of users in order to find the best product with high probability within a predetermined query budget, or how to redirect traffic sequentially in order to optimize a cumulative metric while incurring the least opportunity cost [88], [135], [38].

2) Recommender systems: In a similar setting, online content providers make product recommendations to their subscribers in order to optimize either revenue in the case of e-commerce sites, readership for news sites, or consumption for video and music streaming websites. In contrast to A/B testing, the content provider can make multiple suggestions to any given subscriber. The techniques reviewed in this work have been successfully used for the recommendation of news articles [97], [38], [153].

3) Robotics and Reinforcement learning: Bayesian optimization has also been successfully applied to policy search. For example, by parameterizing a robot's gait it is possible to optimize it for velocity or smoothness as was done on the Sony AIBO ERS-7 in [101]. Similar policy parameterization and search techniques have been used to navigate a robot through landmarks, minimizing uncertainty about its own location and map estimate [110], [108]. See [27] for an example of applying Bayesian optimization to hierarchical reinforcement learning, where the technique is used to automatically tune the parameters of a neural network policy and to learn value functions at higher levels of the hierarchy. Bayesian optimization has also been applied to learn attention policies in image tracking with deep networks [44]

4) Environmental monitoring and sensor networks: Sensor networks are used to monitor environmentally relevant quantities: temperature, concentration of pollutants in the atmosphere, soil, oceans, etc. Whether inside a building or at a planetary scale, these networks make noisy local measurements that are interpolated to produce a global model of the quantity of interest. In some cases, these sensors are expensive to activate but one can answer important questions like what is the hottest or coldest spot in a building by activating a relatively small number of sensors. Bayesian optimization was used for this task and the similar one of finding the location of greatest highway traffic congestion [146]. Also, see [55] for a meteorological application.

When the sensor is mobile, there is a cost associated with making a measurement which relates to the distance travelled by a vehicle on which the sensor is mounted (e.g., a drone). This cost can be incorporated in the decision making process as in [106].

5) Preference learning and interactive interfaces: The computer graphics and animation fields are filled with applications that require the setting of tricky parameters. In many cases, the models are complex and the parameters unintuitive for non-experts. In [28], [26], the authors use Bayesian optimization to set the parameters of several animation systems by showing the user examples of different parametrized animations and asking for feedback. This *interactive Bayesian optimization* strategy is particulary effective as humans can be very good at comparing examples, but unable to produce an objective function whose optimum is the example of interest.

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

The traditional alternatives to cross-validation include racing algorithms that use conservative concentration bounds to rule out underperforming models [107], [113]. Recently, the

Bayesian optimization approach for the model selection and tuning task has received much attention in tuning deep belief networks [16], Markov chain Monte Carlo methods [105], [65], convolutional neural networks [143], [148], and automatically selecting among WEKA and `scikit-learn` offerings [151], [72].

7) Combinatorial optimization: Bayesian optimization has been used to solve difficult combinatorial optimization problems in several applications. One notable approach is called empirical hardness models (EHMs) that use a set of problem features to predict the performance of an algorithm on a specific problem instance [96]. Bayesian optimization with an EHM amounts to finding the best algorithm and configuration for a given problem. This concept has been applied to e.g., tuning mixed integer solvers [78], [159], and tuning approximate nearest neighbour algorithms [109]. Bayesian optimization has also been applied to fast object localization in images [163].

8) Natural language processing and text: Bayesian optimization has been applied to improve text extraction in [158] and to tune text representations for more general text and language tasks in [162].

II. Bayesian optimization with parametric models

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

Since $\mathbf{w}$ is an unobserved quantity, we treat it as a latent random variable with a *prior* distribution $p(\mathbf{w})$, which captures our *a priori* beliefs about probable values for $\mathbf{w}$ before any data is observed. Given data $\mathcal{D}$ and a *likelihood* model $p(\mathcal{D} \,|\, \mathbf{w})$, we can then infer a *posterior* distribution $p(\mathbf{w} \,|\, \mathcal{D})$ using Bayes' rule:

$$p(\mathbf{w} \,|\, \mathcal{D}) = \frac{p(\mathcal{D} \,|\, \mathbf{w})\, p(\mathbf{w})}{p(\mathcal{D})}\,. \tag{2}$$

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

A. Thompson sampling in the Beta-Bernoulli bandit model

We begin our discussion with a treatment of perhaps the simplest statistical model, the Beta-Bernoulli. Imagine that there are K drugs that have unknown effectiveness, where we define "effectiveness" as the probability of a successful cure. We wish to cure patients, but we must also identify which drugs are effective. Such a problem is often called a Bernoulli (or binomial) *bandit* problem by analogy to a group of slot machines, which each yield a prize with some unknown probability. In addition to clinical drug settings, this formalism is useful for A/B testing [135], advertising, and recommender systems [97], [38], among a wide variety of applications. The objective is to identify which *arm* of the bandit to pull, e.g., which drug to administer, which movie to recommend, or which advertisement to display. Initially, we consider the simple case where the arms are independent insofar as observing the success or failure of one provides no information about another.

Returning to the drug application, we can imagine the effectiveness of different drugs (arms on the bandit) as being determined by a function f that takes an index $a \in 1, \ldots, K$ and returns a Bernoulli parameter in the interval $(0, 1)$. With $y_i \in \{0, 1\}$, we denote the Bernoulli outcome of the treatment of patient i, and this has mean parameter $f(a_i)$ if the drug administered was a_i. Note that we are assuming stochastic feedback, in contrast to deterministic or adversarial feedback [9], [10]. With only K arms, we can fully describe the function f with a parameter $\mathbf{w} \in (0, 1)^K$ so that $f_{\mathbf{w}}(a) := w_a$.

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

$$p(\mathbf{w} \,|\, \alpha, \beta) = \prod_{a=1}^{K} \mathrm{Beta}(w_a \,|\, \alpha, \beta)\,, \tag{3}$$

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

$$n_{a,0} = \sum_{i=1}^{n} \mathbb{I}(y_i = 0, a_i = a) \tag{4}$$

$$n_{a,1} = \sum_{i=1}^{n} \mathbb{I}(y_i = 1, a_i = a)\,. \tag{5}$$

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

$$p(\mathbf{w} \,|\, \mathcal{D}) = \prod_{a=1}^{K} \mathrm{Beta}(w_a \,|\, \alpha + n_{a,1}, \beta + n_{a,0})\,. \tag{6}$$

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

In Section IV, we will introduce various strategies for selecting the next arm to pull within models like the Beta-Bernoulli, but for the sake of illustration, we introduce Thompson sampling [150], the earliest and perhaps the simplest non-trivial bandit strategy. This strategy is also commonly known as randomized probability matching [135] because it selects the arm based on the posterior probability of optimality, here

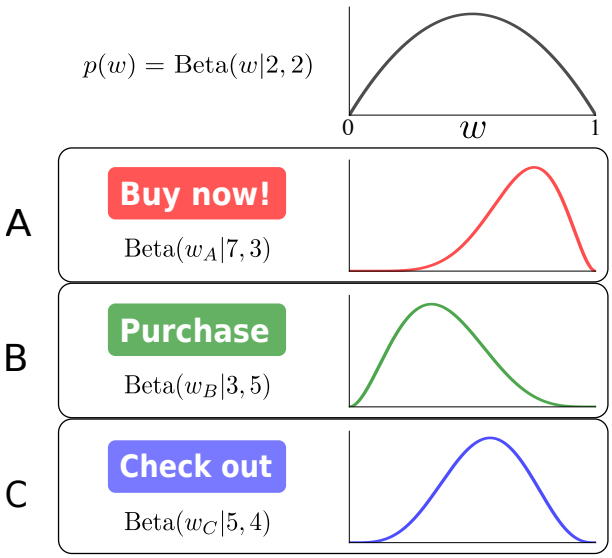

Fig. 2. Example of the Beta-Bernoulli model for A/B testing. Three different buttons are being tested with various colours and text. Each option is given 2 successes (click-throughs) and 2 failures as a prior (top). As data are observed, each option updates its posterior over w. Option A is the current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

given by a beta distribution. In simple models like the Beta-Bernoulli, it is possible to compute this distribution in closed form, but more often it must be estimated via, e.g., Monte Carlo.

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

$$a_{n+1} = \arg\max_a f_{\tilde{\mathbf{w}}}(a) \;\text{ where }\; \tilde{\mathbf{w}} \sim p(\mathbf{w} \mid \mathcal{D}_n). \tag{7}$$

For the Beta-Bernoulli, this corresponds to simply drawing $\tilde{\mathbf{w}}$ from (6) and then choosing the action with the largest $\tilde{w}_a$. This procedure, shown in pseudo-code in Algorithm 2, is also commonly called *posterior sampling* [127]. It is popular for several reasons: 1) there are no free parameters other than the prior hyperparameters of the Bayesian model, 2) the strategy naturally trades off between exploration and exploitation based on its posterior beliefs on $\mathbf{w}$; arms are explored only if they are likely (under the posterior) to be optimal, 3) the strategy is relatively easy to implement as long as Monte Carlo sampling mechanisms are available for the posterior model, and 4) the randomization in Thompson sampling makes it particularly appropriate for batch or delayed feedback settings where many selections a_{n+1} are based on the identical posterior [135], [38].

Algorithm 2 Thompson sampling for Beta-Bernoulli bandit

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

B. Linear models

In many applications, the designs available to the experimenter have components that can be varied independently. For example, in designing an advertisement, one has choices such as artwork, font style, and size; if there are five choices for each, the total number of possible configurations is 125. In general, this number grows combinatorially in the number of components. This presents challenges for approaches such as the independent Beta-Bernoulli model discussed in the previous section: modelling the arms as independent will lead to strategies that must try every option at least once. This rapidly becomes infeasible in the large spaces of real-world problems. In this section, we discuss a parametric approach that captures dependence between the arms via a linear model. For simplicity, we first consider the case of real-valued outputs y and generalize this model to binary outputs in the succeeding section.

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

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

$$\begin{aligned}\text{NIG}(\mathbf{w}, \sigma^2 \mid \mathbf{w}_0, \mathbf{V}_0, \alpha_0, \beta_0) = \\ |2\pi\sigma^2 \mathbf{V}_0|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{w}-\mathbf{w}_0)^T \mathbf{V}_0^{-1}(\mathbf{w}-\mathbf{w}_0)\right\} \\ \times \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)(\sigma^2)^{\alpha_0+1}} \exp\left\{-\frac{\beta_0}{\sigma^2}\right\}.\end{aligned} \tag{8}$$

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

with parameters

$$\mathbf{w}_n = \mathbf{V}_n(\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{X}^T\mathbf{y}) \tag{9}$$
$$\mathbf{V}_n = (\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1} \tag{10}$$
$$\alpha_n = \alpha_0 + n/2 \tag{11}$$
$$\beta_n = \beta_0 + \frac{1}{2}\left(\mathbf{w}_0^T\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{y}^T\mathbf{y} - \mathbf{w}_n^T\mathbf{V}_n^{-1}\mathbf{w}_n\right). \tag{12}$$

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

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

$$a_{n+1} = \arg\max_a \mathbf{x}_a^T\tilde{\mathbf{w}} \text{ where } \tilde{\mathbf{w}} \sim p(\mathbf{w} \mid \mathcal{D}_n). \tag{13}$$

After arm a_{n+1} is pulled and y_{n+1} is observed, the posterior model can be readily updated using equations (9–12).

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

$$\text{maximize} \quad f(\mathbf{x}) \qquad \text{s.t.} \quad \mathbf{x} \in \mathcal{X}. \tag{14}$$

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

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

$$f(\mathbf{x}) = \Phi(\mathbf{x})^T\mathbf{w}, \tag{15}$$

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

$$\phi_j(\mathbf{x}) = \exp\left\{-\tfrac{1}{2}(\mathbf{x} - \mathbf{z}_j)^T\mathbf{\Lambda}(\mathbf{x} - \mathbf{z}_j)\right\}, \tag{16}$$

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

$$\phi_j(\mathbf{x}) = \exp\left\{-i\mathbf{x}^T\boldsymbol{\omega}_j\right\}, \tag{17}$$

with hyperparameters $\{\boldsymbol{\omega}_j\}_{j=1}^J$.

Recently, such basis functions have also been learned from data by training deep belief networks [71], deep neural networks [93], [144], or by factoring the empirical covariance matrix of historical data [146], [72]. For example, in [34] each sigmoidal layer of an L layer neural network is defined as $\mathcal{L}_\ell(\mathbf{x}) := \sigma(\mathbf{W}_\ell\mathbf{x} + \mathbf{B}_\ell)$ where σ is some sigmoidal non-linearity, and $\mathbf{W}_\ell$ and $\mathbf{B}_\ell$ are the layer parameters. Then the feature map $\Phi : \mathbb{R}^d \mapsto \mathbb{R}^J$ can be expressed as $\Phi(\mathbf{x}) = \mathcal{L}_L \circ \cdots \circ \mathcal{L}_1(\mathbf{x})$, where the final layer $\mathcal{L}_L$ has J output units. In [144], the weights of the last layer of a deep neural network are integrated out to result in a tractable Bayesian model with flexible learned basis functions.

Regardless of the feature map Φ, when conditioned on these basis functions, the posterior over the weights $\mathbf{w}$ can be computed analytically using (9-12). Let $\Phi(\mathbf{X})$ denote the $n \times J$ matrix where $[\Phi(\mathbf{X})]_{i,j} = \phi_j(\mathbf{x}_i)$; then the posterior is as in Bayesian linear regression, substituting $\Phi(\mathbf{X})$ for the design matrix $\mathbf{X}$.

C. Generalized linear models

While simple linear models capture the dependence between bandit arms in a straightforward and expressive way, the model as described does not immediately apply to other types of observations, such as binary or count data. Generalized linear models (GLMs) [119] allow more flexibility in the response variable through the introduction of a link function. Here we examine the GLM for binary data such as might arise from drug trials or AB testing.

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

Unfortunately, there is no conjugate prior for the parameters $\mathbf{w}$ when such a likelihood is used and so we must resort to approximate inference. Markov chain Monte Carlo (MCMC) methods [4] approximate the posterior with a sequence of samples that converge to the posterior; this is the approach taken in [135] on the probit model. In contrast, the Laplace approximation fits a Gaussian distribution to the posterior by matching the curvature of the posterior distribution at the mode. For example in [38], Bayesian logistic regression with a Laplace approximation was used to model click-throughs for the recommendation of news articles in a live experiment. In the generalized linear model, Thompson sampling draws a $\tilde{\mathbf{w}}$ from the posterior $p(\mathbf{w} \mid \mathcal{D}_n)$ using MCMC or a Laplace approximation, and then selects the arm with the highest expected reward given the sampled parameter $\tilde{\mathbf{w}}$, i.e., $a_{n+1} = \arg\max_a g^{-1}(\mathbf{x}_a^T\tilde{\mathbf{w}})$.

D. Related literature

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

$$\alpha(\mathbf{X}) := \mathbb{E}_{\mathbf{w}}\mathbb{E}_{\mathbf{y} \mid \mathbf{X},\mathbf{w}}\left[U(\mathbf{X}, \mathbf{y}, \mathbf{w})\right], \tag{18}$$

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

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

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

In the Bayesian experimental design literature, this criterion is known as the D-optimality utility and was first introduced by Lindley [98]. Since this seminal work, many alternative utilities have been proposed in the experimental design literature. See [37] for a detailed survey.

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

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

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

$$\alpha(\mathbf{X}) = \mathbb{E}_{\mathbf{w}}\mathbb{E}_{\mathbf{y} \mid \mathbf{X},\mathbf{w}}\Big[\sum_{i=1}^{n} \gamma^{i-1} y_i\Big]. \tag{19}$$

When $\gamma < 1$, a Bayes-optimal sequence $\mathbf{X}$ can be computed for the Bernoulli bandit via dynamic programming, due to Gittins [59]. However, this solution is intractable for general reward distributions, and so in practice sequential heuristics are used and analyzed in terms of a frequentist measure, namely *cumulative regret* [92], [135], [146], [38], [127].

Cumulative regret is a frequentist measure defined as

$$R_n(\mathbf{w}) = \sum_{i=1}^{n} f^\star_{\mathbf{w}} - f_{\mathbf{w}}(\mathbf{x}_{a_i}), \tag{20}$$

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

Recently, there has been growing interest in the *best arm identification* problem, which is more suitable for the model selection task [104], [30], [7], [51], [50], [72]. When using Bayesian surrogate models, this is equivalent to performing Bayesian optimization on a finite, discrete domain. In this so-called pure exploration settings, in addition to a selection strategy, a *recommendation strategy* ρ is specified to recommend an arm (or ad or drug) at the end of the experimentation based on observed data. The experiment is then judged via the *simple regret*, which depends on the recommendation $\bar{a} = \rho(\mathcal{D})$:

$$r_n(\mathbf{w}) = f^\star_{\mathbf{w}} - f_{\mathbf{w}}(\mathbf{x}_{\bar{a}})\,. \tag{21}$$

III. Non-parametric models

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

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{X}, \sigma^2) &= \int p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}, \sigma^2)\, p(\mathbf{w} \mid 0, \mathbf{V}_0)\, \mathrm{d}\mathbf{w} \\ &= \int \mathcal{N}(\mathbf{y} \mid \mathbf{X}\mathbf{w}, \sigma^2\mathbf{I})\, \mathcal{N}(\mathbf{w} \mid 0, \mathbf{V}_0)\, \mathrm{d}\mathbf{w} \\ &= \mathcal{N}(\mathbf{y} \mid 0, \mathbf{X}\mathbf{V}_0\mathbf{X}^T + \sigma^2\mathbf{I})\,. \end{aligned} \tag{22}$$

As noted earlier, it can be useful to introduce basis functions ϕ and in the context of Bayesian linear regression we in effect replace the design matrix $\mathbf{X}$ with a feature mapping matrix $\mathbf{\Phi} = \Phi(\mathbf{X})$. In Equation (22), this results in a slightly different Gaussian for weights in feature space:

$$p(\mathbf{y} \mid \mathbf{X}, \sigma^2) = \mathcal{N}(\mathbf{y} \mid 0, \mathbf{\Phi}\mathbf{V}_0\mathbf{\Phi}^T + \sigma^2\mathbf{I}) \tag{23}$$

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

$$\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i)\mathbf{V}_0\mathbf{\Phi}(\mathbf{x}_j)^T \tag{24}$$

$$= \langle \mathbf{\Phi}(\mathbf{x}_i), \mathbf{\Phi}(\mathbf{x}_j) \rangle_{\mathbf{V}_0}\,. \tag{25}$$

The kernel trick allows us to specify an intuitive similarity between pairs of points, rather than a feature map $\mathbf{\Phi}$, which in practice can be hard to define. In other words, we can either think of predictions as depending directly on features $\mathbf{\Phi}$, as in the linear regression problem, or on kernels k, as in the lifted variant, depending on which paradigm is more interpretable or computationally tractable. Indeed, the former requires a $J \times J$ matrix inversion compared to the latter's $n \times n$.

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

$$p(\mathbf{y}_\star \,|\, \mathbf{X}_\star, \mathbf{X}, \mathbf{y}, \sigma^2) = \frac{p(\mathbf{y}_\star, \mathbf{y} \,|\, \mathbf{X}_\star, \mathbf{X}, \sigma^2)}{p(\mathbf{y} \,|\, \mathbf{X}, \sigma^2)}\,. \tag{26}$$

Both the numerator and the denominator are Gaussian with the form appearing in Equation (23), and so the predictions are jointly Gaussian and can be computed via some simple linear algebra. Critically, given a kernel k, it becomes unnecessary to explicitly define or compute the features $\mathbf{\Phi}$ because both the predictions and the marginal likelihood only depend on $\mathbf{K}$.

A. The Gaussian process

By kernelizing a marginalized version of Bayesian linear regression, what we have really done is construct an object called a Gaussian process. The Gaussian process $\mathrm{GP}(\mu_0, k)$ is a non-parametric model that is fully characterized by its prior mean function $\mu_0 : \mathcal{X} \mapsto \mathbb{R}$ and its positive-definite kernel, or covariance function, $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ [126]. Consider any finite collection[2] of n points $\mathbf{x}_{1:n}$, and define variables $f_i := f(\mathbf{x}_i)$ and $y_{1:n}$ to represent the unknown function values and noisy observations, respectively. In Gaussian process regression, we assume that $\mathbf{f} := f_{1:n}$ are jointly Gaussian and the observations $\mathbf{y} := y_{1:n}$ are normally distributed given $\mathbf{f}$, resulting in the following generative model:

$$\mathbf{f} \,|\, \mathbf{X} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}) \tag{27}$$

$$\mathbf{y} \,|\, \mathbf{f}, \sigma^2 \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbf{I})\,, \tag{28}$$

where the elements of the mean vector and covariance matrix are defined as $m_i := \mu_0(\mathbf{x}_i)$ and $K_{i,j} := k(\mathbf{x}_i, \mathbf{x}_j)$, respectively. Equation (27) represents the prior distribution $p(\mathbf{f})$ induced by the GP.

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

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} (\mathbf{y} - \mathbf{m}) \tag{29}$$

$$\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}(\mathbf{x})\,, \tag{30}$$

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

The posterior mean and variance evaluated at any point $\mathbf{x}$ represent the model's prediction and uncertainty, respectively, in the objective function at the point $\mathbf{x}$. These posterior functions are used to select the next query point $\mathbf{x}_{n+1}$ as detailed in Section IV.

B. Common kernels

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

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

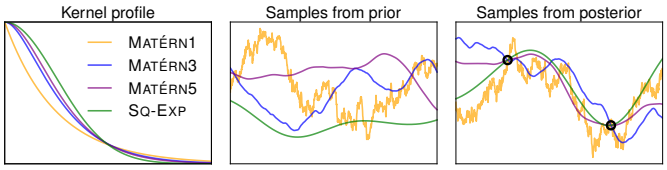

Fig. 3. **Left:** Visualization of various kernel profiles. The horizontal axis represents the distance $r > 0$. **Middle:** Samples from GP priors with the corresponding kernels. **Right:** Samples from GP posteriors given two data points (black circles). Note the sharper drop in the Matérn1 kernel leads to rough features in the associated samples, while samples from a GP with the Matérn3 and Matérn5 kernels are increasingly smooth.

Matérn kernels are a very flexible class of stationary kernels. These kernels are parameterized by a smoothness parameter $\nu > 0$, so called because samples from a GP with such a kernel are differentiable $\lfloor \nu - 1 \rfloor$ times [126]. The exponential kernel is a special case of the Matérn kernel with $\nu = \frac{1}{2}$, and the squared exponential kernel is the limiting kernel when $\nu \to \infty$. The following are the most commonly used kernels, labelled by the smoothness parameter, omitting the factor of $\frac{1}{2}$.

$$k_{\text{MATÉRN1}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-r) \tag{31}$$

$$k_{\text{MATÉRN3}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{3}r)(1 + \sqrt{3}r) \tag{32}$$

$$k_{\text{MATÉRN5}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{5}r)(1 + \sqrt{5}r + \tfrac{5}{3}r^2) \tag{33}$$

$$k_{\text{SQ-EXP}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\tfrac{1}{2}r^2), \tag{34}$$

where $r^2 = (\mathbf{x} - \mathbf{x}')^T \mathbf{\Lambda} (\mathbf{x} - \mathbf{x}')$ and $\mathbf{\Lambda}$ is a diagonal matrix of d squared length scales θ_i^2. This family of covariance functions are therefore parameterized by an amplitude and d length scale hyperparameters, jointly denoted θ. Covariance functions with learnable length scale parameters are also known as automatic relevance determination (ARD) kernels. Figure 3 provides a visualization of the kernel profiles and samples from the corresponding priors and posteriors.

C. Prior mean functions

While the kernel function controls the smoothness and amplitude of samples from the GP, the prior mean provides a possible offset. In practice, this function is set to a constant $\mu_0(\mathbf{x}) \equiv \mu_0$ and inferred from data using techniques covered in Section V-A. Unless otherwise specified, in what follows we assume a constant prior mean function for convenience. However, the prior mean function is a principled way of incorporating expert knowledge of the objective function, if it is available, and the following analysis can be readily applied to non-constant functions μ_0.

D. Marginal likelihood

Another attractive property of the Gaussian process model is that it provides an analytical expression for the marginal likelihood of the data, where marginal refers to the fact that the

unknown latent function f is marginalized out. The expression for the log marginal likelihood is simply given by:

$$\log p(\mathbf{y}|\mathbf{x}_{1:n},\theta) = -\frac{1}{2}(\mathbf{y}-\mathbf{m}_\theta)^T(\mathbf{K}^\theta+\sigma^2\mathbf{I})^{-1}(\mathbf{y}-\mathbf{m}_\theta) - \frac{1}{2}\log|\mathbf{K}^\theta+\sigma^2\mathbf{I}| - \frac{n}{2}\log(2\pi), \quad (35)$$

where in a slight abuse of notation we augment the vector $\theta := (\theta_{0:d}, \mu_0, \sigma^2)$; and the dependence on θ is made explicit by adding a superscript to the covariance matrix $\mathbf{K}^\theta$. The marginal likelihood is very useful in learning the hyperparameters, as we will see in Section V-A. The right hand side of (35) can be broken into three terms: the first term quantifies how well the model fits the data, which is simply a Mahalanobis distance between the model predictions and the data; the second term quantifies the model complexity – smoother covariance matrices will have smaller determinants and therefore lower complexity penalties; finally, the last term is simply a linear function of the number of data points n, indicating that the likelihood of data tends to decrease with larger datasets.

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

E. Computational costs and other regression models

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

1) Sparse pseudo-input Gaussian processes (SPGP): One early approach to modelling large n with Gaussian processes considered using $m < n$ *inducing pseudo-inputs* to reduce the rank of the covariance matrix to m, resulting in a significant reduction in computational cost [137], [140]. By forcing the interaction between the n data points $\mathbf{x}_{1:n}$ and any test point $\mathbf{x}$ to go through this set of m inducing pseudo-inputs, these methods can compute an approximate posterior in $O(nm^2+m^3)$ time. Pseudo-input methods have since been unified in a single theory based on the following overarching approximation.

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

$$p(\mathbf{f}_\star,\mathbf{f}) = \int p(\mathbf{f}_\star,\mathbf{f},\mathbf{u})\,\mathrm{d}\mathbf{u} \quad (36)$$

$$\approx \int q(\mathbf{f}_\star\,|\,\mathbf{u})\,q(\mathbf{f}\,|\,\mathbf{u})\,p(\mathbf{u})\,\mathrm{d}\mathbf{u} = q(\mathbf{f},\mathbf{f}_\star) \quad (37)$$

where $\mathbf{u}$ is the vector of function values at the pseudo-inputs. All sparse pseudo-input GP approximations can be specified in terms of the form used for the training and test conditionals, $q(\mathbf{f}\,|\,\mathbf{u})$ and $q(\mathbf{f}_\star\,|\,\mathbf{u})$, respectively [124].

In the seminal works on pseudo-input methods, the locations of the pseudo-inputs were selected to optimize the marginal likelihood of the SPGP [137], [140]. In contrast, a variational approach has since been proposed to marginalize the pseudo-inputs to maximize fidelity to the original exact GP [152] rather than the likelihood of the approximate GP.

The computational savings in the pseudo-input approach to approximating the GP comes at the cost of poor variance estimates. As can be observed in Figure 4, the uncertainty (blue shaded area) exhibits unwanted pinching at pseudo-inputs, while it is overly conservative in between and away from pseudo-inputs. In this instance, the 10 inducing points, indicated with black crosses, were not optimized to emphasize the potential pathologies of the method. Since in Bayesian optimization we use the credible intervals to guide exploration, these artefacts can mislead our search.

2) Sparse spectrum Gaussian processes (SSGP): While inducing pseudo-inputs reduce computational complexity by using a fixed number of points in the search space, sparse spectrum Gaussian processes (SSGP) take a similar approach to the kernel's spectral space [94]. Bochner's theorem states that any stationary kernel $k(\mathbf{x},\mathbf{x}') = k(\mathbf{x}-\mathbf{x}')$ has a positive and finite Fourier spectrum $s(\boldsymbol{\omega})$, i.e.,

$$k(\mathbf{x}) = \frac{1}{(2\pi)^d}\int e^{-i\boldsymbol{\omega}^T\mathbf{x}}s(\boldsymbol{\omega})\,\mathrm{d}\boldsymbol{\omega}\,. \quad (38)$$

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

$$k(\mathbf{x},\mathbf{x}') = \nu\,\mathbb{E}_{\boldsymbol{\omega}}[e^{-i\boldsymbol{\omega}^T(\mathbf{x}-\mathbf{x}')}]. \quad (39)$$

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

$$k(\mathbf{x},\mathbf{x}') \approx \frac{\nu}{m}\sum_{i=1}^{m} e^{-i\boldsymbol{\omega}^{(i)T}\mathbf{x}}e^{i\boldsymbol{\omega}^{(i)T}\mathbf{x}'} \quad (40)$$

where $\boldsymbol{\omega}^{(i)} \sim s(\boldsymbol{\omega})/\nu$. The resulting finite dimensional problem is equivalent to Bayesian linear regression with m basis functions and the computational cost is once again reduced to $O(nm^2+m^3)$.

As with the pseudo-inputs, the spectral points can also be tuned via marginal likelihood optimization. Although this violates the Monte Carlo assumption and introduces a risk of

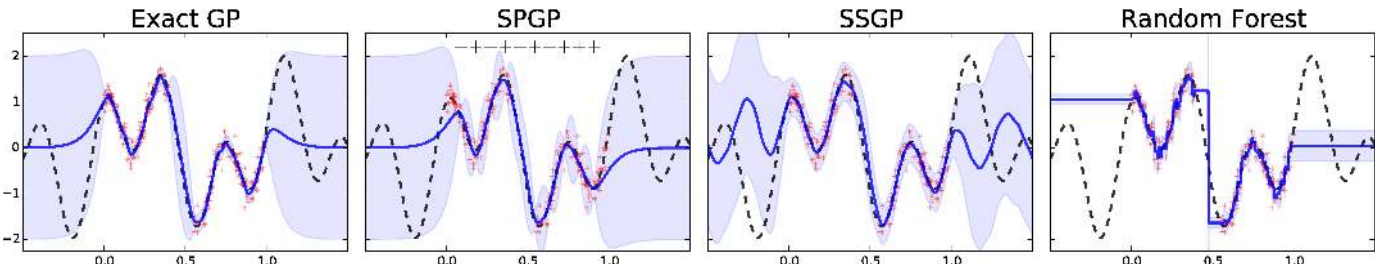

Fig. 4. Comparison of surrogate regression models. Four different surrogate model posteriors are shown in blue (shaded area delimits 95% credible intervals), given noisy evaluations (red crosses) of a synthetic function (dashed line). The 10 pseudo-inputs for the SPGP method are shown as black crosses. The SSGP model used a basis of 80 Fourier features.

overfitting, it allows for a smaller number of basis functions with good predictive power [94]. Once again, in Figure 4 we have not tuned the 80 spectral points in this way. Whereas around observed data (red crosses) the uncertainty estimates are smoother than the pseudo-inputs method, away from observations both the prediction and uncertainty regions exhibit spurious oscillations. This is highly undesirable for Bayesian optimization where we expect our surrogate model to fall back on the prior away from observed data.

3) Random forests: Finally, as an alternative to Gaussian processes, random forest regression has been proposed as an expressive and flexible surrogate model in the context of sequential model-based algorithm configuration (SMAC) [79]. Introduced in 2001 [24], random forests are a class of scalable and highly parallelizable regression models that have been very successful in practice [42]. More precisely, the random forest is an ensemble method where the weak learners are decision trees trained on random subsamples of the data [24]. Averaging the predictions of the individual trees produces an accurate response surface.

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

The exploration strategy in SMAC still requires an uncertainty estimate for predictions at test points. While the random forest does not provide an estimate of the variance of its predictions, Hutter et al. proposed using the empirical variance in the predictions across trees in the ensemble [79]. Though these are not principled uncertainty estimates, this heuristic has been shown to work well in practice for the SMAC algorithm.

Although random forests are good interpolators in the sense that they output good predictions in the neighbourhood of training data, they are very poor extrapolators. Indeed, far from the data, the predictions of all trees could be identical, resulting in a poor prediction; more importantly, using the variance estimate of SMAC results in extremely confident intervals. In Figure 4 for example, away from data the shaded area is very narrow around a very poor constant prediction. Even more troubling is the fact that in areas of missing data multiple conflicting predictions can cause the empirical variance to blow up sharply, as can be seen in Figure 4. While Gaussian processes are also poor extrapolators (when used with local kernels), they produce relatively uncertain predictions away from the data by reverting to the prior – a more desirable behavior when trading off exploration and exploitation.

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

IV. Acquisition functions

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

Consider the utility function $U : \mathbb{R}^d \times \mathbb{R} \times \Theta \mapsto \mathbb{R}$ which maps an arbitrary query point $\mathbf{x}$, its corresponding function value $v = f(\mathbf{x})$, and a setting of the model hyperparameters θ to a measure of quality of the experiment, e.g., how much information this query will provide as in [98]. Given some data accumulated thus far, we can marginalize the unseen outcome y and the unknown model hyperparameters θ to obtain the expected utility of a query point $\mathbf{x}$:

$$\alpha(\mathbf{x}; \mathcal{D}_n) = \mathbb{E}_\theta \mathbb{E}_{v \mid \mathbf{x}, \theta}[U(\mathbf{x}, v, \theta)] \tag{41}$$

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

Whereas in experimental design and decision theory, the function α is called the expected utility, in Bayesian optimization it is often called the acquisition or infill function. These acquisition functions are carefully designed to trade off

exploration of the search space and exploitation of current promising areas. We first present traditional improvement-based and optimistic acquisition functions, followed by more recent information-based approaches.

A. Improvement-based policies

Improvement-based acquisition functions favour points that are likely to improve upon an incumbent target τ. An early strategy in the literature, probability of improvement (PI) [91], measures the probability that a point $\mathbf{x}$ leads to an improvement upon τ. Since the posterior distribution of $v = f(\mathbf{x})$ is Gaussian, we can analytically compute this probability as follows:

$$\alpha_{\text{PI}}(\mathbf{x}; \mathcal{D}_n) := \mathbb{P}[v > \tau] = \Phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right), \tag{42}$$

where Φ is the standard normal cumulative distribution function. Recall that α_{PI} is then maximized to select the next query point. For this criterion, the utility function is simply an indicator of improvement $U(\mathbf{x}, v, \theta) = \mathbb{I}[v > \tau]$, where the utility function is expressed (and marginalized) with respect to the latent variable v. Therefore, all improvements are treated equal and PI simply accumulates the posterior probability mass above τ at $\mathbf{x}$.

Although probability of improvement can perform very well when the target is known, in general the heuristic used for an unknown target τ causes PI to exploit quite aggressively [81].

One could instead measure the expected improvement (EI) [115] which incorporates the *amount* of improvement. This new criterion corresponds to a different utility that is called the improvement function, denoted by $I(\mathbf{x})$. Formally, the improvement function I is defined as follows

$$I(\mathbf{x}, v, \theta) := (v - \tau)\,\mathbb{I}(v > \tau). \tag{43}$$

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

$$\begin{aligned}\alpha_{\text{EI}}(\mathbf{x}; \mathcal{D}_n) &:= \mathbb{E}\left[I(\mathbf{x}, v, \theta)\right] \\ &= (\mu_n(\mathbf{x}) - \tau)\Phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right) \\ &\quad + \sigma_n(\mathbf{x})\phi\left(\frac{\mu_n(\mathbf{x}) - \tau}{\sigma_n(\mathbf{x})}\right),\end{aligned} \tag{44}$$

when $\sigma_n > 0$ and vanishes otherwise. Here, not to be confused with the previous section, ϕ is the standard normal probability density function. These improvement strategies have been empirically studied in the literature [82], [81], [27] and recently convergence rates have been proven for EI [32].

Finally, although the target objective value (i.e., the best reachable objective value) is often unknown, in practice τ is adaptively set to the best observed value $y^+ = \max_{i=1:n} y_i$. Whereas for PI this heuristic can lead to an overly greedy optimization [81], it works reasonably with EI in practice [143]. When the objective function being minimized is very noisy, using the lowest mean value as the target is reasonable [157].

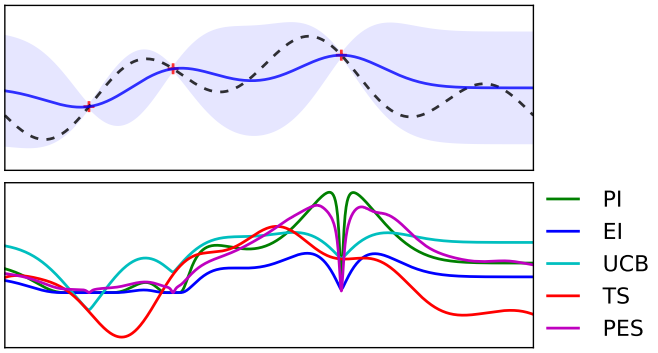

Fig. 5. Visualization of the surrogate regression model and various acquisition functions. **(Top)** The true objective function is shown as a dashed line and the probabilistic regression model is shown as a blue line with a shaded region delimiting the $2\sigma_n$ credible intervals. Finally, the observations are shown as red crosses. **(Bottom)** Four acquisition functions are shown. In the case of PI, the optimal mode is much closer to the best observation as in the alternative methods, which explains its greedy behaviour. In contrast, the randomization in TS allows it to explore more aggressively.

B. Optimistic policies

Dating back to the seminal work of Lai & Robbins [92] on the multi-armed bandit problem, the upper confidence bound criterion has been a popular way of negotiating exploration and exploitation, often with provable cumulative regret bounds. The guiding principle behind this class of strategies is to be optimistic in the face of uncertainty. Indeed, using the upper confidence for every query point $\mathbf{x}$ corresponds to effectively using a fixed probability best case scenario according to the model. Originally, the upper confidence was given by frequentist Chernoff–Hoeffding bounds [8].

More recently, the Gaussian process upper confidence bound (GP-UCB [146]) algorithm was proposed as a Bayesian optimistic algorithm with provable cumulative regret bounds. In the deterministic case, a branch-and-bound extension to GP-UCB was proven to have exponentially vanishing instantaneous regret [43]. The GP-UCB algorithm has since been generalized to other Bayesian models by considering upper quantiles [84] instead of Equation (45) defined below, which is more reminiscent of frequentist concentration bounds. In the GP case, since the posterior at any arbitrary point $\mathbf{x}$ is a Gaussian, any quantile of the distribution of $f(\mathbf{x})$ is computed with its corresponding value of β_n as follows:

$$\alpha_{\text{UCB}}(\mathbf{x}; \mathcal{D}_n) := \mu_n(\mathbf{x}) + \beta_n\sigma_n(\mathbf{x}). \tag{45}$$

There are theoretically motivated guidelines for setting and scheduling the hyperparameter β_n to achieve optimal regret [146] and, as with τ in the improvement policies, tuning this parameter within these guidelines can offer a performance boost.

Finally, there also exist variants of these algorithms for the contextual bandits [153] (see Section VIII-D) and parallel querying [45] (see Section V-E).

C. Information-based policies

In contrast to the acquisition functions introduced so far, information-based policies consider the posterior distribution

over the unknown minimizer $\mathbf{x}^\star$, denoted $p_\star(\mathbf{x} \mid \mathcal{D}_n)$. This distribution is implicitly induced by the posterior over objective functions f. There are two policies in this class, namely Thompson sampling and entropy search.

Though it was introduced in 1933 [150], Thompson sampling has attracted renewed interest in the multi-armed bandit community, producing empirical evaluations [135], [38] as well as theoretical results [85], [2], [127]. Thompson sampling (TS) is a randomized strategy which samples a reward function from the posterior and selects the arm with the highest simulated reward. Therefore the selection made by TS can be expressed as the randomized acquisition function $\mathbf{x}_{n+1} \sim p_\star(\mathbf{x} \mid \mathcal{D}_n)$.

However, in continuous search spaces, the analog of Thompson sampling is to draw a continuous function $f^{(n)}$ from the posterior GP and optimize it to obtain $\mathbf{x}_{n+1}$. In order to be optimized, the sample $f^{(n)}$ needs to be fixed so it can be queried at arbitrary points; unfortunately, it is not clear how to fix an exact sample from the GP. However, using recent spectral sampling techniques [20], [125], [94], we can draw an approximate sample from the posterior that can be evaluated at any arbitrary point $\mathbf{x}$ [69], which extends TS to continuous search spaces. As an acquisition function, TS can be formulated as

$$\alpha_{\mathrm{TS}}(\mathbf{x}; \mathcal{D}_n) := f^{(n)}(\mathbf{x}) \quad \text{where } f^{(n)} \overset{s.s.}{\sim} \mathrm{GP}(\mu_0, k \mid \mathcal{D}_n) \tag{46}$$

where $\overset{s.s.}{\sim}$ indicates approximate simulation via spectral sampling. Empirical evaluations show good performance which, however, seems to deteriorate in high dimensional problems, likely due to aggressive exploration [139].

Instead of sampling the distribution $p_\star(\mathbf{x} \mid \mathcal{D}_n)$, entropy search (ES) techniques aim to reduce the uncertainty in the location $\mathbf{x}^\star$ by selecting the point that is expected to cause the largest reduction in entropy of the distribution $p_\star(\mathbf{x} \mid \mathcal{D}_n)$ [156], [67], [69]. In terms of utility, entropy search methods use the information gain defined as follows

$$U(\mathbf{x}, y, \theta) = H(\mathbf{x}^\star \mid \mathcal{D}_n) - H(\mathbf{x}^\star \mid \mathcal{D}_n \cup \{(\mathbf{x}, y)\}), \tag{47}$$

where the θ implicitly parameterizes the distribution of y.

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

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

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

Once again, this function is not tractable for continuous search spaces $\mathcal{X}$ so approximations must be made. Early work discretized the space $\mathcal{X}$ and computed the conditional entropy via Monte Carlo sampling [156]. More recent work uses a discretization of the $\mathcal{X}$ to obtain a smooth approximation to $p_\star$ and its expected information gain [67]. This method is unfortunately $O(M^4)$ where M is the number of discrete so-called representer points.

Finally, predictive entropy search (PES) removes the need for a discretization and approximates the acquisition function in $O((n+d)^3)$ time, which, for $d < n$ is of the same order as EI [69]. This is achieved by using the symmetric property of mutual information to rewrite $\alpha_{\mathrm{ES}}(\mathbf{x})$ as

$$\alpha_{\mathrm{PES}}(\mathbf{x}; \mathcal{D}_n) := H(y \mid \mathcal{D}_n, \mathbf{x}) - \mathbb{E}_{\mathbf{x}^\star \mid \mathcal{D}_n}\left[H(y \mid \mathcal{D}_n, \mathbf{x}, \mathbf{x}^\star)\right]$$

The expectation can be approximated via Monte Carlo with Thompson samples; and three simplifying assumptions are made to compute $H(y \mid \mathcal{D}_n, \mathbf{x}, \mathbf{x}^\star)$. Empirically, this algorithm has been shown to perform as well or better than the discretized version without the unappealing quartic term [69], making it arguably the state of the art in entropy search approximation.

D. Portfolios of acquisition functions

No single acquisition strategy provides better performance over all problem instances. In fact, it has been empirically observed that the preferred strategy can change at various stages of the sequential optimization process. To address this issue, [73] proposed the use of a portfolio containing multiple acquisition strategies. At each iteration, each strategy in the portfolio provides a candidate query point and meta-criterion is used to select the next query point among these candidates. The meta-criterion is analogous to an acquisition function at a higher level; whereas acquisition functions are optimized in the entire input space, a meta-criterion is only optimized within the set of candidates suggested by its base strategies.

The earlier approach of Hoffman et al. is based on a modification of the well-known Hedge algorithm [9], designed for the full-information adversarial multi-armed bandit. This particular portfolio algorithm relies on using the past performance of each acquisition function to predict future performance, where performance is measured by the objective function. However, this performance metric does not account for valuable information that is gained through exploration.

A more recent approach, the so-called entropy search portfolio (ESP), considers the use of an information-based metric instead [139]. In contrast to the GP-Hedge portfolio, ESP selects among different candidates by considering the gain of information towards the optimum. Removing the constant entropy at the current time, the ESP meta-criterion reduces to

$$\alpha_{\mathrm{ESP}}(\mathbf{x}; \mathcal{D}_n) = -\mathbb{E}_{y \mid \mathcal{D}_n, \mathbf{x}}\Big[H\big[\mathbf{x}_\star \mid \mathcal{D}_n \cup \{(\mathbf{x}, y)\}\big]\Big] \tag{48}$$

$$\mathbf{x}_n = \operatorname*{arg\,max}_{\mathbf{x}_{1:K,n}} \alpha_{\mathrm{ESP}}(\mathbf{x}; \mathcal{D}_n), \tag{49}$$

where $\mathbf{x}_{1:K,n}$ represent the candidates provided by the K base acquisition functions. In other words the candidate selected by this criterion is the one that results in the greatest expected reduction in entropy about the minimizer $\mathbf{x}_\star$. If the meta-criterion $\alpha_{\mathrm{ESP}}(\mathbf{x} | \mathcal{D}_n)$ were minimized over the entire space $\mathcal{X}$, ESP reduces to the acquisition functions proposed by [156], [67], [69]. However, ESP restricts this minimization to the set of candidates made by each portfolio member.

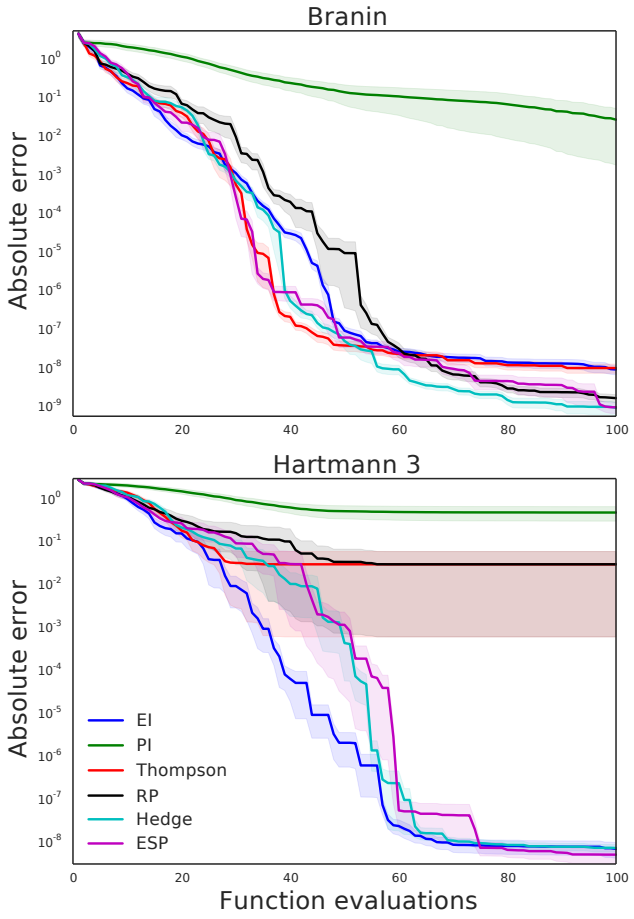

Fig. 6. Absolute error of the best observation for the Branin and Hartmann 3 synthetic functions. Plotting the mean and standard error (shaded area) over 25 repeated runs.

V. Practical considerations

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

A. Handling hyperparameters

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

$$\alpha_n(x) := \mathbb{E}_{\theta|\mathcal{D}_n}\left[\alpha(x;\theta)\right] = \int \alpha(x;\theta)p(\theta \,|\, \mathcal{D}_n)\mathrm{d}\theta. \tag{50}$$

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

$$p(\theta \,|\, \mathcal{D}_n) = \frac{p(\mathbf{y} \,|\, \mathbf{X}, \theta)p(\theta)}{p(\mathcal{D}_n)}. \tag{51}$$

The simplest approach to tackling (50) is to fit the hyperparameter to observed data using a point estimate $\hat{\theta}_n^{\mathrm{ML}}$ or $\hat{\theta}_n^{\mathrm{MAP}}$, corresponding to type II maximum likelihood or maximum *a posteriori* estimates, respectively. The posterior is then replaced by a delta measure at the corresponding $\hat{\theta}_n$ which yields

$$\hat{\alpha}_n(x) = \alpha(x;\hat{\theta}_n). \tag{52}$$

The estimators $\hat{\theta}_n^{\mathrm{ML}}$ and $\hat{\theta}_n^{\mathrm{MAP}}$ can be obtained by optimizing the marginal likelihood or the unnormalized posterior, respectively. For certain priors and likelihoods, these quantities as well as their gradients can be computed analytically. For example, the GP regression model yields the following marginal likelihood defined in (35), which we denote here by $\mathcal{L}_n$. Therefore it is common to use multi-started quasi-Newton hill-climbers (e.g., the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method) on objectives such as the likelihood $\mathcal{L}_n$ or the unnormalized posterior.

In Bayesian optimization, our uncertainty about the response surface plays a key role in guiding exploration and therefore it is important to incorporate our uncertainty about θ in the regression model. Naturally, these point estimates cannot capture this uncertainty. For this reason we consider marginalizing out the hyperparameters using either quadrature or Monte Carlo [120], [26], [143].

The common component in Monte Carlo (MC) methods is that they approximate the integral in (50) using M samples $\{\theta_n^{(i)}\}_{i=1}^M$ from the posterior distribution $p(\theta \,|\, \mathcal{D}_n)$:

$$\mathbb{E}_{\theta|\mathcal{D}_n}\left[\alpha(x;\theta)\right] \approx \frac{1}{M}\sum_{i=1}^{M}\alpha(x;\theta_n^{(i)}). \tag{53}$$

However, in practice it is impossible to sample directly from the posterior so Markov chain Monte Carlo (MCMC) and sequential Monte Carlo (SMC) techniques are used to produce a sequence of samples that are marginally distributed according to $p(\theta \,|\, \mathcal{D}_n)$ in the limit of infinitely long chains. Once the M hyperparameter samples are obtained, the acquisition function is evaluated and averaged over all samples; this marginal acquisition function incorporates the uncertainty in θ. In addition to MC methods, one could also use quadrature as shown in [120]. Here, samples (not necessarily drawn from the posterior) are combined using a weighted mixture:

$$\mathbb{E}_{\theta|\mathcal{D}_n}\left[\alpha(x;\theta)\right] \approx \sum_{i=1}^{M}\omega_i\alpha(x;\theta_n^{(i)}). \tag{54}$$

We could do away with samples entirely and approximately integrate out the hyperparameters as shown in [53]. To make the integral tractable, the authors adopted a linear approximation to the likelihood which enables them to derive an approximate posterior. This method, however, has not been demonstrated in the setting of Bayesian optimization.

Estimating the hyperparameters of GP kernels with very few function evaluations is a challenging task, often with disastrous

consequences as illustrated by a simple example in [15]. The typical estimation of the hyperparameters by maximizing the marginal likelihood [126], [82] can easily fall into traps, as shown in [32]. Several authors have proposed to integrate out the hyperparameters using quadrature or Monte Carlo methods [120], [26], [143]. These more advanced techniques can still fall in traps as illustrated with a simple simulation example in [157], where theoretical bounds are used to ensure that Bayesian optimization is robust with respect to the choice of hyperparameters.

B. Optimizing acquisition functions

A central step of the Bayesian optimization framework is the maximization of the acquisition function. Naturally, an acquisition function is only useful if it is cheap to evaluate relative to the objective function f. Nevertheless, the acquisition function is often multimodal and maximizing it is not a trivial task. In practice, the community has resorted to using various techniques such as discretization [143] and adaptive grids [13], or similarly, the divided rectangles approach of [83], which was used in [28], [110], [105]. When gradients are available, or can be cheaply approximated, one can use a multi-started quasi-Newton hill-climbing approach [100], [143]. Alternatively, [16] and [159] use the CMA-ES method of [66], and [79] apply multi-start local search.

Unfortunately, these auxiliary optimization techniques can be problematic for several reasons. First, in practice it is difficult to assess whether the auxiliary optimizer has found the global maximizer of the acquisition function. This raises important concerns about the convergence of Bayesian optimization algorithms because theoretical guarantees are only valid with the assumption that the exact optimizer is found and selected; see for example [146], [154] and [32]. Second, between any two consecutive iterations of the Bayesian optimization algorithm, the acquisition function may not change dramatically. Therefore, rerunning the auxiliary optimizer can be unnecessarily wasteful.

Recent proposed optimistic optimization methods provide an alternative to Bayesian optimization [87], [31], [116]. These methods sequentially build space-partitioning trees by splitting leaves with high function values or upper confidence bounds; the objective function is then evaluated at the centre of the chosen leaves. Simultaneous optimistic optimization (SOO) can reach the global optimum without knowledge of the function's smoothness [116]. Since SOO is optimistic at multiple scales (i.e., it expands several leaves simultaneously, with at most one leaf per level) it has also been referred to as multi-scale optimistic optimization [158].

Though these optimistic optimization methods do not require any auxiliary optimization, these methods are not as competitive as Bayesian optimization in practical domains where prior knowledge is available. The Bayesian multi-scale SOO (BamSOO) algorithm combines the tree partitioning idea of SOO with the surrogate model of Bayesian optimization [158], eliminating the need for auxiliary optimization. BamSOO also boasts some theoretical guarantees that do not depend on the exact optimization of an acquisition function.

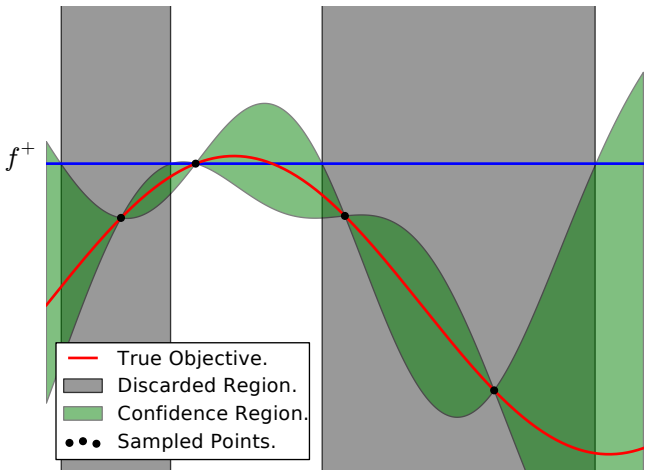

Fig. 7. Conditioned on the unknown objective function (red) lying between the surrogate confidence bounds (green region) with high probability, we can discard regions of the space where the upper bound is lower than the best lower bound encountered thus far. Figure from [43].

Intuitively, the method implements SOO to optimize the objective function directly, but avoids querying points that are deemed unlikely to be optimal by the surrogate model's confidence bounds.

In other words, BaMSOO uses the surrogate model to reduce the number of function evaluations, increasing sample efficiency. This work is also reminiscent of the theoretical work in [43], which proposes to only search in regions where the upper bound on the objective is greater than the best lower bound encountered thus far. Figure 7 illustrates how regions are discarded. Guided by the probabilistic model, the most promising regions are explored first, which avoids covering the entire space. Figure 8 compares SOO and BaMSOO on a simple one-dimensional example. Incorporating the surrogate model leads to better more refined optimization for the same number of query points.

C. Conditional Spaces

It is often the case that some variables will only influence the function being optimized when other variables take on certain values. These are called conditional variables and are said to be *active* or *inactive*. For example, when the function involves selecting between different algorithms as well as optimizing their hyperparameters, then certain sets of hyperparameters belonging to a given algorithm will be inactive if that algorithm is not selected [79], [16].

More formally, consider a variable $x_1 \in \mathcal{X}_2$ and another variable $x_2 \in \mathcal{X}_2$. x_1 is said to be a child of x_2 if it is only active when x_2 takes on certain values in $\mathcal{X}_2$. This conditional structure can be extended with multiple variables to form more complicated tree or directed acyclic graph structures. This greatly extends the capabilities of the Bayesian optimization framework, allowing it to chain together individual algorithms to form sophisticated pipelines that can be jointly optimized [151], [19].

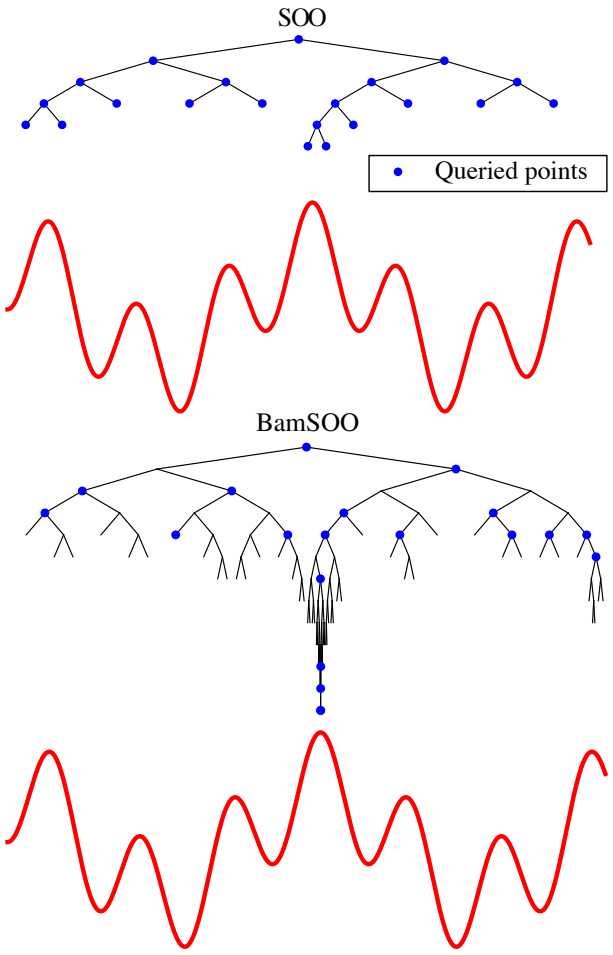

Fig. 8. Comparison of SOO (top) and BamSOO (bottom) on $f(x) = \frac{1}{2}\sin(15x)\sin(27x)$ in $[0, 1]$. Blue dots represent nodes where the objective was evaluated. BaMSOO does not evaluate f at points that are suboptimal with high probability under the surrogate model (not shown). Figure from [158].

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

Gaussian processes are not immediately suitable for conditional spaces because standard kernels are not defined over variable-length spaces. A simple approach is to define a separate GP for each group of jointly active hyperparameters [16], however this ignores dependencies between groups. Recent work has focused on defining a fixed-length embedding of conditional spaces where a standard kernel using Euclidean distance can be applied [147]. This is currently a very new area of research and more work needs to be done before GPs can work in conditional spaces as well as tree-based models.

D. Non-stationarity

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

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

a) Non-stationary kernels: One way to create a non-stationary process is to use a non-stationary kernel. One strategy is to convert a stationary kernel into a non-stationary one by transforming $\mathbf{x}$ using a parametric warping function $\mathbf{x}^{(w)} = w(\mathbf{x})$ and then applying a stationary kernel to $\mathbf{x}^{(w)}$ [129], [145]. If w is chosen appropriately, the data will follow a stationary process in the transformed space.

In Bayesian optimization, the inputs are traditionally projected onto the unit hypercube and this fact was exploited in [145], who chose the warping function to be the cumulative distribution function (CDF) of the beta distribution,

$$w_d(\mathbf{x}) = \frac{\mathbf{x}_d^{\alpha-1}(1-\mathbf{x}_d)^{\beta-1}}{B(\alpha,\beta)}, \tag{55}$$

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

Examples of functions before and after applying beta warping are shown in Figure 9. Despite having only two parameters, the beta CDF is able to express a wide variety of transformations. These transformations contract portions of the input space, and expand others, which has the effect of decreasing and increasing the length scale in those portions, respectively. The beta warping approach has been shown to be highly effective on several benchmark problems as well as hyperparameter optimization for machine learning models [145], [18].

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

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

$$w_d(\mathbf{x}) = 1 - (1 - \mathbf{x}_d^{\alpha})^{\beta}. \tag{56}$$

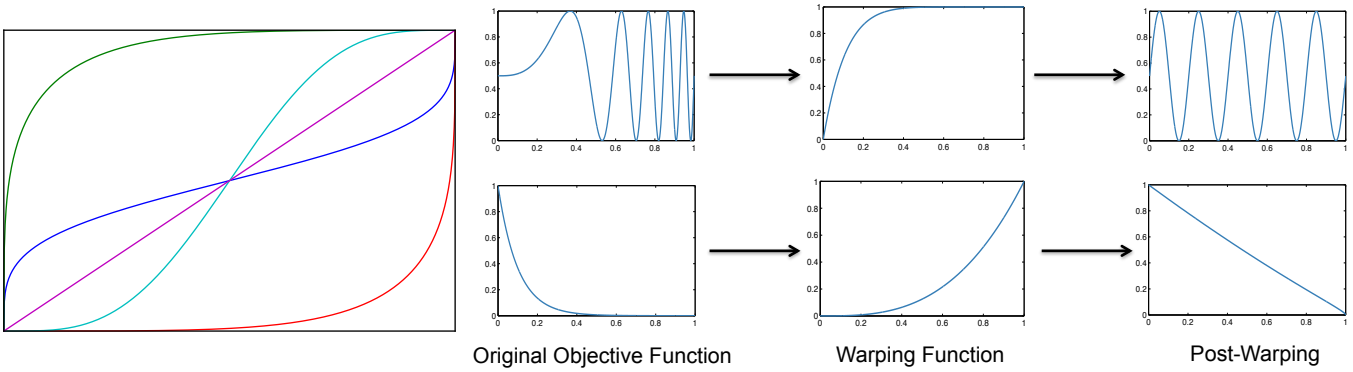

Fig. 9. **Left:** Examples of beta CDF warpings under different settings of the shape parameters α and β. **Right:** Examples of functions after applying a beta CDF warping (originally from [145]). The regions where the CDF has a slope greater than 1 are expanded along the horizontal axis, while regions where the CDF has slope less than 1 are contracted.

There are many other examples of non-stationary covariance functions [70], [121], [132], [126], [118], [22], [3], [54] that have been proposed for GP regression along with closely related output warping techniques [141] that can also model certain kinds of non-stationary processes.

b) Partitioning: An alternative approach to modelling non-stationarity that has been useful in practice is to partition the space into distinct regions and then to model each region as a separate stationary process. In a random forest model [79], [27], this is achieved by finer partitioning in regions of the space where the function changes rapidly, and more granular partitioning in regions where the function changes slowly. Partitioning can also be an effective strategy for GPs. For example, [63] proposed the treed GP model, which partitions the data and then applies a separate GP to each region.

c) Heteroscedasticity: Heteroscedasticity is a close analogue of non-stationarity, but refers to non-stationary behaviour in the *noise* process governing the observation model, instead of the true process that we wish to capture. Standard GP regression using an isotropic noise kernel assumes by default that the noise process is constant everywhere, and is therefore stationary by definition. In practice, it is possible to have non-stationarity in both the true process and the noise process. Heteroscedasticity has been widely addressed in the GP literature, see e.g., [95], [86], [103].

For Bayesian optimization in particular, one approach to handling heteroscedastic noise was proposed in [5] using a partitioning approach. The idea is to build a partition using classification and regression trees (CART) [25]; however, splitting was restricted to occur at data points rather than between them. This ensured that the variance estimates of the GP would remain smooth between partitions.

Another form of non-stationarity that is closely related to heteroscedasticity is a non-stationary amplitude [1], [54]. This is where the magnitude of the output process changes as a function of the input. To our knowledge this has not been directly addressed in the Bayesian optimization literature. There have however been attempts to be robust to this effect by integrating out the amplitude parameter of the GP kernel. This was done numerically in [143] and analytically using conjugate priors in [138], resulting in a latent GP with t-distributed predictions and an input-dependent noise covariance.

E. Parallelization

Bayesian optimization is conventionally posed as a sequential problem where each experiment is completed before a new one is proposed. In practice it may be possible and advantageous to run multiple function evaluations in parallel. Even if the number of experiments required to reach the minimum does not change, parallel approaches can yield a substantial reduction in terms of wall-clock time [80], [143].

Ginsbourger et al. [57] proposed several approaches based on imputing the results of currently running experiments. The idea is that given the current observations $\mathcal{D}_n = \{(x_n, y_n)\}$ and pending experiments $\mathcal{D}_p = \{x_p\}$, one can impute a set of set of experimental outcomes $\tilde{\mathcal{D}}_p = \{(x_p, \tilde{y}_p)\}$ and then perform a step of Bayesian optimization using the augmented dataset $\mathcal{D}_t \cup \tilde{\mathcal{D}}_p$.

One simple strategy is the *constant liar*, where a constant L is chosen such that $\tilde{y}_p = L$, $\forall p$. Another strategy is the *Kriging believer*, which uses the GP predictive mean $\tilde{y}_p = \mu_n(x_p)$. [143] used an approach where a set of S fantasies are sampled for each unfinished experiment from the full GP posterior predictive distribution. These are then combined to estimate the following parallel integrated acquisition function,

$$\alpha(x; \mathcal{D}_n, \mathcal{D}_p) = \int_{\mathbb{R}^J} \alpha(x; \mathcal{D}_n \cup \tilde{\mathcal{D}}_p) P(\tilde{y}_{1:J}; \mathcal{D}_n) dy_{p1:J}, \quad (57)$$

$$\approx \frac{1}{S} \sum_{s=1}^{S} \alpha(x; \mathcal{D}_n \cup \tilde{\mathcal{D}}_p^{(s)}), \quad (58)$$

$$\tilde{\mathcal{D}}_p^{(s)} \sim P(\tilde{y}_{1:J}; \mathcal{D}_n), \quad (59)$$

where J is the number of currently pending experiments. This approach has been shown to be very effective in practice when α is chosen to be EI. Similar approaches are proposed in [58], [40] and a similar parallel extension to GP-UCB is proposed [45].

Although the imputation approaches deal with parallel experiments, the nature in which they propose candidates is still inherently sequential. A truly parallel approach would

simultaneously propose a set of candidates. Jones [81] and Hutter et al. [80] proposed an approach based on GP-UCB where α_{UCB} is optimized using a range of β_n values, which produces a set of points that favour a range of exploration and exploitation.

F. Software implementations

As of this writing, there are several open source packages implementing various forms of Bayesian optimization. We highlight several popular libraries in Table I.

VI. Theory of Bayesian optimization

There exist a vast literature on the theoretical properties of bandit algorithms in general. Theoretical properties of Bayesian optimization, however, have only been established recently. In this section, we focus on the results concerning Gaussian process based Bayesian optimization and defer detailed discussions of bandit algorithms to other dedicated surveys [29], [117].

There exist several early consistency proofs for Gaussian process based Bayesian optimization algorithms, in the one-dimensional setting [102] and one for a simplification of the algorithm using simplicial partitioning in higher dimensions [164]. The consistency of the algorithm using multivariate Gaussian processes has been established in [155].

More recently, [146] provided the first finite sample bound for Gaussian process based Bayesian optimization. In this work, the authors showed that the GP-UCB algorithm suffers from sub-linear cumulative regret in the stochastic setting. The regret bounds, however, allow only fixed hyperparameters. In [32], Bull provided both upper and lower bounds of simple regret for the EGO algorithm [82] in the deterministic setting. In addition to regret bounds concerning fixed hyperparameters, the author also provided simple regret bounds while allowing varying hyperparameters.

Since the pioneering work of [146] and [32], there emerged a large body of results on this topic including, exponentially vanishing simple regret bounds in the deterministic setting [43]; bounds for contextual Gaussian process bandits [89]; Bayes regret bounds for Thompson sampling [85], [127]; bounds for high-dimensional problems with a underlying low-rank structure [46]; bounds for parallel Bayesian optimization [45]; and improved regret bounds using mutual information [41].

Despite the recent surge in theoretical contributions, there is still a wide gap between theory and practice. Regret bounds or even consistency results, for example, have not been established for approaches that use a full Bayesian treatment of hyperparameters [143]. Such theoretical results could advance the field of Bayesian optimization and provide insight for practitioners.

VII. History of Bayesian optimization and related approaches

Arguably the earliest work related to Bayesian optimization was that of William Thompson in 1933 where he considered the likelihood that one unknown Bernoulli probability is greater than another given observational data [150]. In his article, Thompson argues that when considering, for example, two alternative medical treatments one should not eliminate the worst one based on a single clinical trial. Instead, he proposes, one should estimate the probability that one treatment is better than the other and weigh future trials in favour of the seemingly better treatment while still trying the seemingly suboptimal one. Thompson rightly argues that by adopting a single treatment following a clinical trial, there is a fixed chance that all subsequent patients will be given suboptimal treatment. In contrast, by dynamically selecting a fraction of patients for each treatment, this sacrifice becomes vanishingly small.

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

Although modern experimental design had been developed a decade earlier by Ronald Fisher's work on agricultural crops, Thompson introduced the idea of making design choices dynamically as new evidence becomes available; a general strategy known as sequential experimental design or, in the multi-armed bandit literature, adaptive or dynamic allocation rules [92], [59].

The term Bayesian optimization was coined in the seventies [115], but a popular version of the method has been known as efficient global optimization in the experimental design literature since the nineties [134]. Since the approximation of the objective function is often obtained using Gaussian process priors, the technique is also referred to as Gaussian process bandits [146].

In the nonparametric setting, Kushner [91] used Wiener processes for unconstrained one-dimensional optimization problems. Kushner's decision model was based on maximizing the probability of improvement. He also included a parameter that controlled the trade-off between 'more global' and 'more local' optimization, in the same spirit as the exploration–exploitation trade-off. Meanwhile, in the former Soviet Union, Močkus and colleagues developed a multidimensional Bayesian optimization method using linear combinations of Wiener fields [115], [114]. Both of these methods, probility of and expected improvement, were in studied in detail in [81].

At the same time, a large, related body of work emerged under the name *kriging*, in honour of the South African student who developed this technique at the University of the Witwatersrand [90], though largely popularized by Matheron and colleagues (e.g., [111]). In kriging, the goal is interpolation of a random field via a linear predictor. The errors on this model are typically assumed to *not* be independent, and are modelled with a Gaussian process.

Kriging has been applied to experimental design under the name DACE, after design and analysis of computer ex-

TABLE I
A LIST OF SEVERAL POPULAR OPEN SOURCE SOFTWARE LIBRARIES FOR BAYESIAN OPTIMIZATION AS OF MAY, 2015.

Package	License	URL	Language	Model
SMAC	Academic non-commercial license.	http://www.cs.ubc.ca/labs/beta/Projects/SMAC	Java	Random forest
Hyperopt	BSD	https://github.com/hyperopt/hyperopt	Python	Tree Parzen estimator
Spearmint	Academic non-commercial license.	https://github.com/HIPS/Spearmint	Python	Gaussian process
Bayesopt	GPL	http://rmcantin.bitbucket.org/html	C++	Gaussian process
PyBO	BSD	https://github.com/mwhoffman/pybo	Python	Gaussian process
MOE	Apache 2.0	https://github.com/Yelp/MOE	Python / C++	Gaussian process

periments, the title of a paper by Sacks et al. [128] (and more recently a book by Santner et al. [130]). In DACE, the regression model is a best linear unbiased predictor (BLUP), and the residual model is a noise-free Gaussian process. The goal is to find a design point or points that optimizes some criterion. Experimental design is usually non-adaptive: the entire experiment is designed before data is collected. However, sequential design is an important and active subfield (e.g., [160], [33].

The efficient global optimization (EGO) algorithm is the combination of DACE model with the sequential expected improvement acquisition criterion. It was published in a paper by Jones et al. [82] as a refinement of the SPACE algorithm (stochastic process analysis of computer experiments) [133]. Since EGO's publication, there has evolved a body of work devoted to extending the algorithm, particularly in adding constraints to the optimization problem [6], [131], [23], and in modelling noisy functions [14], [75], [76].

In the bandits setting, Lai and Robbins [92] introduced upper confidence bounds (UCB) as approximate alternatives to Gittins indices in 1985. Auer studied these bounds using frequentist techniques, and in adversarial multi-armed bandit settings [9], [8].

The literature on multi-armed bandits is vast. The book of Cesa-Bianchi [36] is a good reference on the topic of online learning with experts and bandits in adversarial settings. There are many results on exploration [30], [51], [50] and contextual bandits [97], [112], [2]. These contextual bandits, may also be seen as myopic approximations to Markov decision processes.

VIII. EXTENSIONS AND OPEN QUESTIONS

A. Constrained Bayesian optimization

In [56] a scenario was outlined in which a food company wished to design the best tasting cookie subject to the number of calories being below a certain level. This is an example of constrained optimization, where certain regions of the design space $\mathcal{X}$ are invalid. In machine learning, this can arise when certain hyperparameter configurations result in models that diverge during training, or that run out of computer memory. When the constraints are known *a priori*, they can be incorporated into the optimization of the acquisition function. The more challenging case arises when it is not known in advance which configurations will result in a constraint violation. Several approaches deal with this problem by altering the acquisition function itself.

Gramacy and Lee [62] proposed the integrated expected conditional improvement (IECI) acquisition function:

$$\alpha_{\mathrm{IECI}}(\mathbf{x}) = \int_{\mathbf{x}'} (\alpha_{\mathrm{EI}}(\mathbf{x}', \mathcal{D}_n) - \alpha_{\mathrm{EI}}(\mathbf{x}', \mathcal{D}_n \cup \mathbf{x})|\mathbf{x}))h(\mathbf{x}')d\mathbf{x}. \tag{60}$$

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

Snoek [142] and Gelbart et al. [56] proposed the weighted expected improvement criterion (wEI) that multiplies EI by the probability of satisfying the constraints:

$$\alpha_{\mathrm{wEI}}(\mathbf{x}) = \alpha_{\mathrm{EI}}(\mathbf{x}, \mathcal{D}_n)h(\mathbf{x}, \mathcal{D}_n). \tag{61}$$

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

A variant of wEI was proposed in [52] to deal with the case where the function is constrained to be less than some value λ. They used $h(\mathbf{x}, \mathcal{D}_t) = \mathbb{P}(f(x) < \lambda \mid \mathcal{D}_t)$, the posterior probability of satisfying this constraint under the Gaussian process model of the function.

Hernández-Lobato et al. [68] recently proposed a variation of the predictive entropy search acquisition function to deal with the decoupled case, where the function and constraints can be evaluated independently.

In a different approach, Gramacy et al. [61] adapted the augmented Lagrangian approach to the Bayesian optimization setting, with unconstrained Bayesian optimization approximately solving the inner loop of the algorithm.

B. Cost-sensitivity

In some cases, each function evaluation may return both a value along with an associated cost. In other words, it may be more expensive to evaluate the function in some parts of the design space than others. If there is a limited budget, then the search should be biased toward low-cost areas. In [143], the goal was to train a machine learning model and the cost was the time it took to train the model. They used expected improvement per second, $\mathrm{EI}(\mathbf{x}, \mathcal{D}_n)/c(\mathbf{x})$ in order to bias the search toward good models with fast training times. Here, $c(\mathbf{x})$ was the estimated cost of querying the objective at $\mathbf{x}$ and was modelled using a Gaussian process with response $\log(c(\mathbf{x}))$.

C. High-dimensional problems

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

For *linear* bandits, Carpentier et al. [35] recently proposed a compressed sensing strategy to attack problems with a high degree of sparsity. Also recently, Chen et al. [39] made significant progress by introducing a two stage strategy for optimization and variable selection of high-dimensional GPs. In the first stage, sequential likelihood ratio tests with a couple of tuning parameters are used to select the relevant dimensions. This, however, requires the relevant dimensions to be axis-aligned with an ARD kernel. Chen et al. provide empirical results only for synthetic examples (of up to 400 dimensions), but they provide key theoretical guarantees.

Hutter et al. [79] used Bayesian optimization with random forests based on frequentist uncertainty estimates. Their method does not have theoretical guarantees for continuous optimization, but it achieved state-of-the-art performance for tuning up to 76 parameters of algorithms for solving combinatorial problems. Note that in constructing the trees that make the forest, one samples and selects the most promising features (dimensions). That is, random forests naturally select the relevant dimensions of the problem, and so not surprisingly have worked well in practice.

Many researchers have noted that for certain classes of problems most dimensions do not change the objective function significantly; examples include hyperparameter optimization for neural networks and deep belief networks [17] and automatic configuration of state-of-the-art algorithms for solving $\mathcal{NP}$-hard problems [77]. That is to say these problems have low *effective dimensionality*. To take advantage of this property, Bergstra and Bengio [17] proposed to simply use random search for optimization – the rationale being that points sampled uniformly at random in each dimension can densely cover each low-dimensional subspace. As such, random search can exploit low effective dimensionality *without knowing which dimensions are important*. In [159], the authors exploit the same property, while still capitalizing on the strengths of Bayesian optimization. By combining randomization with Bayesian optimization, they were able to derive a new approach that outperforms each of the individual components.

Figure 10 illustrates the approach in a nutshell. Assume we know that a given $D = 2$ dimensional black-box function $f(x_1, x_2)$ only has $d = 1$ important dimensions, but we do not know which of the two dimensions is the important one. We can then perform optimization in the embedded 1-dimensional subspace defined by $x_1 = x_2$ since this is guaranteed to include the optimum. This idea enables us to perform Bayesian optimization in a low-dimensional space to optimize a high-dimensional function with low intrinsic dimensionality. Importantly, it is not restricted to cases with axis-aligned intrinsic dimensions.

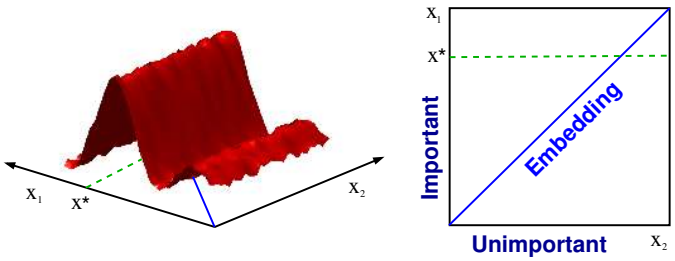

Fig. 10. This function in $D = 2$ dimesions only has $d = 1$ *effective dimension*: the vertical axis indicated with the word important on the right hand side figure. Hence, the one-dimensional embedding includes the two-dimensional function's optimizer. It is more efficient to search for the optimum along the one-dimensional random embedding than in the original two-dimensional space.

To make the discussion more precise, a function $f : \mathbb{R}^D \mapsto \mathbb{R}$ will have *effective dimensionality* d_e, with $d_e < D$, if there exists a linear *effective subspace* $\mathcal{T}$ of dimension d_e such that for all $\mathbf{x}_\top \in \mathcal{T} \subset \mathbb{R}^D$ and $\mathbf{x}_\perp \in \mathcal{T}^\perp \subset \mathbb{R}^D$, and $f(\mathbf{x}) = f(\mathbf{x}_\top + \mathbf{x}_\perp) = f(\mathbf{x}_\top)$, where the so-called *constant subspace* $\mathcal{T}^\perp$ denotes the orthogonal complement of $\mathcal{T}$. This definition simply states that the function does not change along the coordinates $\mathbf{x}_\perp$, and hence the name for $\mathcal{T}^\perp$.

Given this definition, Theorem 1 of [159] shows that problems of low effective dimensionality can be solved via random embedding. The theorem assumes we are given a function $f : \mathbb{R}^D \mapsto \mathbb{R}$ with effective dimensionality d_e and a random matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$ with independent entries sampled according to $\mathcal{N}(0, 1)$ and $d \geq d_e$. It then shows that, with probability 1, for any $\mathbf{x} \in \mathbb{R}^D$, there exists a $\mathbf{z} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = f(\mathbf{Az})$.

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

Algorithm 3 REMBO

1: Generate a random matrix $\mathbf{A}$
2: Choose the set $\mathcal{Z}$
3: **for** $n = 1, 2, \ldots$ **do**
4: select $\mathbf{z}_{n+1}$ by optimizing the acquisition function α:

$$\mathbf{z}_{n+1} = \arg\max_{\mathbf{z}\in\mathcal{Z}} \alpha(\mathbf{z}|\mathcal{D}_n)$$

5: augment the data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{z}_{n+1}, f(\mathbf{Az}_{n+1})\}$
6: update the kernel hyperparameters
7: **end for**

An important detail is how REMBO chooses the bounded region $\mathcal{Z}$, inside which it performs Bayesian optimization. This is important because its effectiveness depends on the size of $\mathcal{Z}$.

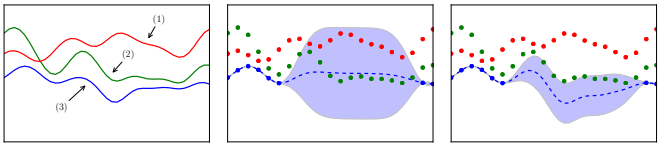

Fig. 11. **Left:** Three correlated functions drawn from a multi-output GP. **Middle:** The GP posterior predictive distribution of function (3) when the functions are assumed to be independent. This is equivalent to ignoring observations from functions (1) and (2). **Right:** Posterior predictive distribution of function (3) when the correlations are taken into account. Here, the observations from functions (1) and (2) act as weak observations for function (3). This results in a much more accurate prediction.

Locating the optimum within $\mathcal{Z}$ is easier if $\mathcal{Z}$ is small, but if we set $\mathcal{Z}$ too small it may not actually contain the global optimizer. We refer the readers to the original paper for details.

D. Multi-Task

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

There have been several attempts to exploit this property within the Bayesian optimization framework [89], [79], [12], [148], [161], [49]. The idea is that there are several correlated functions, $\mathcal{T} = \{1, 2, \ldots, M\}$, called *tasks* and that we are interested in optimizing some subset of these tasks. In essence, the data from one task can provide information about another task.

One way to share information between tasks in a Bayesian optimization routine is to modify the underlying Gaussian process model. There has been a great deal of work on extending Gaussian processes to the multi-task scenario. These extensions are also known as multi-output Gaussian processes. The key is to define a valid covariance over input and task pairs, $k((\mathbf{x}, m), (\mathbf{x}', m'))$. One method is to use the intrinsic model of coregionalization (ICM) [60], [136], [21] that utilizes the product kernel,

$$k((\mathbf{x}, m), (\mathbf{x}', m')) = k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}')k_{\mathcal{T}}(m, m'). \tag{62}$$

Where $m,\ m' \in \mathcal{T}$. $k_{\mathcal{T}}$ defines the covariance between tasks. There are many ways to parameterize the task covariance function [123]. Figure 11 illustrates how knowledge of correlations between tasks can be used with a multi-output GP to make more accurate predictions.

An alternative view of the ICM model is that it defines a latent process that is rotated and scaled to produce each of the individual tasks. The problem of defining a multi-output GP can then be viewed as learning a latent function, or a set of latent functions, that can be transformed to produce the output tasks. [12] proposed an approach that learns a latent ranking function at each iteration using pairs of observations from within each task. By learning a single ranking function that works across tasks, the tasks are effectively jointly embedded in a latent space that is invariant to potentially different output scales across tasks.

Each task may come with additional side information, or *context* features. In this case, it is possible to define a joint model that uses this context. This was considered for algorithm configuration in [79] using a random forest model. When starting a new task, [49] uses task features to find similar tasks. The best inputs from the most similar tasks are then used as the initial design for the new task.

E. Freeze-Thaw

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

An attempt to incorporate this into the Bayesian optimization framework is given in [149]. They identify that many loss functions in machine learning follow an exponential decay pattern during training, and construct a basis set of exponentially decaying functions of the form $f(t, \lambda) = e^{-\lambda t}$, where λ represents the rate of decay over time, represented by t, in order to forecast model performance. It is possible to construct a nonstationary kernel from this basis set:

$$k(t, t') = \frac{\beta^{\alpha}}{(t + t' + \beta)^{\alpha}}, \tag{63}$$

where α and β are hyperparameters that control the shape of the kernel. This kernel is used within a Gaussian process to jointly model $(\mathbf{x}, t)$ pairs. Given the ability to forecast curves, [149] then uses an entropy search-based acquisition function in order to determine whether to *freeze* a currently running experiment, *thaw* a previous experiment in order to resume training, or start a new experiment.

Rather than constructing a kernel, [48], [47] built a basis set manually based on previously collected training curves. This basis set is then used with Bayesian linear regression in order to forecast training curves, and an early stopping rule is given based on the probability of improvement using the forecasted value.

An alternative view of this procedure is to consider Gaussian process models that incorporate partial feedback. This view is used in [122], where they construct a Gaussian process with non-stationary noise process that starts high when the experiment begins, and decays over time.

IX. Concluding Remarks

In this paper we have introduced Bayesian optimization from a modelling perspective. Beginning with the Beta-Bernoulli and linear models, and extending them to non-parametric models, we recover a wide range of approaches

to Bayesian optimization that have been introduced in the literature. There has been a great deal of work that has focussed heavily on designing acquisition functions, however we have taken the perspective that the importance of this plays a secondary role to the choice of the underlying surrogate model.

In addition to outlining different modelling choices, we have considered many of the design decisions that are used to build Bayesian optimization systems. We further highlighted relevant theory as well as practical considerations that are used when applying these techniques to real-world problems. We provided a history of Bayesian optimization and related fields and surveyed some of the many successful applications of these methods. We finally discussed extensions of the basic framework to new problem domains, which often require new kinds of surrogate models.

Although the underpinnings of Bayesian optimization are quite old, the field itself is undergoing a resurgence, aided by new problems, models, theory, and software implementations. In this paper, we have attempted to summarize the current state of Bayesian optimization methods; however, it is clear that the field itself has only scratched the surface and that there will surely be many new problems, discoveries, and insights in the future.

REFERENCES

[1] R. P. Adams and O. Stegle. Gaussian process product models for nonparametric nonstationarity. In *International Conference on Machine Learning*, pages 1–8, 2008.

[2] S. Agrawal and N. Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, 2013.

[3] E. B. Anderes and M. L. Stein. Estimating deformations of isotropic Gaussian random fields on the plane. *The Annals of Statistics*, 36(2):719–741, 2008.

[4] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

[5] J.-A. M. Assael, Z. Wang, and N. de Freitas. Heteroscedastic treed Bayesian optimisation. *arXiv preprint arXiv:1410.7172*, 2014.

[6] C. Audet, J. Jr, Dennis, D. W. Moore, A. Booker, and P. D. Frank. Surrogate-model-based method for constrained optimization. In *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2000.

[7] J. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *Conference on Learning Theory*, 2010.

[8] P. Auer. Using confidence bounds for exploitation-exploration tradeoffs. *Journal of Machine Learning Research*, 3:397–422, 2003.

[9] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Symposium on Foundations of Computer Science*, pages 322–331, 1995.

[10] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[11] J. Azimi, A. Jalali, and X. Fern. Hybrid batch Bayesian optimization. In *International Conference on Machine Learning*, 2012.

[12] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In *International Conference on Machine Learning*, 2013.

[13] R. Bardenet and B. Kégl. Surrogating the surrogate: accelerating Gaussian-process-based global optimization with a mixture cross-entropy algorithm. In *International Conference on Machine Learning*, pages 55–62, 2010.

[14] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *Proc. CEC-05*, 2005.

[15] R. Benassi, J. Bect, and E. Vazquez. Robust Gaussian process-based global optimization using a fully Bayesian expected improvement criterion. In C. Coello, editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin / Heidelberg, 2011.

[16] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

[17] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[18] J. Bergstra, B. Komer, C. Eliasmith, and D. Warde-Farley. Preliminary evaluation of hyperopt algorithms on HPOLib. *International Conference on Machine Learning AutoML Workshop*, 2014.

[19] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, 2013.

[20] S. Bochner. *Lectures on Fourier integrals*. Princeton University Press, 1959.

[21] E. V. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems*, 2008.

[22] L. Bornn, G. Shaddick, and J. V. Zidek. Modeling nonstationary processes through dimension expansion. *Journal of the American Statistical Society*, 107(497), 2012.

[23] P. Boyle. *Gaussian Processes for Regression and Optimisation*. PhD thesis, Victoria University of Wellington, Wellington, New Zealand, 2007.

[24] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[25] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

[26] E. Brochu, T. Brochu, and N. de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 103–112, 2010.

[27] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report UBC TR-2009-23 and arXiv:1012.2599v1, Dept. of Computer Science, University of British Columbia, 2009.

[28] E. Brochu, N. de Freitas, and A. Ghosh. Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems*, pages 409–416, 2007.

[29] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[30] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *International Conference on Algorithmic Learning Theory*, 2009.

[31] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

[32] A. D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12:2879–2904, 2011.

[33] D. Busby. Hierarchical adaptive experimental design for Gaussian process emulators. *Reliability Engineering and System Safety*, 94(7):1183–1193, July 2009.

[34] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian processes for regression. *arXiv preprint arXiv:1402.5876*, 2014.

[35] A. Carpentier and R. Munos. Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In *AI and Statistics*, pages 190–198, 2012.

[36] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, 2006.

[37] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[38] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems*, pages 2249–2257, 2011.

[39] B. Chen, R. Castro, and A. Krause. Joint optimization and variable selection of high-dimensional Gaussian processes. In *International Conference on Machine Learning*, 2012.

[40] S. Clark. *Parallel Machine Learning Algorithms In Bioinformatics And Global Optimization*. PhD thesis, Cornell University, 2012.

[41] E. Contal, V. Perchet, and N. Vayatis. Gaussian process optimization with mutual information. In *International Conference on Machine Learning*, 2013.

[42] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, (7):81–227, 2011.

[43] N. de Freitas, A. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *International Conference on Machine Learning*, 2012.

[44] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. Learning where to attend with deep architectures for image tracking. *Neural Computation*, 24(8):2151–2184, 2012.

[45] T. Desautels, A. Krause, and J. Burdick. Parallelizing exploration-exploitation tradeoffs with Gaussian process bandit optimization. *Journal of Machine Learning Research*, 2014.

[46] J. Djolonga, A. Krause, and V. Cevher. High dimensional Gaussian process bandits. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2013.

[47] T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, July 2015.

[48] T. Domhan, T. Springenberg, and F. Hutter. Extrapolating learning curves of deep neural networks. In *International Conference on Machine Learning AutoML Workshop*, 2014.

[49] M. Feurer, T. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *National Conference on Artificial Intelligence (AAAI)*, 2015.

[50] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, pages 3212–3220, 2012.

[51] V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck. Multi-bandit best arm identification. In *Advances in Neural Information Processing Systems*, pages 2222–2230, 2011.

[52] J. R. Gardner, M. J. Kusner, Z. Xu, K. Q. Weinberger, and J. P. Cunningham. Bayesian optimization with inequality constraints. In *International Conference on Machine Learning*, pages 937–945, 2014.

[53] R. Garnett, M. A. Osborne, and P. Hennig. Active learning of linear embeddings for Gaussian processes. In *Uncertainty in Artificial Intelligence*, 2014.

[54] R. Garnett, M. A. Osborne, S. Reece, A. Rogers, and S. J. Roberts. Sequential Bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 53(9):1430–1446, 2010.

[55] R. Garnett, M. A. Osborne, and S. J. Roberts. Bayesian optimization for sensor set selection. In *ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219. ACM, 2010.

[56] M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. In *Uncertainty in Artificial Intelligence*, 2014.

[57] D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

[58] D. Ginsbourger and R. L. Riche. Dealing with asynchronicity in parallel Gaussian process based global optimization. http://hal.archives-ouvertes.fr/hal-00507632, 2010.

[59] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

[60] P. Goovaerts. *Geostatistics for natural resources evaluation*. Oxford University Press, 1997.

[61] R. B. Gramacy, G. A. Gray, S. L. Digabel, H. K. Lee, P. Ranjan, G. Wells, and S. M. Wild. Modeling an augmented Lagrangian for improved blackbox constrained optimization. *arXiv preprint arXiv:1403.4890*, 2014.

[62] R. B. Gramacy and H. K. Lee. Optimization under unknown constraints. *arXiv preprint arXiv:1004.4027*, 2010.

[63] R. B. Gramacy, H. K. H. Lee, and W. G. Macready. Parameter space exploration with Gaussian process trees. In *International Conference on Machine Learning*, pages 45–52, 2004.

[64] S. Grunewalder, J. Audibert, M. Opper, and J. Shawe-Taylor. Regret bounds for Gaussian process bandit problems. In *AI and Statistics*, pages 273–280, 2010.

[65] F. Hamze, Z. Wang, and N. de Freitas. Self-avoiding random dynamics on integer complex systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 23(1):9, 2013.

[66] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[67] P. Hennig and C. Schuler. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research*, 13(1):1809–1837, 2012.

[68] J. M. Hernández-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani. Predictive entropy search for Bayesian optimization with unknown constraints. In *International Conference on Machine Learning*, 2015.

[69] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*. 2014.

[70] D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. *Bayesian Statistics*, 6, 1998.

[71] G. E. Hinton and R. Salakhutdinov. Using deep belief nets to learn covariance kernels for Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1249–1256, 2008.

[72] M. Hoffman, B. Shahriari, and N. de Freitas. On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *AI and Statistics*, pages 365–374, 2014.

[73] M. W. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for Bayesian optimization. In *Uncertainty in Artificial Intelligence*, pages 327–336, 2011.

[74] H. H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, 2012.

[75] D. Huang, T. Allen, W. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential Kriging meta-models. *J. of Global Optimization*, 34(3):441–466, 2006.

[76] F. Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, Vancouver, Canada, 2009.

[77] F. Hutter, H. Hoos, and K. Leyton-Brown. Identifying key algorithm parameters and instance features using forward selection. In *LION*, 2013.

[78] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In *CPAIOR*, pages 186–202, 2010.

[79] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, pages 507–523, 2011.

[80] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Parallel algorithm configuration. In *LION*, pages 55–70, 2012.

[81] D. Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21(4):345–383, 2001.

[82] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *J. of Global optimization*, 13(4):455–492, 1998.

[83] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181, 1993.

[84] E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pages 592–600, 2012.

[85] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, volume 7568 of *Lecture Notes in Computer Science*, pages 199–213. Springer Berlin Heidelberg, 2012.

[86] K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic Gaussian process regression. In *International Conference on Machine Learning*, pages 393–400, 2007.

[87] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. 2006.

[88] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery*, 18(1):140–181, 2009.

[89] A. Krause and C. S. Ong. Contextual Gaussian process bandit optimization. In *Advances in Neural Information Processing Systems*, pages 2447–2455, 2011.

[90] D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of Chemical, Metallurgical, and Mining Society of South Africa*, 1951.

[91] H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering*, 86(1):97–106, 1964.

[92] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[93] M. Lázaro-Gredilla and A. R. Figueiras-Vidal. Marginalized neural network mixtures for large-scale regression. *Neural Networks, IEEE Transactions on*, 21(8):1345–1351, 2010.

[94] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

[95] Q. V. Le, A. J. Smola, and S. Canu. Heteroscedastic Gaussian process regression. In *International Conference on Machine Learning*, pages 489–496, 2005.

[96] K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In *Principles and Practice of Constraint Programming*, pages 556–572. Springer, 2002.

[97] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *World Wide Web*, pages 661–670, 2010.

[98] D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

[99] D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Canada, 2008.

[100] D. Lizotte, R. Greiner, and D. Schuurmans. An experimental methodology for response surface optimization methods. *J. of Global Optimization*, pages 1–38, 2011.

[101] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with Gaussian process regression. In *Proc. of IJCAI*, pages 944–949, 2007.

[102] M. Locatelli. Bayesian algorithms for one-dimensional global optimization. *J. Global Optimization*, 1997.

[103] M. Lzaro-gredilla and M. K. Titsias. Variational heteroscedastic Gaussian process regression. In *International Conference on Machine Learning*, pages 841–848. ACM, 2011.

[104] O. Madani, D. Lizotte, and R. Greiner. Active model selection. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2004.

[105] N. Mahendran, Z. Wang, F. Hamze, and N. de Freitas. Adaptive MCMC with Bayesian optimization. *Journal of Machine Learning Research - Proceedings Track*, 22:751–760, 2012.

[106] R. Marchant and F. Ramos. Bayesian optimisation for intelligent environmental monitoring. In *NIPS workshop on Bayesian Optimization and Decision Making*, 2012.

[107] O. Maron and A. W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. *Robotics Institute*, page 263, 1993.

[108] R. Martinez–Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103, 2009.

[109] J. Martinez, J. J. Little, and N. de Freitas. Bayesian optimization with an empirical hardness model for approximate nearest neighbour search. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[110] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos. Active policy learning for robot planning and exploration under uncertainty. *Robotics Science and Systems*, 2007.

[111] G. Matheron. The theory of regionalized variables and its applications. *Cahier du Centre de Morphologie Mathematique, Ecoles des Mines*, 1971.

[112] B. C. May, N. Korda, A. Lee, and D. S. Leslie. Optimistic Bayesian sampling in contextual bandit problems. Technical Report 11:01, Statistics Group, School of Mathematics, University of Bristol, 2011.

[113] V. Mnih, C. Szepesvári, and J.-Y. Audibert. Empirical Bernstein stopping. In *International Conference on Machine Learning*, pages 672–679. ACM, 2008.

[114] J. Močkus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *J. of Global Optimization*, 4(4):347–365, 1994.

[115] J. Močkus, V. Tiesis, and A. Žilinskas. The application of Bayesian methods for seeking the extremum. In L. Dixon and G. Szego, editors, *Toward Global Optimization*, volume 2. Elsevier, 1978.

[116] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems*, pages 783–791, 2011.

[117] R. Munos. From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning. Technical Report hal-00747575, INRIA Lille, 2014.

[118] R. M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995.

[119] J. Nelder and R. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society, Series A*, 135(3):370–384, 1972.

[120] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimisation. In *LION*, 2009.

[121] C. Paciorek and M. Schervish. Nonstationary covariance functions for gaussian process regression. In *Advances in Neural Information Processing Systems*, volume 16, pages 273–280, 2004.

[122] V. Picheny and D. Ginsbourger. A nonstationary space-time Gaussian process model for partially converged simulations. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):57–78, 2013.

[123] J. C. Pinheiro and D. M. Bates. Unconstrained parametrizations for variance-covariance matrices. *Statistics and Computing*, 6(3):289–296, 1996.

[124] J. Quiñonero-Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

[125] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

[126] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[127] D. Russo and B. Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[128] J. Sacks, W. J. Welch, T. J. Welch, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[129] P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, 1992.

[130] T. J. Santner, B. Williams, and W. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

[131] M. J. Sasena. *Flexibility and Efficiency Enhancement for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, 2002.

[132] A. M. Schmidt and A. O'Hagan. Bayesian inference for nonstationary spatial covariance structures via spatial deformations. *Journal of the Royal Statistical Society, Series B*, 65:743–758, 2003.

[133] M. Schonlau. *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 1997.

[134] M. Schonlau, W. J. Welch, and D. R. Jones. Global versus local search in constrained optimization of computer models. *Lecture Notes-Monograph Series*, 34:11–25, 1998.

[135] S. L. Scott. A modern Bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[136] M. Seeger, Y.-W. Teh, and M. I. Jordan. Semiparametric latent factor models. In *AI and Statistics*, 2005.

[137] M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse Gaussian process regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318, 2003.

[138] A. Shah, A. G. Wilson, and Z. Ghahramani. Student-t processes as alternatives to Gaussian processes. In *AI and Statistics*, pages 877–885, 2014.

[139] B. Shahriari, Z. Wang, M. W. Hoffman, A. Bouchard-Côté, and N. de Freitas. An entropy search portfolio. In *NIPS workshop on Bayesian Optimization*, 2014.

[140] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2005.

[141] E. Snelson, C. E. Rasmussen, and Z. Ghahramani. Warped Gaussian processes. In *Advances in Neural Information Processing Systems*, 2003.

[142] J. Snoek. *Bayesian Optimization and Semiparametric Models with Applications to Assistive Technology*. PhD thesis, University of Toronto, Toronto, Canada, 2013.

[143] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[144] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, Prabhat, and R. Adams. Scalable Bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

[145] J. Snoek, K. Swersky, R. S. Zemel, and R. P. Adams. Input warping for Bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014.

[146] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, pages 1015–1022, 2010.

[147] K. Swersky, D. Duvenaud, J. Snoek, F. Hutter, and M. A. Osborne. Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011*, 2014.

[148] K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 2004–2012, 2013.

[149] K. Swersky, J. Snoek, and R. P. Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

[150] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[151] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Knowledge Discovery and Data Mining*, pages 847–855, 2013.

[152] M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

[153] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause. Explore-exploit in top-N recommender systems via Gaussian processes. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 225–232. ACM, 2014.

[154] E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *J. of Statistical Planning and Inference*, 140:3088–3095, 2010.

[155] E. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, 140(11):3088–3095, 2010.

[156] J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *J. of Global Optimization*, 44(4):509–534, 2009.

[157] Z. Wang and N. de Freitas. Theoretical analysis of Bayesian optimisation with unknown Gaussian process hyper-parameters. *arXiv preprint arXiv:1406.7758*, 2014.

[158] Z. Wang, B. Shakibi, L. Jin, and N. de Freitas. Bayesian multi-scale optimistic optimization. In *AI and Statistics*, pages 1005–1014, 2014.

[159] Z. Wang, M. Zoghi, D. Matheson, F. Hutter, and N. de Freitas. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conference on Artificial Intelligence*, pages 1778–1784, 2013.

[160] B. J. Williams, T. J. Santner, and W. I. Notz. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, 10:1133–1152, 2000.

[161] D. Yogatama and G. Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *AI and Statistics*, pages 1077–1085, 2014.

[162] D. Yogatama and N. A. Smith. Bayesian optimization of text representations. *arXiv preprint arXiv:1503.00693*, 2015.

[163] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee. Improving object detection with deep convolutional networks via Bayesian optimization and structured prediction. In *IEEE Computer Vision and Pattern Recognition Conference*, 2015.

[164] A. Žilinskas and J. Žilinskas. Global optimization based on a statistical model and simplical partitioning. *Computers and Mathematics with Applications*, 44:957–967, 2002.