Continuous Multi-Task Bayesian Optimisation with Correlation

Michael Pearce[a], Juergen Branke[b,*]

[a] *Complexity Science, University of Warwick, Coventry CV4 7AL, UK*
[b] *Warwick Business School, University of Warwick, Coventry CV4 7AL, UK*

Abstract

This paper considers the problem of simultaneously identifying the optima for a (continuous or discrete) set of correlated tasks, where the performance of a particular input parameter on a particular task can only be estimated from (potentially noisy) samples. This has many applications, for example, identifying a stochastic algorithm's optimal parameter settings for various tasks described by continuous feature values. We adapt the framework of Bayesian Optimisation to this problem. We propose a general multi-task optimisation framework and two myopic sampling procedures that determine task and parameter values for sampling, in order to efficiently find the best parameter setting for all tasks simultaneously. We show experimentally that our methods are much more efficient than collecting information randomly, and also more efficient than two other Bayesian multi-task optimisation algorithms from the literature.

Keywords: Heuristics, Parameter tuning, Multi-task optimisation, Global optimisation

1. Introduction

Optimisation is the task of finding the input of a function that maximises the output. We consider the problem where one is given multiple optimisation problems on the same domain where the optima of one problem can teach us about the optima of another. Given a set of tasks that can be described by continuous features, and a tool with continuous parameters to be tuned, we would like to identify the unique optimal parameter setting for each task. We can perform experiments to collect information, i.e., run the tool with a specific parameter setting on a specific task, and obtain a sample of a noisy performance measurement. Given a finite budget, the goal is to decide

*Corresponding author
Email addresses: m.a.l.pearce@warwick.ac.uk (Michael Pearce), juergen.branke@wbs.ac.uk (Juergen Branke)

which sequence of experiments to perform that would allow us to construct the best mapping from task features to optimal parameter settings.

This general problem occurs in many applications, including

- *Tuning of Optimisation Algorithms.* Many optimisation algorithms have parameters that need to be tuned specific to the problem instance at hand. It is thus desirable to construct a mapping that suggests the best parameter setting depending on features of the problem instance. In the machine learning community, training of deep neural networks requires setting the learning rate for gradient descent depending on the dataset and model. In the meta-learning community, much work has been done on deriving a mapping from problem features to best algorithm parameters based on a given set of performance data (e.g. Smith-Miles, 2008). Our method can be used to decide which experiments should be conducted to identify the best mapping given a budget of experiments.

- *Dose-Finding Clinical Trials.* The dosage and the compound mixture of a drug may have a strong impact on a drug's effect, and different patients react differently to the drug. Our method can be used to design more efficient clinical trials to identify the best dosage or compound mixture for each patient, based on patient characteristics such as age or biomarker response. A related application has been considered by Krause and Ong (2011).

- *Online Advertising.* In online advertisement, it is easy to deploy several different advertisements and advertisement formats (banner, video, etc.) simultaneously, and pick for each viewer the advertisement that one believes results in the highest return (in terms of click-through rate or money spent). Often, some information is available on the viewers, such as search terms, websites visited or order history. Our method could be used to identify which advertisement would be most effective for each viewer, based on some viewer characteristics.

- *Online Operating Policies.* A complex system may best be operated in different ways depending on the context such as environmental conditions. For example, a factory may be using dispatching rules for real-time scheduling, and the dispatching rules have some parameters whose optimal setting depends on shop floor conditions such as utilisation level or product mix. Heger et al. (2016) use a large number of experiments to derive a policy for setting dispatching rule parameters depending on shop floor conditions. Our method can be used

to identify suitable experiments that will allow constructing an appropriate mapping more efficiently.

In this paper we propose two myopic sequential sampling methods for collecting performance measurements of task-parameter pairs such that one can construct a mapping that maximises the total expected performance across all tasks. Our method uses Gaussian Processes to model the unknown function from task features and parameters to performance, and thus exploits covariance in the space of tasks and parameters. It iteratively suggests the next sample where it suspects the highest immediate information gain. We demonstrate empirically that our methods outperform two state-of-the-art Bayesian multi-task optimisation algorithms from the literature. Furthermore, to the best of our knowledge, we are also the first to explicitly consider two different ways for a decision maker to pick a solution: the solution with the best *sampled* performance, and the solution with the best *predicted* (but not necessarily sampled) performance.

The paper is structured as follows. In Section 2 we provide an overview of similar problems and solutions considered in the literature. In Section 3 we provide the general problem framework, and then describe the Gaussian Process model and our two new sampling procedures in Section 4. A benchmark algorithm is described in Section 5. In Section 6 we empirically evaluate our methods on three synthetic benchmarks and we conclude in Section 7 with a summary and some ideas for future work.

2. Literature Review

In this paper, we look at continuous multi-task Bayesian Optimisation, where the goal is to simultaneously identify the global optimum for each task when there is a continuous distribution of correlated tasks.

When trying to find the global optimum of an expensive to evaluate function without information on the gradient, Bayesian Optimisation based on Gaussian Processes, or Kriging models, have gained much attention. These methods build a surrogate model of the objective function based on a few initial samples and then use an acquisition function (sometimes called infill criterion) to iteratively decide where to sample next to improve the model and find better solutions. The most popular Bayesian Optimisation algorithm is probably the Efficient Global Optimisation

(EGO) algorithm of Jones et al. (1998) that combines a Gaussian Process to interpolate an expensive function and an expected improvement criterion for deciding where to sample next. The SKO algorithm (Huang et al., 2006b) extends the EGO algorithm by proposing an acquisition function that also accounts for noise in function evaluations. Especially in the area of multi-armed bandits, UCB-type algorithms are often used (Srinivas et al., 2010).

The Knowledge Gradient for Correlated Beliefs (Frazier et al., 2009) is a myopic sampling policy that aims to maximise the new predicted performance after one sample. It can be applied when using a Gaussian Process with a discretised input, has a theoretical basis in dynamic programming and provides myopic and asymptotic guarantees. The Knowledge Gradient policy for Continuous Parameters (Scott et al., 2011) generalises the EGO algorithm to noisy functions. Perhaps the most interesting difference between the Knowledge Gradient policy and previous policies is that the Knowledge Gradient accounts for covariance when judging the value of a sample, i.e., the expected improvement takes into account the possibility that as a result of the new sample, the predicted performance at other previously sampled points may change. Other popular Bayesian optimisation variants that take into account covariance include Stepwise Uncertainty Reduction (Villemonteix et al., 2009; Chevalier et al., 2014), Predictive Entropy Search (Hernandez-Lobato et al., 2014), and the Integrated Expected Conditional Improvement (IECI) (Gramacy and Lee, 2011). An extensive overview and empirical comparison of various infill criteria on a range of noisy problems has been provided by Picheny et al. (2013b).

The use of multi-task Gaussian Process for optimisation is widely used in multi-fidelity optimisation. One exploits information from a secondary low fidelity model that can be sampled more cheaply in order to optimise an expensive high fidelity model, and multitask learning/co-kriging can be used to integrate information from several models. Forrester et al. (2007) provide a concise example and Huang et al. (2006a) extend the SKO algorithm to multi-fidelity setting where a fidelity level and input parameters are to be selected for each sample. Swersky et al. (2013) use an acquisition function based on entropy search whilst accounting for the cost of a new sample and Poloczek et al. (2017) propose a similar model with the Knowledge Gradient used to measure improvement in the high fidelity model. Picheny et al. (2013a) considers an interesting case where the precision of an evaluation can be chosen, at the expense of higher computational cost for higher precision.

A few papers extend the idea of Bayesian Optimisation to the case where several related optimisation problems have to be solved sequentially. Morales-Enciso and Branke (2015) consider the optimisation of a changing objective function with EGO, and propose three different ways to re-use information gathered on previous objective functions to speed up optimisation of the current objective function: using the old posterior mean as the new prior mean, treating old samples as noisy, and treating time as an extra input dimension to the learnt function. The latter idea is also used by Poloczek et al. (2016) to warm-start Bayesian Optimisation, however with the Knowledge Gradient as the acquisition function for the current optimisation task. A similar problem has been much studied in the field of Contextual multi-armed bandits, an extension of the multi-armed bandit problem where the best arm depends on a context that is randomly changing with time. Such models are often viewed as idealised models for reinforcement learning where one must learn the optimal action for every state. In such a case, sampling policies must aim to maximise cumulative reward, see Zhou (2015) for a survey. Krause and Ong (2011) use Gaussian Processes with inputs that are both decision variables and context variables and propose an upper confidence bound acquisition function to find the best arm for each context.

Related to contextual multi-armed bandits is the field of contextual policy search, which tries to identify the best parameters of a lower level policy depending on the context, see Deisenroth et al. (2013) for a survey. Most papers in this area assume the context is a consequence of previous actions, however Metzen (2015) considers a setting where the context is free to be chosen by the learner during optimisation, and the goal is to find parameters for a policy to optimise rewards over all contexts in which the policy will be used. For this closely related problem, Metzen (2015) proposes an adaptation of entropy search that also exploits the correlation between contexts, and applies it to a robotic control task.

The specific goal we consider in this work is to identify the best solution for multiple optimisation problems simultaneously. Different from the approaches to contextual multi-armed bandits and most reinforcement learning applications, the "context" (or state or optimisation problem) is not changing externally, but is within the control of the experimenter during sampling. An example would be when tuning an algorithm for different problem instances, it is up to the experimenter to decide which algorithm parameters to test on which problem instance. Pearce and Branke (2016) consider the problem of learning the best of a finite set of tools for each of a finite set of

tasks, with correlation across tasks but independence between tools. They provide an acquisition function that accounts for the effect a new sample will have on the performance prediction of all the tasks. Hu and Ludkovski (2015) considered the same problem with continuous tasks. They use independent Gaussian Processes to approximate each tool's performance over the task space and propose an acquisition function based on the expectation of the maxima of all functions, however do not account for correlation across task space. Bardenet et al. (2013) consider the problem of finding the best hyper-parameters for training deep belief networks where the best parameter settings vary with features of the dataset. They use a Gaussian Process from the joint space (dataset features, algorithm parameters) to algorithm performance. For data acquisition, the proposed SCoT algorithm visits the datasets in turn and samples the hyper-parameter setting that maximises EGO expected improvement on that dataset. Finally, Ginsbourger et al. (2014) solve the same problem of optimising parameters conditional on a task for a range of tasks. When estimating the improvement resulting from sampling a particular parameter on a particular task, they use an EGO-like approach and consider the improvement that this point may have over the current best predicted parameter on the same task. Since this algorithm is closely related to our methods, it will be described in more detail in Section 5, and is used as benchmark in our empirical comparison in Section 6

Given the success of multi-fidelity methods and contextual optimisation methods, when doing multi-task optimisation, exploiting covariance across tasks can provide significant performance gains. Unlike multi-fidelity models that aim to optimise a single expensive function, or contextual multi-armed bandit problems or time dependent problems, we consider the problem of directly optimising multiple tasks simultaneously, where there is a distribution of tasks. To the best of our knowledge, our approach is the first that accounts for covariance across tasks when deciding where to allocate the next sample and works on continuous domains. We look at a generalised multi-task simultaneous optimisation problem and propose two novel acquisition functions that myopically aim to maximise the predicted improvement at the next step.

3. Problem Formulation

We assume that there exists a (discrete or continuous) set of tasks described by D_X features, $x \in X$, and the tasks are distributed according to a known density $\mathbb{P}[x]$. There is a tool with

D_A tuneable parameters $a \in A$. Executing the tool with parameters a on a task with features x yields a performance measurement $Y_{x,a} = \theta(x, a) + \epsilon$ where $\theta : X \times A \to \mathbb{R}$ is a deterministic latent function from (task, parameter) to expected performance and ϵ is independent and identically distributed observation noise $\epsilon \sim N(0, \sigma_\epsilon^2)$. Our aim is to find a mapping $S : X \to A$ from a given task to the optimal parameter setting for this task that approximates the true optimal mapping $S^*(x) = \underset{a}{\text{argmax}}\ \theta(x, a)$ with incomplete information. The quality of our derived mapping $S(x)$ at the end of sampling is the corresponding true expected performance over all tasks:

$$\int_{x \in X} \theta(x, S(x)) \mathbb{P}[x] dx. \tag{1}$$

We assume we have a fixed budget of N samples (tests of a parameter setting on a task), and that we can sample iteratively, i.e., we can select the task x and the parameter setting a from which to sample performance $Y_{x,a}$ based on the information collected so far.

Given this formulation, if x is constant, the problem reduces to a single global optimisation over A. However, because there is a range of x, one must find the global optimal $\max_a \theta(x, a)$ for each x. Examples of benchmark functions used later can be seen in Figure **??**.

Our formulation also accounts for task distribution because in practice, under a constrained budget, finding the optimal parameters for tasks with unusual outlying features (low $\mathbb{P}[x]$) is less useful than finding the optimal parameters for common tasks (high $\mathbb{P}[x]$). We use the notation of $\mathbb{P}[x]$ to describe task density, however we do not assume x is random but simply that the density is normalised to integrate to 1. Note that one may want to weight the tasks such that some types of tasks have higher priority than others. In this framework, $\mathbb{P}[x]$ can be used to describe task importance as well as or instead of task distribution. Note that if the set of tasks is finite, the integral in Equation 1 is replaced with a summation and $\mathbb{P}[x]$ is a probability mass function.

In the clinical trials dose finding example given earlier, x may be a patient's cancer cell biomarker measurements, $\mathbb{P}[x]$ would be the patient population distribution across biomarker space X, and a would be the quantities of compounds to use in a treatment. $\theta(x, a)$ would be the expected measured outcome of the treatment such as average reduction in tumour size. A clinician would then like to find the compound mixture a for each patient x that maximises the expected tumour reduction $\theta(x, a)$ across all patients $\mathbb{P}[x]$. The clinician can then create a mapping from a given patient x to the optimal compounds $a = S(x)$.

In the earlier example of tuning an optimisation algorithm, x would be the features of a problem

instance, $\mathbb{P}[x]$ would be the distribution of problem instances one expects to encounter across feature space X, and a would be the parameters of the optimisation algorithm. $\theta(x, a)$ would be a measured performance metric of the algorithm when run with parameters a on problem instance x. For a given problem instance x, a user would like to know the best parameters $a = S(x)$ in order to maximise the expected performance $\theta(x, a)$.

Using this framework, we consider two ways to derive a mapping $S(x)$. In general, for a risk neutral decision maker, $S(x)$ would return the solution with the best *predicted* performance based on the samples collected and the derived performance prediction model $\mu(x, a)$, such a mapping is given by

$$S^1(x) = \underset{a}{\operatorname{argmax}}\ \mu(x, a). \tag{2}$$

As a special case of the above framework, when the task distribution is discrete, the sampling budget is greater than the number of tasks, and there is no observation noise ($\sigma_\epsilon = 0$), one may also consider a risk averse decision maker. In this paper, we define such a decision maker as one that only selects parameters a for a task x_i from parameters that have been measured, i.e., of which the performance is known with certainty. If the set of sampled parameters is denoted $A^N = \{a^1, ..., a^N\}$ and the subset corresponding to task x_i is denoted $A_i^N = \{a^n | x^n = x_i, n \in \{1, ..., N\}\}$, a mapping for such a decision maker is then given by

$$S^2(x_i) = \underset{a \in A_i^N}{\operatorname{argmax}}\ \theta(x_i, a). \tag{3}$$

where we have noted that the true objective function is known for past evaluated points $\{y^1, ..., y^n\} = \{\theta(x^1, a^1), ..., \theta(x^n, a^n)\}$ as there is no observation noise.

4. Myopic Sampling Methods

We propose to use a Gaussian Process regression model to predict the underlying latent function $\theta(x, a)$ based on the n data points collected so far $(x^1, a^1, y^1), ..., (x^n, a^n, y^n)$. Gaussian Process regression, also known as kriging, has been used successfully for Bayesian Optimisation due to its simplicity and the possibility to derive closed form solutions for the posterior distribution over possible function values. Given closed form posterior distributions, calculating expected incremental improvements in various metrics given a sample can also be done analytically.

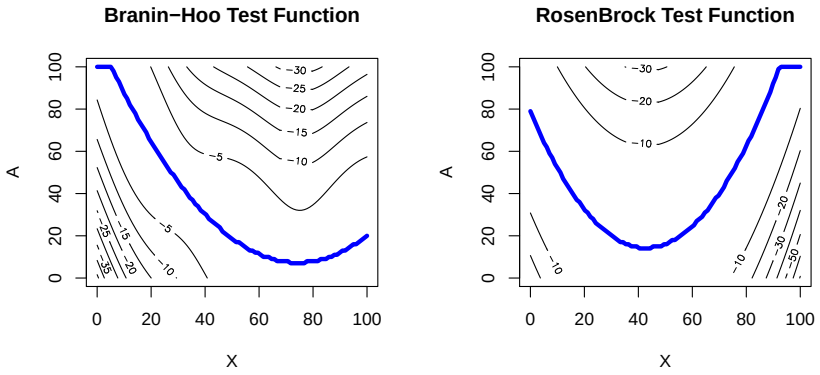


Figure 1: Given functions of multiple inputs, we aim to find the optimum of some inputs conditioned on the remaining inputs. In 2D, for each point along the horizontal axis we aim to find the optimal value along the vertical axis as shown by the thick blue lines.

For a given input (x, a), Gaussian Process regression treats the value of the unknown latent function $\theta(x, a)$ as a Gaussian random variable whose mean and variance are determined by the expected similarity between the response at the given input and the responses for evaluated inputs. Similarity between responses is determined by the kernel function and encodes properties such as smoothness.

For this section we shall use the notation $\tilde{x} = (x, a) \in X \times A$ and write functions of both input variables as $\mu^n(x, a) = \mu^n(\tilde{x})$. We define the set of sampled task features values $\{x^1, ..., x^n\} = X^n$, parameters $\{a^1, ..., a^n\} = A^n$, the set of input pairs $\{(x^1, a^1), ..., (x^n, a^n)\} = \tilde{X}^n$ and column vector of responses $(y^1, ..., y^n) = Y^n$. We define the sequence of filtrations, $\mathcal{F}^n$, as sigma algebras generated by the data collected up to time n, $\mathcal{F}^n = \sigma\{(x^1, a^1, y^1), \ldots, (x^n, a^n, y^n)\}$. Given prior mean and covariance functions $\mu^0(x, a)$, $k^0((x, a), (x', a'))$, one may use the Matrix Inversion Lemma (Hager, 1989) to condition the prior functions yielding the posterior mean and covariance functions as follows (Rasmussen and Williams, 2004),

$$\mathbb{E}\left[\theta(x, a)|\mathcal{F}^n\right] = \quad \mu^n(\tilde{x}) = \quad \mu^0(\tilde{x}) + k^0(\tilde{x}, \tilde{X}^n)(K^n + \sigma_\epsilon^2 I)^{-1}(Y^n - \mu^0(\tilde{X}^n)) \tag{4}$$

$$Cov[\theta(x, a), \theta(x', a')|\mathcal{F}^n] = \quad k^n(\tilde{x}, \tilde{x}') = \quad k^0(\tilde{x}, \tilde{x}') - k^0(\tilde{x}, \tilde{X}^n)(K^n + \sigma_\epsilon^2 I)^{-1}k^0(\tilde{X}^n, \tilde{x}') \tag{5}$$

where $K_{ij}^n = k^0((x^i, a^i), (x^j, a^j))$ is the $n \times n$ matrix composed of the prior covariance function evaluated for all the sampled input points $\tilde{X}^n$. We have written $k^0(\tilde{x}, \tilde{X}^n)$ to denote the $1 \times n$ matrix of prior covariance between $\tilde{x}$ and all points in $\tilde{X}^n$ and likewise $k^0(\tilde{X}^n, \tilde{x}')$ is the $n \times 1$ matrix.

As mentioned in Section 3, there are at least two choices of the derived mapping $S(x)$ that are not clearly distinguished in the current literature. The EGO and SKO algorithms were implicitly designed on the basis that the input corresponding to best sampled point will be returned to the user at the end of sampling, which corresponds to the mapping $S^2(x)$ from Equation 3. For the EGO algorithm, samples are allocated to maximise the expected improvement (EI) of the largest sampled value

$$\begin{aligned} EI(a) &= \mathbb{E}\left[max\{y^1, ..., y^n, y^{n+1}\}|\mathcal{F}^n, a^{n+1} = a\right] - \max\{y^1, \ldots, y^n\} \quad (6) \\ &= \mu^n(a)\Phi\left(\frac{\bar{y} - \mu^n(a)}{\delta^n(a)}\right) - \delta^n(a)\phi\left(\frac{\bar{y} - \mu^n(a)}{\delta^n(a)}\right) \quad (7) \end{aligned}$$

where $\Phi(z)$ and $\phi(z)$ are standard normal cumulative and density functions, $\bar{y} = \max\{y^1, ..., y^n\}$ is the current best sampled function value and $\delta^n(a) = \sqrt{k^n(a, a)}$ is the posterior standard deviation.

In the multi-task setting however, $S^2(x)$ and EGO can only be used if the task distribution is discrete and the sampling budget is greater than the number of tasks, therefore cannot be used in the continuous task case (infinite tasks) or whenever the tasks outnumber the sampling budget. Using a regression model, such as Gaussian Process Regression, one is able to predict the performance of any point, and a risk-neutral decision maker is more likely to choose the point with the best *predicted* (but not necessarily sampled) performance. This is the assumption used by Frazier et al. (2009) in their paper on Knowledge Gradient for Correlated Normal Beliefs, although that algorithm was not designed specifically for continuous optimisation. In our case of multi-task optimisation, we would choose the mapping $S^1(x)$ as given by Equation 2, which is defined for any sampled or unsampled task and thus applicable also if the task distribution is continuous. Our methods below will mainly target the mapping S^1, but we will also propose a new way to deal with S^2 in Section 4.3

In order to allocate samples to maximise improvement, we require an estimate of the current performance upon which we must improve, and we note that as with constructing the mapping, it is not possible to use the highest sampled point as this does not exist for all tasks. Instead we use the model's prediction of expected performance. Given the mapping $S^1(x)$, the current predicted performance on a task x is $\mu^n(x, S^1(x)) = \max_a \mu^n(x, a)$ and so the total predicted performance after n samples across the task distribution is given by

$$P^n = \int_{x \in X} \mu^n(x, S^1(x))\mathbb{P}[x]\, dx = \int_{x \in X} \max_a \mu^n(x, a)\mathbb{P}[x]\, dx$$

that is the same as Equation 1 with the true function, $\theta(x, a)$, replaced by the prediction $\mu^n(x, a)$.

Given a measure of performance, we intend to derive myopic sampling policies that maximise the incremental gain, $P^{n+1} - P^n$. Therefore we require an updating formula for the predicted performance after the next sample $\mathbb{E}\left[P^{n+1}|\mathcal{F}^n, \tilde{x}^{n+1}\right]$ which requires the new posterior mean $\mu^{n+1}(x, a)$.

Equation 4 gives the update to the prior mean and covariance functions from 0 samples to $n > 0$ samples. Since the equations derive from a conjugate prior we may use Bayesian updating, a simple change of indices from 0 and n to n and $n + 1$ yields the single step update to the posterior mean

$$\mu^{n+1}(\tilde{x}) = \mu^n(\tilde{x}) + \frac{k^n(\tilde{x}, \tilde{x}^{n+1})\ (y^{n+1} - \mu^n(\tilde{x}^{n+1}))}{k^n(\tilde{x}^{n+1}, \tilde{x}^{n+1}) + \sigma_\epsilon^2}$$

which requires knowing y^{n+1}. Before measuring the new response value y^{n+1}, the posterior predictive distribution of y^{n+1} conditioned on $\mathcal{F}^n$ is given by the uncertainty in $\theta(\tilde{x}^{n+1})$ and the noise, ϵ, therefore

$$y^{n+1} \sim N(\mu^n(\tilde{x}^{n+1}),\ k^n(\tilde{x}^{n+1}, \tilde{x}^{n+1}) + \sigma_\epsilon^2),$$

and the above equation may be factorised as follows

$$\mu^{n+1}(\tilde{x}) = \mu^n(\tilde{x}) + \tilde{\sigma}^n(\tilde{x}; \tilde{x}^{n+1})\ Z^{n+1} \tag{8}$$

where Z^{n+1} is the stochastic z-score of the new response value on the prior predictive distribution and is given by

$$Z^{n+1} = \frac{y^{n+1} - \mu^n(\tilde{x}^{n+1})}{\sqrt{k^n(\tilde{x}^{n+1}, \tilde{x}^{n+1}) + \sigma_\epsilon^2}}$$

which is a standard normal random variable $Z^{n+1} \sim N(0, 1)$ when conditioned on $\mathcal{F}^n$. The remaining term $\tilde{\sigma}^n(\tilde{x}; \tilde{x}^{n+1})$ which is given by

$$\tilde{\sigma}^n(\tilde{x}; \tilde{x}^{n+1}) = \frac{k^n(\tilde{x}, \tilde{x}^{n+1})}{\sqrt{k^n(\tilde{x}^{n+1}, \tilde{x}^{n+1}) + \sigma_\epsilon^2}}$$

may be seen as a deterministic function of $\tilde{x}$ parametrised by $\tilde{x}^{n+1}$ that is the additive update to the posterior mean caused by the new sample at $\tilde{x}^{n+1}$ and scaled by the stochastic Z^{n+1}. With a predictive distribution of the posterior mean after a new sample we can express the expectation, over $Z^{n+1} \sim N(0, 1)$, of the performance prediction after the next sample as

$$\mathbb{E}\left[P^{n+1}|\mathcal{F}^n, \tilde{x}^{n+1}\right] \quad = \quad \int_{x \in X} \mathbb{E}\left[\max_a \{\mu^n(x, a) + \tilde{\sigma}^n((x, a); \tilde{x}^{n+1}) Z^{n+1}\}\right] \mathbb{P}[x]\, dx. \tag{9}$$

Finally, we may write the incremental gain, or expected improvement, the difference in predicted performance between consecutive samples as follows

$$\begin{aligned}\mathcal{I}(\tilde{x}) &= \mathbb{E}\left[P^{n+1} - P^n | \mathcal{F}^n, \tilde{x}^{n+1} = \tilde{x}\right] \\ &= \int_{x' \in X} \mathbb{E}\left[\max_{a'}\{\mu^n(x', a') + \tilde{\sigma}^n((x', a'); \tilde{x}) Z^{n+1}\} - \max_{a'} \mu^n(x', a')\right] \mathbb{P}[x']\, dx'. \end{aligned} \quad (10)$$

The above expression can be evaluated exactly when X and A are finite sets. However, when X and A are infinite continuous sets, there is no solution for the expectation over Z^{n+1} of the max function, nor is it possible to integrate across tasks for arbitrary $\mathbb{P}[x]$. Next, we propose the CLEVI and REVI sampling policies based on approximations to Equation 10.

4.1. CLEVI Sampling Policy

We aim to allocate samples in order to maximise $\mathcal{I}(\tilde{x})$, the expected improvement in predicted performance calculated across all tasks, however this integral must be approximated. The Convolutional Local Expected Value of Improvement (CLEVI) policy makes two assumptions in order to evaluate the integral. Firstly, for each x, the maximisation over continuous A may be approximated by a maximisation over a finite set A_D, and secondly, the improvement at unsampled tasks may be approximated by the improvement at the sampled task and the covariance across tasks. By replacing the maximisation over A with a maximisation over A_D, with $n_A = |A_D|$, the expectation in the integrand of $\mathcal{I}(\tilde{x})$ may be written as:

$$\mathbb{E}\left[\max\{\mu^n(x', A_D) + \tilde{\sigma}^n((x', A_D); (x, a)) Z^{n+1}\} - \max \mu^n(x', A_D)\right] \quad (11)$$

where $\mu^n(x, A_D) = (\mu^n(x, a_1), \mu^n(x, a_2), \dots) \in \mathbb{R}^{n_A}$ is the vector of means and similarly for $\tilde{\sigma}^n((x, A_D); (x, a))$. Gathering terms, Equation 11 is thus of the form

$$\mathbb{E}\left[\max\{\mu_1 + \sigma_1 Z, \dots, \mu_{n_A} + \sigma_{n_A} Z\}\right] \quad (12)$$

which is the normal expectation of the maximum of linear functions. This expectation can be cheaply evaluated using Algorithm 1 in Knowledge Gradient for Correlated Normal Beliefs (Frazier et al., 2009) reproduced here in Algorithm 1. To summarise briefly, which of the linear functions is the largest varies with Z, finding the highest function for each Z and calculating the intersections can be done in $O(n_A log(n_A))$ time. Once the piece-wise linear "ceiling" over

all the linear functions is known, the expectation over Z is a sum of expectations of each linear piece, a sum of means of truncated normal distributions. Graphically, the maximal a of $\mu^{n+1}(x,a) = \mu^n(x,a) + \tilde{\sigma}^n((x,a);\tilde{x}^{n+1})$ Z moves away from the maximal a of $\mu^n(x,a)$ as Z moves away from 0 and Algorithm 1 calculates the expectation of the height of the new maxima over the normally distributed scale Z. Further information can be found in Frazier et al. (2009). For convenience, we define the function $KG : \mathbb{R}^{n_A} \times \mathbb{R}^{n_A} \to \mathbb{R}$ that takes a vector of means (intercepts) and a vector of additive updates (gradients) and returns the expectation as given by Algorithm 1. Given this function we may write the integral as

$$\mathcal{I}(\tilde{x}) \approx \int_X KG\big(\,\mu^n(x', A_D),\, \tilde{\sigma}^n((x', A_D);(x,a))\,\big)\mathbb{P}[x']dx'. \tag{13}$$

The second assumption we make is that the expected improvement at an unsampled task $x' \neq x$ may be approximated by the improvement at the sampled task and the covariance between tasks:

$$\mathrm{KG}\big(\,\mu^n(x', A_D),\, \tilde{\sigma}^n((x', A_D);(x,a))\,\big) \approx \mathrm{KG}\big(\,\mu^n(x, A_D),\, \tilde{\sigma}^n((x, A_D);(x,a))\,\big)\, k_X(x,x') \tag{14}$$

where $k_X(x,x')$ comes from the Gaussian Process model and we assume that the kernel is factorisable $k^0((x,a),(x',a')) = \sigma_0^2 k_X(x,x')k_A(a,a')$ which is true for the Matern and exponential classes of kernels. This assumption means that the KG function need only be called once. Plugging both of these assumptions into Equation 10 and rearranging yields the following formula

$$\mathcal{I}(\tilde{x}) \approx\; KG\big(\,\mu^n(x, A_D),\, \tilde{\sigma}^n((x, A_D);(x,a))\,\big) \int_X k_X(x,x')\mathbb{P}[x']dx'. \tag{15}$$

where the integral of the right hand side is the convolution between the task covariance and the underlying task distribution which may be found analytically in special cases such as a Gaussian kernel and either a uniform distribution, a triangular distribution, Gaussian distribution or even sums of these distributions by the linearity of the convolution operator. If no analytical expression can be found, then Monte-Carlo integration may be used replacing the true distribution with a kernel density estimate from samples of $P[x]$ where the kernel is the same $k_X(x,x')$ as the Gaussian process. Given these two assumptions, the CLEVI acquisition function may be written as

$$\mathrm{CLEVI}(x,a) \;\;=\;\; KG\big(\,\mu^n(x, A_D),\, \tilde{\sigma}^n((x, A_D);(x,a))\,\big)\tilde{P}[x] \tag{16}$$

where $\tilde{P}[x]$ is simply the task distribution with the convolution applied. New samples are allocated to the (x,a) that maximise this CLEVI acquisition function. The CLEVI sampling policy treats

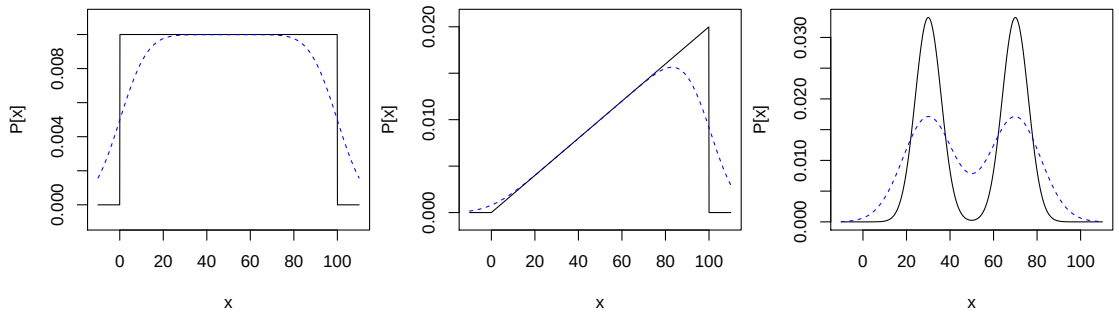


Figure 2: In all plots, black solid line: the task distribution $P[x]$, blue dotted line: transformed task distribution $\tilde{P}[x]$ with a Gaussian with length scale $l_X = 10$. Using the transformed $\tilde{\mathbb{P}}[x]$ instead of $\mathbb{P}[x]$ down weighs sampling at sharp boundaries as shown in the left plot, and up weighs sampling between clusters as shown in the right plot.

each task as a single global optimisation and applies the Knowledge Gradient Policy to evaluate the value of a new sample on that task alone. However this improvement is weighted by $\tilde{\mathbb{P}}[x]$ which gives the task a relative importance. If the untransformed $\mathbb{P}[x]$ were used instead, the policy may allocate samples to where the single point task density, $\mathbb{P}[x]$, is high. However this may not be ideal in certain cases. For example, given a uniform task distribution and a squared exponential kernel, allocating a sample to the boundary of the task space will only affect other tasks that are near the boundary, whereas a sample allocated to a task that is slightly away from the boundary will influence tasks on the boundary and also deeper tasks. Further examples are given in Figure 2.

Next we discuss the choice of discretisation A_D. For single-task optimisation, the Knowledge Gradient for Continuous Parameters (Scott et al., 2011) also calculates an estimate of the expected improvement by discretising over the decision variable, A. They propose to discretise A using past evaluated points and the current point $A_D = A^n \cup \{a^{n+1}\}$ therefore $n_A = n + 1$. This has the advantage that points will cluster around the global maxima, and if there is no observation noise the output of Algorithm 1 reduces to the popular EI function as used in the EGO algorithm. However, we would like to use the same set of points A_D for every task value $x \in X$ for which the optimal a may be different. Therefore we use a uniform latin hypercube instead of past observations specifically in order to avoid clustering. We use the same A_D for all tasks so that the CLEVI function is a smooth deterministic function that is easily optimised and like the Knowledge Gradient policy, the derivative of $CLEVI(x, a)$ with respect to (x, a) can be derived in closed form and used with multiple starts in gradient ascent optimisers. When choosing n_A, it is necessary to have enough reference points in order to estimate changes in the peak posterior mean for any given task x^{n+1}.

Since the total n samples are spread out over all $X \times A$ and only a subset of samples influence task x^{n+1}, by setting $n_A = n + 1$, the $n + 1$ reference points, which are confined to $x^{n+1} \times A$, are thus denser on the task than the relevant samples and therefore are sufficient to capture changes in the posterior mean across the range of tasks.

A single evaluation of the EGO expected improvement (EI) function requires evaluating both the posterior mean $O(n)$ and covariance $O(n^2)$. Evaluating the CLEVI function requires evaluating the posterior mean and covariance n_A times, and one call to the KG function which has complexity $O(n_A log(n_A))$, thus the complexity of one CLEVI call is $O(n_A n^2 + n_A log(n_A))$, strictly greater than one EI call. In this problem formulation, the expected improvement can only be measured using the posterior mean which is changed by the sample, therefore requiring extra calls to the posterior covariance. This complexity may be easily reduced by assuming that points in A_D that have low correlation with a^{n+1} will have zero posterior correlation, reducing computation with little loss of efficiency. This "zeroing" of posterior covariance is discussed in Section 4.4.

The CLEVI sampling policy is readily adapted to the case where the set of tasks is finite and each task has an associated probability. The convolution is simply a summation, and the CLEVI function can be optimised over A for each task individually. Likewise, if A is finite, one may set $A_D = A$, and continuously optimise over x. In the special case where there are finite tasks, finite parameters and only correlation across tasks, the KG function reduces to the EI function and the CLEVI policy is equivalent to the NEVI policy of Pearce and Branke (2016).

4.2. REVI Sampling Policy

The Regional Expected Value of Improvement policy (REVI) improves upon the CLEVI policy by not making the second assumption and instead accounting for improvement of similar tasks by evaluation rather than approximation, however this requires greater computation. Equation 9 gives the expected improvement across the entire distribution of tasks $\mathbb{P}[x]$. However this cannot be evaluated exactly if X and A are continuous sets, therefore we must approximate the expression by discretising over X and A. By discretising over A as described above in the CLEVI policy, for a given x the expectation over y^{n+1} and Z^{n+1} can be computed exactly using the KG function. We discretise over X by replacing it with a finite set X_{MC} of n_X task feature values distributed according to $\mathbb{P}[x]$. We replace a calculation of the expected improvement over the whole distribution of X with the expected improvement over n_X randomly generated tasks X_{MC}, thus converting the

continuous integral into a Monte-Carlo integral. We define the Monte-Carlo estimate of Equation 10 as $\hat{\mathcal{I}}(x,a) \approx \mathcal{I}(x,a)$ given by

$$\hat{\mathcal{I}}(x,a) = \frac{1}{n_X} \sum_{x_i \in X_{MC}} \mathbb{E}\left[\max_{a_j \in A_D} \{\mu^n(x_i,a_j) + \tilde{\sigma}^n((x_i,a_j);(x,a))Z^{n+1}\} - \max_{a_j \in A_D} \mu^n(x_i,a_j) \right] \quad (17)$$

that tends to the true value $\mathcal{I}(x,a)$ as n_X and n_A tend to infinity. In the summation in Equation 17, each term is the expectation over the maximum of linear functions of Z^{n+1}, therefore each term may be calculated using the KG function and the summation yields the REVI acquisition function

$$REVI(x,a) = \frac{1}{n_X} \sum_{x_i \in X_{MC}} KG\big(\, \mu^n(x_i, A_D),\, \tilde{\sigma}^n((x_i,A_D);(x,a))\,\big). \quad (18)$$

The above Monte-Carlo integral does not usually include the proposed sample task x^{n+1}, because x^{n+1} is not a sample from $\mathbb{P}[x]$. It may be included by adding $CLEVI(x,a)$ such that REVI is a mix of two estimates of $\mathcal{I}(\tilde{x})$ although we do not consider such approximations here. The (task, parameter) pair that maximises the REVI function is chosen for sampling. At each time step, the random sets X_{MC} and A_D are generated and held constant until the next time step when they are regenerated. Jittering the discretisation in both domains ensures the learnt mapping does not overfit to one particular discretisation and in the long term the learnt mapping converges to the true optimal mapping. We discuss more efficient choice of n_X and X_{MC} in Section 4.4.

One call to REVI requires the computation of $\hat{\mathcal{I}}(x^{n+1}, a^{n+1})$ which can be decomposed. There are $n_X n_A$ fixed points in the discretisation. For $n_X(n_A - 1)$ of the points that do not vary with $(x,a)^{n+1}$, the posterior means and final two terms of the posterior covariance can be precomputed and stored between REVI calls. Therefore, only the first two terms of the matrix multiplication for the posterior covariance are necessary, resulting in an $O(n_X n_A n)$ computation per call. The remaining posterior means and covariances for the n_X points corresponding to (X_{MC}, a^{n+1}) must be computed resulting in a cost of $O(n_X n^2)$ and the KG function must be called n_X times. Overall, one call to REVI requires $O(n_X n^2 + n_X n_A n + n_X n_A log(n_A))$. Assuming $n_A = n$ and in our experiments we set $n_X = 4\sqrt{n}$, each REVI call has leading order complexity $O(n^{2.5})$ which is greater than one call to EI, $O(n^2)$, however less than the $O(n^3)$ required to fit a Gaussian Process. Much of the computation may be reduced by assuming points in the discretisation that are uncorrelated with the new sample may be set to 0 which is discussed in Section 4.4.

Algorithm 1 The KG Function The following algorithm takes a vector of intercepts and gradients finds the piece-wise linear epigraph, or "ceiling", of all the overlapping linear functions and calculates the expectation over a normally distributed input Z. $\tilde{Z}$ is the vector of Z values at the vertices of the epigraph, I is the vector of indices of the corresponding linear functions that are part of the epigraph. The algorithm starts with an epigraph of two lines with the lowest gradients, and at each step adds a steeper line and updates the epigraph. All vector indices to start from 1.

Require: $\underline{\mu}, \underline{\sigma} \in \mathbb{R}^{n_A}$

Remove dominated pairs from $\underline{\mu}$ and $\underline{\sigma}$

Sort the elements of $\underline{\mu}$ and $\underline{\sigma}$ in order of increasing $\underline{\sigma}$

Initialize $\underline{\mu} \leftarrow \underline{\mu} - \max\{\underline{\mu}\}$, $I \leftarrow [1, 2]$, $\tilde{Z} \leftarrow [-\infty, \frac{\mu_1 - \mu_2}{\sigma_2 - \sigma_1}]$

for $i = 3$ **to** $|\underline{\mu}|$ **do**

(1) $j \leftarrow last(I)$, $z \leftarrow \frac{\mu_i - \mu_j}{\sigma_j - \sigma_i}$

if $z < last(\tilde{Z})$ **then** Delete last element of I and last element of $\tilde{Z}$, return to (1)

Add i to end of I and z to end of $\tilde{Z}$

end for

$\tilde{Z} \leftarrow [\tilde{Z}, \infty]$

return $\sum_{i=1}^{length(I)} \mu_{I_i}(\Phi(\tilde{Z}_{i+1}) - \Phi(\tilde{Z}_i)) + \sigma_{I_i}(\phi(\tilde{Z}_i) - \phi(\tilde{Z}_{i+1}))$

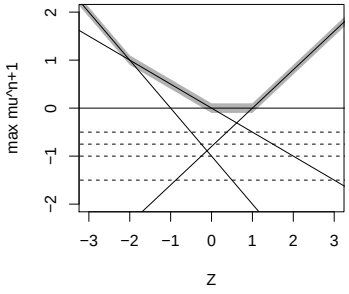


Figure 3: Each line represents the posterior mean at a given point after a new sample y^{n+1} with z-score given by Z. Algorithm 1 removes the dominated functions (dotted), finds the epigraph (highlighted), and calculates the expectation of the epigraph over $Z \sim N(0, 1)$.

4.3. CLEVI and REVI Sampling Policy for Discrete Task Distributions

Adapting the CLEVI policy to discrete tasks or discrete parameters is easily done as explained in Section 4.1. Adapting the REVI policy when the set X is finite and $\mathbb{P}[x]$ is a probability mass

function, the expected improvement over all tasks can be computed and weighted according to the relative probabilities of each task, the new acquisition function becomes

$$\text{REVI}_D(x_j, a) \quad = \quad \sum_{x_i \in X} \mathbb{P}[x_i]\, KG\big(\, \mu^n(x_i, A_D),\, \tilde{\sigma}^n((x_i, A_D), (x_j, a))\,\big) \tag{19}$$

which is equivalent to taking the limit $\lim_{n_x \to \infty} \text{REVI}(x_j, a)$. As the sample size grows, the ratios of tasks in the Monte-Carlo approximation $\hat{\mathcal{I}}(x_j, a)$ tend to the true probabilities. If we further assume that the parameter space is discrete with no correlation between parameters, the KG function reduces to the EI function and $\text{REVI}_D(x, a)$ is equivalent to the REVI acquisition function of Pearce and Branke (2016) which is proven myopically and asymptotically optimal in this special case.

In the case of a risk averse decision maker who only chooses from sampled points when selecting parameters for a given task, it is important to ensure that good solutions for each task are actually sampled. As discussed in Section 3, the mapping for such a decision maker is defined as

$$S^2(x_i) = \underset{a \in A_i^N}{\text{argmax}}\ \mu^N(x_i, a)$$

where $A_i^N = \{a^n | x^n = x_i, n \in \{1, ..., N\}\}$ is the subset of a values that have been measured on task x_i and we have utilised that $y^n = \mu^N(x^n, a^n)$. The REVI and CLEVI policies aim to maximise the peak posterior mean for each x_i, which can result in some tasks not being sampled. For example, if two tasks are very similar, it is only necessary to sample one of them to learn about both. However, for such an unsampled task, the decision maker would be restricted to select the best of the few randomly generated initial samples, and the overall performance of CLEVI and REVI is possibly rather poor when using $S^2(x_i)$.

In order to gain the performance advantages of using the posterior mean while only selecting parameters from sampled points, we propose here to sample according to REVI and CLEVI, however allocate the *final* n_X samples of the budget according to the EGO algorithm. Since we have n_X tasks, we sequentially allocate one sample to each task, and determine the parameter to sample by maximising the expected improvement of the next sample. The procedure is outlined in Algorithm 2.

As we show in the next section, maximising the posterior mean before applying EGO for the final samples is superior when compared to using EGO for all samples. We apply EGO

Algorithm 2 Risk averse REVI

Initialise n_0 samples using Latin Hypercube Design

update functions μ^{n_0}, k^{n_0}

for $n = n_0 + 1$ **to** $N - n_X$ **do**

$(x, a)^{n+1} \leftarrow \text{argmax } REVI_D(x, a)$

$y^{n+1} \leftarrow \theta(x^{n+1}, a^{n+1})$

update functions μ^{n+1}, k^{n+1}

end for

$i \leftarrow 1$

for $n = N - n_X$ **to** N **do**

$a^{n+1} \leftarrow \text{argmax } EGO(x_i, a)$

$y^{n+1} \leftarrow \theta(x_i, a^{n+1})$

update functions μ^{n+1}, k^{n+1}

$i \leftarrow i + 1$

end for

return $S^1(x_i) = \underset{a \in A_i^N}{\text{argmax}}\ \mu^N(x_i, a)$

for the last samples, instead of pure exploitation, because this is more efficient due to Jensen's inequality. Maximising the peak of the new dataset yields better samples than sampling the point with highest expectation $\mathbb{E}\left[\max\{y^1, ..., y^{n+1}\}\right] = \mathbb{E}\left[\max\{y^1, ..., \mu^n(x, a) + \sqrt{k^n(\tilde{x}, \tilde{x})}Z\}\right] > \max\{y^1, ..., y^n, \max_a \mu^n(x_i, a)\}$.

4.4. Efficient Monte Carlo Integration

Many previous works have looked at performing Monte-Carlo integration over Gaussian processes when calculating expected improvement. The Integrated Expected Conditional Improvement infill criteria of Gramacy and Lee (2011) Monte-Carlo integrates over a Gaussian process with a proposal distribution $g(x)$ that allocates samples to where the improvement in the model is considered greatest and is therefore most informative. For REVI we desire the X_{MC} to be distributed where the change in the Gaussian process is greatest, and importance sampling distributed according to the task kernel $g(x|x^{n+1}) \sim k_X(x, x^{n+1})$ seems a natural choice. However, we found that this resulted in REVI(x,a) becoming rough and not easily optimised since the X_{MC} points

move with x^{n+1} and also the precomputing of means described above cannot be done since X_{MC} is not constant. Instead, we propose to set $X_{MC} \sim \mathbb{P}[x]$ but filter the points such that tasks with little correlation with the sampled tasks are excluded. Essentially, this is importance sampling with $g(x|x^{n+1}) \sim \mathbb{P}[x]\mathbb{1}_{k_X(x,x^{n+1})>\delta}$. The choice of n_X must be large enough so that $\hat{\mathcal{I}}(x,a)$ is not dominated by a single term but multiple terms and not so large that $\hat{\mathcal{I}}(x,a)$ is estimated to unnecessarily high accuracy at great computational cost. Therefore n_X should be determined by meeting a minimum density requirement dependent on the Gaussian process kernel so that there are multiple correlating tasks with the sampled task.

For regions in $\tilde{X}$ that are not highly correlated with the new sample point, $\{\tilde{x} \in \tilde{X}, k^0(\tilde{x}, \tilde{x}^{n+1}) < \delta\}$, we propose to make the approximation $\tilde{\sigma}^n(\tilde{x}; \tilde{x}^{n+1}) \approx 0$. By enforcing sparsity on the vector of additive updates results in two computational speed-ups. Firstly, we may avoid the costly matrix multiplication involved in computing $\tilde{\sigma}^n(\tilde{x}; \tilde{x}^{n+1})$ for reference points that are largely unaffected by the sample. Secondly, for a given task x_i and set A_D, the vector of changes is given by $\tilde{\sigma}^n((x_i, A_D); (x,a)^{n+1})$ and will become sparse. When taking the max of linear functions with equal gradients, only the function with the highest intercept needs to be considered. Likewise, if multiple elements of $\tilde{\sigma}^n((x_i, A_D); (x,a)^{n+1})$ are zero, then all but the highest can be removed in the $KG(\mu, \tilde{\sigma})$ function before the sorting step and the for-loop of Algorithm 1 which will be applied to much shorter vectors. An example is given in Figure 3. For tasks that are uncorrelated with the sample task, the corresponding elements in the summation in $\hat{\mathcal{I}}(x,a)$ will be zero and only the large dominant terms will be calculated. Stationary kernels have intrinsic length scales and so we "sparsify" reference points that are beyond $r = 3$ length scales, i.e., where the Mahanalobis distance $\sqrt{(\tilde{x} - \tilde{x}^{n+1})D(\tilde{x} - \tilde{x}^{n+1})} > 3$ with D being a diagonal matrix of the square inverse of the GP length scales.

5. Comparison with the Profile Expected Improvement Algorithm

The Profile Expected Improvement (PEI) algorithm of Ginsbourger et al. (2014) considers almost the exact same problem we consider here, with the added assumptions that the task distribution is uniform. The algorithm they propose is a modification of the EGO algorithm where a sample maximises the expected improvement over a target value for the given task,

$$PEI(x,a) = \mathbb{E}\left[max\{y^{n+1} - T(x), 0\}\right]$$

where the new sample is given by the Gaussian Process, $y^{n+1} \sim N(\mu^n(x,a), k^n((x,a),(x,a))$. The target of improvement is given by the peak posterior mean, however capped by the highest value of the data seen so far $T(x) = \min\{\max_a \mu^n(x,a), \max Y^n\}$. We now show that this is a slightly modified simplification of the CLEVI algorithm. If we take the CLEVI acquisition function, firstly assume that the task distribution is uniform and the convolution is not applied so that the $\mathbb{P}[x]$ term can be discarded as constant. Secondly, set the sparsity approximation such that all points have zero additive update except for the sampled point, $\tilde{\sigma}^n((x, A_D \setminus \{a\}); (x,a)) = 0$. Thirdly, by assuming no noise in function observations the posterior standard error and the update to the mean at the sampled point are equal $\tilde{\sigma}^n((x,a);(x,a)) = \sqrt{k^n((x,a),(x,a))}$. Finally, if we augment the set A_D with the highest mean of the current task, $\text{argmax}_a \mu^n(x,a) \in A_D$, the CLEVI function simplifies to

$$\begin{aligned} CLEVI'(x,a) &= \mathbb{E}\left[\max\{\mu^n(x,a) + \sqrt{k^n((x,a),(x,a))}Z, \max_a \mu^n(x,a)\}\right] - \max_a \mu^n(x,a) \quad (20) \\ &= \mathbb{E}\left[\max\{y^{n+1} - T(x), 0\}\right] \quad (21) \end{aligned}$$

where $y^{n+1} = \mu^n(x,a) + \sqrt{k^n((x,a),(x,a))}Z$ and the target level is given by $T(x) = \max_a \mu^n(x,a)$. The only differences between the PEI acquisition function and the CLEVI function for a uniform task distribution without convolution, zero noise and maximum sparsity, is the addition of $\text{argmax}_a \mu^n(x,a)$ to the set A_D and the capping of the target value $T(x)$. By augmenting the set A_D, this has the advantage that the target level for improvement is more accurately measured. This requires an extra optimisation for each call to PEI in order to find the optimal a for the current task, though in our benchmarks we found this to be negligible. Another consequence of this optimisation is that the $PEI(x,a)$ acquisition function is no longer differentiable with respect to (x,a) since $dT(x)/dx$ is not analytically tractable and therefore PEI cannot be optimised by gradient descent with multiple starts which may cause excess evaluation in high dimensions, although this also may easily be remedied by taking the max over A_D instead of A as with CLEVI and REVI. Secondly, by assuming maximum sparsity, the effect one sample has on other predictions and on the target level itself is neglected and sampling is less efficient as we show in Section 6, particularly when there are long length scales in the Gaussian Process. The advantage of maximum sparsity however is that there are fewer posterior covariance calls which are each $O(n^2)$. Although the PEI algorithm was not designed with noisy problems in mind, the authors note that the Gaussian Process model may easily be adapted to account for noise and the acquisition function itself is

still applicable. In our benchmarks we find that when the assumptions are satisfied, negligible observation noise, uniform task distribution and unaffected target level, the performances of PEI, CLEVI and REVI are similar. However, on more varied scenarios PEI performs significantly worse than CLEVI and furthermore REVI significantly outperforms both. In our numerical experiments with non-uniform task distribution we modify the PEI algorithm to account for task density by weighting the acquisition function according to the point-wise task density,

$$PEI_{\mathbb{P}[x]}(x, a) = \mathbb{P}[x]\mathbb{E}\left[max\{y^{n+1} - T(x), 0\}\right]$$

such that high density tasks are given priority when sampling and low density tasks are only sampled if their improvement over their target level is high enough.

6. Numerical Experiments

We perform numerical experiments on three benchmark problems. In the first benchmark we use a continuous distribution of tasks and the popular Rosenbrock optimisation benchmark function comparing our algorithms against PEI and latin hypercube sampling. In the second benchmark we investigate the effect of dimensionality upon the REVI and CLEVI acquisition functions, we generate random functions from a Gaussian Process prior with dimensions varying from two to six, and again compare our algorithms against PEI and uniform sampling. Finally we consider the discrete task case and compare CLEVI and REVI against the SCoT algorithm (Bardenet et al., 2013) on risk neutral and risk averse scenarios.

6.1. Rosenbrock Test Function

For the first continuous benchmark problem, we use the Rosenbrock test function scaled such that it has domain $X \times A = [0, 100]^2$ and takes values in the range $y \in [-45, 0]$ and we add noise of variance $\sigma_\epsilon \in \{0.1^2, 1.0^2\}$. We test two different task distributions, a uniform distribution $\mathbb{P}[X] = 1/100$ and a triangular distribution $\mathbb{P}[X] = X * 2 * 10^{-4}$. Two noise levels and two task distributions yield four different experimental setups and for each setup we apply each algorithm 100 times with different initial designs and noise values. For each application, an initial budget of 20 samples is allocated by latin hypercube over the $X \times A$ domain after which a Gaussian Process with a squared exponential kernel is fitted. The hyper parameters of the Gaussian Process are estimated via maximum likelihood and updated after every new sample. Samples are sequentially

added to the initial design according to four algorithms, PEI, CLEVI, REVI and finally LEVI which is the CLEVI algorithm however without the convolution applied to the task distribution such that it is simply the integrand of Equation10. Each method is applied until a sampling budget of 80 samples has been exhausted. We also compare against latin hypercube sampling. To measure the quality of a mapping learnt by each method, for each experiment, a test set of 250 tasks values, X_{test}, are generated from $\mathbb{P}[x]$, and the difference in performance between the true optimal a and the performance of the a value determined by the mapping is averaged over all $x_i \in X_{test}$

$$\text{Opportunity Cost} = \frac{1}{250} \sum_{x_i \in X_{test}} \max_a \theta(x_i, a) - \theta(x_i, S^N(x_i))$$

The resulting average opportunity cost over 100 simulation runs for each algorithm for each budget is given in Figure 4 as well as one exemplary final sample design from each sampling method. All acquisition functions were maximised using the Nelder-Mead optimisation algorithm with $\min\{2n, 120\}$ random restarts and all default parameters in R's 'optim" function with the exception that the number of iterations was reduced to 50.

We see that in all cases REVI is the quickest to converge to the true optimal mapping and the CLEVI/LEVI methods are either similar or slightly worse. PEI frequently converges more slowly and in experiments not shown here this performance is replicated when using the CLEVI algorithm with maximum sparsity, therefore it is probably the assumption of a fixed target level that prevents PEI from converging as quickly. It is proven that in the infinite sample limit the PEI algorithm will converge, however the finite time behaviour is apparently different. The length scale in the parameter in the domain A is typically $l_A \approx 122$ while the largest possible distance between two points is 100, thus a sample at $(x, a)^{n+1}$ will affect the model $\mu^{n+1}(x^{n+1}, a)$ for all $a \in A$. Therefore, on this test function, the fixed target assumption is violated. The same behaviour was observed on the Branin-Hoo function that also has a length scale $l_A > 100$ when the domain is scaled to $A = [0, 100]$. The increase in noise reduces the speed of convergence, however does not change the relative ranking of algorithms. Comparing the uniform distribution with the triangular distribution, we see that the LEVI algorithm performs marginally worse due to its failure to account for the difference between the mode of the task distribution (which is also a boundary) and the maximum influence of a sample over the task distribution which is away from the boundary.

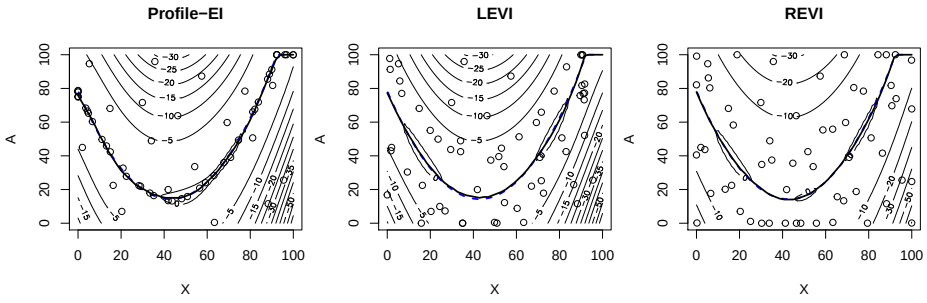
In Figure 4, we see the final design of experiments. The PEI algorithm allocates more samples to the predicted peak of each task however the CLEVI and REVI algorithms allocate samples more

evenly. REVI and CLEVI aim to maximise the posterior mean of the model and therefore allocate samples such that the whole model is updated to accurately predict the true peak and samples are scattered around $\pm 0.1 l_A$ of the true peak. This has the advantage that convergence is quicker and since the samples are more spread out there will be less chance of numerical issues when inverting the covariance matrix whilst fitting the Gaussian Process. This extra convergence must be traded off with the extra uncertainty over the *predicted* peaks and in this problem setting with a risk neutral decision maker REVI and CLEVI perform as expected. The CLEVI and REVI algorithms may be easily modified to maximise a lower confidence bound instead of the posterior mean (as done by Picheny et al. (2013a)) but we do not consider this case in our problem formulation.

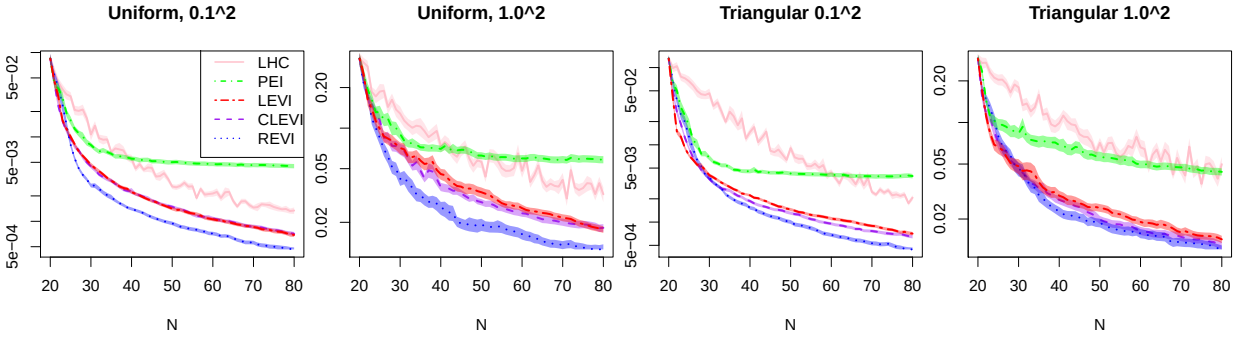
6.2. High Dimensional Test Functions

In our second benchmark we generate test functions from a Gaussian Process prior where the dimension of the task space X varies from one to five dimensions, $D_X \in \{1, 2, 3, 5\}$ and we fix $D_A = 1$. We do this for two reasons. Firstly, we aim to create a scenario where the assumptions of PEI are met and show that it performs well in this case, that is, where length scales are smaller than the domain and the target level is not always changed by the new sample. This may be done by increasing D_A or by reducing l_A, and to avoid sparsity we chose the latter. Secondly, the REVI algorithm requires a Monte-Carlo integral which can perform poorly as the number of dimensions increases. We initialise each sampling procedure with $10(D_X + 1)$ samples in a latin hypercube. In all experiments, we use a uniform task distribution.

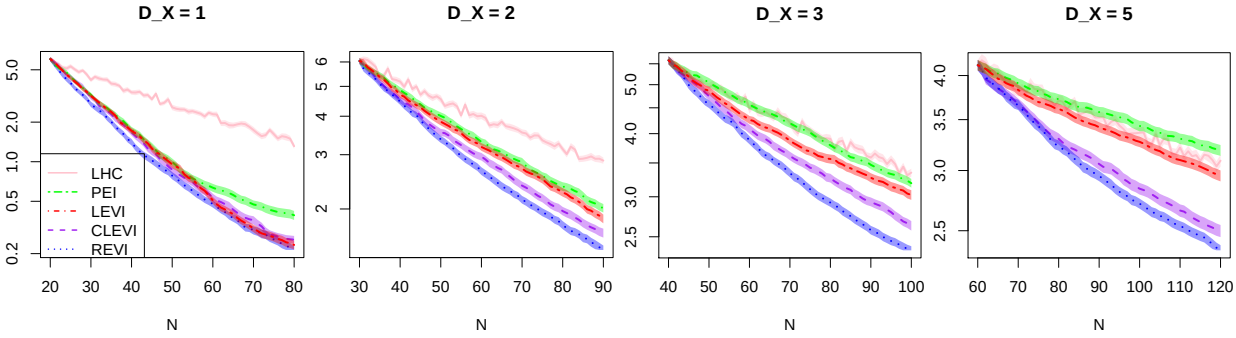
Gaussian Processes are well known to struggle in higher dimensions due to either data sparsity in high dimensional space or the n^3 computational cost or matrix ill-conditioning when data is dense in high dimensional space. In preliminary experiments, we found that all algorithms performed equal with latin hypercube sampling for $D_X = 3, 5$ when all length scales were $l_X = l_A = 10$. The initial design was too sparse and the initial samples were allocated to fill empty space and the advantages of REVI and CLEVI provided no significant benefit over PEI or uniform. To create a scenario without the data sparsity we increase the length scale with dimension such that the nearest neighbour in the initial design is approximately $1.3 * L$ where L is the length scale of the kernel used for all dimensions. The resulting length scales are 10, 16, 22, 28 and 33 for 1, 2, 3 and 5 dimensions, respectively, where the design space is $[0, 100]^{D_X+1}$ and the process variance is $\sigma^2 = 10^2$. The kernel parameters for generating the functions were also used when fitting the



(a) Final sampling allocations



(b) Rosenbrock test function opportunity cost



(c) GP test functions opportunity cost

Figure 4: In all cases, REVI produces the best mappings for all experiments and budget sizes, CLEVI and PEI are often equal, however diverge for large budgets where noise variance becomes significant. For the Triangular distribution, REVI outperforms other methods due to its ability to account for regional effects.

Gaussian Process, therefore the only difference between experiments is the acquisition functions. We apply all the same algorithms as from the previous benchmark, however the optimiser has more restarts $\min\{(3+D_X)n, 90+30D_X\}$ and for REVI we set $n_X = (3+D_X)\sqrt{n}$ to be consistent with

the previous experiment. All function evaluations have a noise added, $\epsilon \sim N(0, 1)$.

For the lowest dimensional case we see that all algorithms perform equally. We see that as dimensions increases, the methods that neglect covarying tasks, PEI and LEVI, get worse, and when $D_X = 5$, they do not significantly differ from latin hypercube sampling. Likewise, the REVI and CLEVI algorithms do not suffer as much with increasing dimension. The tasks are uniformly distributed in a hypercube, samples on the boundary of the hypercube have fewer neighbouring tasks which may be improved by the sample. As the number of dimensions increases, there are more edges, vertices and boundaries to avoid, and with increasing length scale the boundary affects more space within the hypercube. The REVI function measures the improvement at tasks, and at boundaries there are fewer tasks and thus smaller improvement in the mapping. Consequently, such areas are less favourable to sample. For CLEVI, by taking the convolution of the task distribution, the sharp boundaries in the true task distribution are rounded and reduced and CLEVI also tends to sample away from boundaries.

6.3. Finite Tasks

In our third benchmark, we compare the discrete task versions of CLEVI and REVI against the Surrogate based Collaborative Tuning algorithm, (SCoT) proposed by Bardenet et al. (2013). The SCoT algorithm tackles the complex problem of predicting good hyper parameters for machine learning algorithms based on the features of the dataset to which the algorithm is applied. Therefore, X is the space of dataset features, A is the hyper parameters of a machine learning optimisation algorithm and $\theta(x, a)$ is the test set accuracy of the trained algorithm with the given parameters on the given dataset. The proposed algorithm fits a Gaussian Process to predict the test accuracy of new algorithm parameters on a new dataset and sequentially executes a parameter setting on a dataset in order to learn the optimal parameter setting for all datasets. Samples are allocated to tasks x_i in a round robin fashion and the parameter setting a is determined by maximising the expected improvement of a new test accuracy measurement over the current best test accuracy measurement for the current task, that is by the EGO algorithm. Therefore the SCoT algorithm is equivalent to repeated application of the second stage of Algorithm 2. The authors note that the framework can accommodate any acquisition function and we apply the Knowledge Gradient acquisition function to find the optimal parameter for each task with the task determined by round robin allocation. Therefore this algorithm is equivalent to the CLEVI algorithm

without the approximation to account for the influence on other tasks and where the task sampling sequence is predetermined. We also again compare with random allocation. We generate random test functions as with the $D_X = D_A = 1$ case described above, however when sampling and measuring opportunity cost, task values are restricted to a finite set of randomly generated numbers $X = X_{MC} = X_{test} \in \mathbb{R}^{20}$ that are distributed according to $x_1, ..., x_{10} \sim N(20, 10^2)$ and $x_{11}, ..., x_{20} \sim N(50, 5^2)$. We measure the opportunity cost using the two mappings discussed in Section 3: the risk neutral mapping, executing the best predicted parameter for each task, and the risk averse mapping, executing the best evaluated parameter for each task.

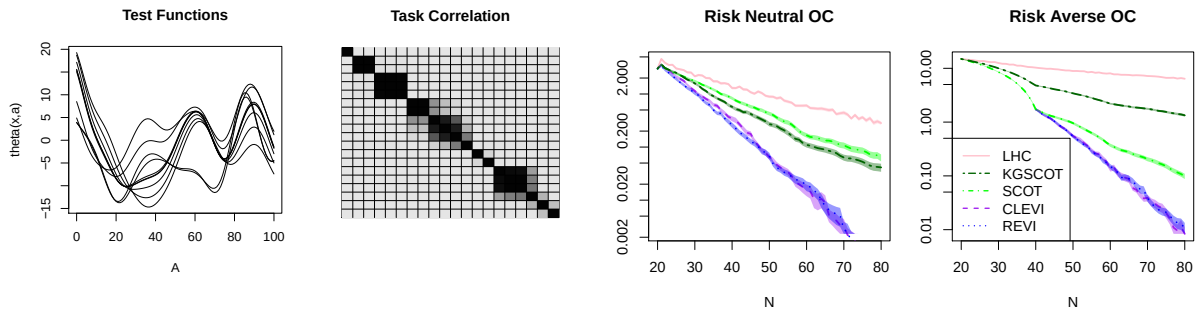


Figure 5: (a) one test function realisation, (b) one example task covariance matrix, (c) and (d) opportunity cost for a risk neutral/averse user averaged over 200 test function realisations.

As can be seen in Figure 5, in the risk neutral case, REVI and CLEVI perform best for all budget sizes. Replacing EGO with Knowledge Gradient in the SCoT algorithm did not yield much improvement suggesting that forcing the task allocation to be round robin accounts for most of the performance difference between CLEVI and SCoT. The REVI and CLEVI algorithms without the risk averse modification perform worse than both variations of SCoT and are not shown on the plot. However, with the risk averse modification, there is a very large improvement and therefore optimising the posterior mean before optimising the sample values yields a great benefit. In general, the risk averse opportunity cost is higher than the risk neutral, demonstrating the price a risk averse user must pay. In both cases, REVI and CLEVI do not differ significantly, suggesting a small number of positively correlated tasks in low dimensions do not benefit from the added accuracy of REVI, although further investigation is required.

7. Conclusion and Future Work

We have considered the problem of simultaneously identifying the optimal parameters for a set of tasks with correlation across tasks and where the performance of a particular parameter on a particular task has to be inferred from (potentially noisy) samples. To this end, we provide a general problem formulation, and propose two myopic information collection policies, CLEVI and REVI, that both aim to approximate the overall improvement across all tasks. CLEVI aims to maximise the expected improvement at the sampled task, weighted according to the regional influence the sample is expected to have, whereas REVI more accurately takes into account the regional influence, the information gain for other tasks due to the correlation structure. As expected, while CLEVI is computationally cheaper, REVI performs better, and both methods have equal leading order worst case complexity. We show that an alternative algorithm developed for the same problem, Profile Expected Improvement, that we consider state of the art from the literature, is a special case of our CLEVI algorithm and under certain conditions its performance is comparable. However, in almost all cases CLEVI and REVI converge toward the true optimal mapping much faster. Further empirical tests show that on discrete task sets, CLEVI and REVI also significantly outperform the SCoT algorithm, another algorithm from the literature, by a wide margin.

Furthermore, we have pointed out that the problem can be considered with two possible goals: Identifying a mapping that predicts the best parameter setting for any given task, and identifying a mapping that selects the best *sampled* parameter setting for each task. The latter is sensible in particular for a risk averse decision maker under a deterministic setting with a small number of tasks. We demonstrate that for such a setting, one should still collect information based on REVI, but switch to SCoT for selecting the last sample for each task.

There are several possible avenues for future work. In this work we have not considered model mismatch, real-world applications essentially always have model mismatch and this can affect the relative performance of Bayesian Optimisation algorithms as demonstated by Schulz et al. (2016). The REVI algorithm in particular maximally exploits covariance across tasks and decision variables and will likely suffer the most from inaccurately estimated covariance structure. Therefore the question remains, can exploiting poorly estimated covariance (REVI) be worse than ignoring only task covariance (CLEVI) or ignoring all covariance (PEI, SCoT)? To this end the proposed algorithms should be applied to various real world problems, including those applications mentioned in the

introduction, to reveal any application specific flaws or benefits. An extension to batch parallel sampling should be straightforward and speed up optimisation in practice. The distinction between searching for the solution with the best estimated performance, and searching for the solution with the best sampled performance, applies to all types of problems where Bayesian Optimisation is used, and should be examined also in other contexts. Finally, one might consider other notions of risk aversion, such as lower confidence bound, instead of the single extreme case we consider.

Acknowledgements

The first author gratefully acknowledges funding through the UK Engineering and Physical Sciences Research Council.

References

Bardenet, R., Brendel, M., Kégl, B., Sebag, M., 2013. Collaborative hyperparameter tuning. In: International Conference on Machine Learning. pp. 199–207.

Chevalier, C., Bect, J., Ginsbourger, D., Vazquez, E., Picheny, V., Richet, Y., 2014. Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set. Technometrics 56 (4), 455–465.

Deisenroth, M. P., Neumann, G., Peters, J., et al., 2013. A survey on policy search for robotics. Foundations and Trends® in Robotics 2 (1–2), 1–142.

Forrester, A. I., Sóbester, A., Keane, A. J., 2007. Multi-fidelity optimization via surrogate modelling. In: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences. Vol. 463. The Royal Society, pp. 3251–3269.

Frazier, P., Powell, W., Dayanik, S., 2009. The knowledge-gradient policy for correlated normal beliefs. INFORMS Journal on Computing 21 (4), 599–613.

Ginsbourger, D., Baccou, J., Chevalier, C., Perales, F., Garland, N., Monerie, Y., 2014. Bayesian adaptive reconstruction of profile optima and optimizers. SIAM/ASA Journal on Uncertainty Quantification 2 (1), 490–510.

Gramacy, R. B., Lee, H. K. H., 2011. Optimization under unknown constraints. In: Bayesian Statistics 9.

Hager, W. W., 1989. Updating the inverse of a matrix. SIAM review 31 (2), 221–239.

Heger, J., Branke, J., Hildebrandt, T., Scholz-Reiter, B., 2016. Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times. International Journal of Production Research, 6812–6824.

Hernandez-Lobato, J. M., Hoffman, M. W., Ghahramani, Z., 2014. Predictive entropy search for efficient global optimization of black-box functions. In: Advances in Neural Information Processing Systems. Curran Associates, Inc., pp. 918–926.

Hu, R., Ludkovski, M., 2015. Sequential design for ranking response surfaces. arXiv preprint arXiv:1509.00980.

Huang, D., Allen, T., Notz, W., Miller, R., 2006a. Sequential kriging optimization using multiple-fidelity evaluations. Structural and Multidisciplinary Optimization 32 (5), 369–382.

Huang, D., Allen, T. T., Notz, W. I., Zeng, N., 2006b. Global optimization of stochastic black-box systems via sequential kriging meta-models. Journal of Global Optimization 34 (3), 441–466.

Jones, D. R., Schonlau, M., Welch, W. J., 1998. Efficient global optimization of expensive black-box functions. Journal of Global optimization 13 (4), 455–492.

Krause, A., Ong, C. S., 2011. Contextual gaussian process bandit optimization. In: Advances in Neural Information Processing Systems. pp. 2447–2455.

Metzen, J. H., 2015. Active contextual entropy search. arXiv preprint arXiv:1511.04211.

Morales-Enciso, S., Branke, J., 2015. Tracking global optima in dynamic environments with efficient global optimization. European Journal of Operational Research 242, 744–755.

Pearce, M., Branke, J., 2016. Efficient information collection on portfolios. Tech. rep., University of Warwick.

Picheny, V., Ginsbourger, D., Richet, Y., Caplin, G., 2013a. Quantile-based optimization of noisy computer experiments with tunable precision. Technometrics 55 (1), 2–13.

Picheny, V., Wagner, T., Ginsbourger, D., 2013b. A benchmark of kriging-based infill criteria for noisy optimization. Structural and Multidisciplinary Optimization 48 (3), 607–626.

Poloczek, M., Wang, J., Frazier, P., 2017. Multi-information source optimization. In: Advances in Neural Information Processing Systems. pp. 4289–4299.

Poloczek, M., Wang, J., Frazier, P. I., 2016. Warm starting Bayesian optimization. In: Winter Simulation Conference. IEEE, pp. 770–781.

Rasmussen, C. E., Williams, C. K. I., 2004. Gaussian Processes for Machine Learning. MIT Press.

Schulz, E., Speekenbrink, M., Hernández-Lobato, J., Ghahramani, Z., Gershman, S., 2016. Quantifying mismatch in bayesian optimization. In: Nips workshop on bayesian optimization: Black-box optimization and beyond.

Scott, W., Frazier, P., Powell, W., 2011. The correlated knowledge gradient for simulation optimization of continuous parameters using Gaussian process regression. SIAM Journal on Optimization 21 (3), 996–1026.

Smith-Miles, K. A., 2008. Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys 41 (1), 1–25.

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

Swersky, K., Snoek, J., Adams, R. P., 2013. Multi-task Bayesian optimization. In: Advances in Neural Information Processing Systems. pp. 2004–2012.

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

Zhou, L., 2015. A survey on contextual multi-armed bandits. CoRR abs/1508.03326.
URL `http://arxiv.org/abs/1508.03326`