Optimizing Bayesian Optimization

Mark McLeod

Magdalen College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2018

Acknowledgements

I am extremely grateful to all the people that have provided support to me during my time as a student. Foremost must be my supervisors, Prof. Stephen Roberts and Prof. Mike Osborne, who have given me the freedom to pursue whatever avenues of research I have been inclined towards, while still providing guidance and assistance throughout. I would also like to thank all the of members, and visitors to, the Machine Learning Research Group for their friendship and help throughout, particularly Jonathan Downing, Ahsan Alvi, and Rasmus Bonnevie for an uncountable number of conversations about Gaussian Processes. Finally, I am profoundly grateful to Emma, for her unconditional support, without which this thesis would have been unimaginably more difficult to produce.

This work would not have been possible without funding from the Engineering and Physical Sciences Research Council.

All work contained within this document was completed by the author under supervision by Prof. Stephen Roberts and Prof. Mike Osborne. The material presented in Chapter 3 is based on

McLeod, M., Osborne, M.A. and Roberts, S.J., Optimization, Fast and Slow: Optimally Switching between Local and Bayesian Optimization. In *Proceedings of the 35thInternational Conference on Machine Learning.*

The material presented in Chapter 6 is based on

McLeod, M., Osborne, M.A. and Roberts, S.J., Adaptive Quadrature for Fast Sequential Hyperparameter Marginalization of Gaussian Processes. In *Workshop on Automatic Machine Learning 2018.*

Abstract

We are concerned primarily with improving the practical applicability of Bayesian optimization. We make contributions in three key areas.

We develop an intuitive online stopping criterion, allowing only as many steps as necessary to achieve the desired target to be taken. By combining this with intelligent online switching between acquisition functions and pure local optimization we are also able to substantially improve convergence to the local minimum associated with our final solution.

In cases where a continuum of reduced cost, but also reduced accuracy, evaluations are available we develop a Bayesian Optimization acquisition function to select both the location and cost of each evaluation. We achieve this with lower overheads than previous methods, translating to a real increase in performance. Part of this improvement is achieved by way of a new, more efficient, method for generating support points to sample the minimum of a Gaussian process. Further, in the case that the reduced cost estimates are unbiased we show that a practical solution cannot exist in most cases without also taking into consideration both computational overheads and a restriction on available resources. Given this knowledge we then develop a method which provides a viable solution in this setting.

Finally, we outline practical implementation details for Bayesian optimization which allow substantial reductions in the overhead costs without changing the theoretical properties of optimization. This is primarily achieved by use of adaptive quadrature to marginalize Gaussian process hyperparameters in place of the more common slice sampling approach.

Contents

List of Common Symbols

$\mathbf{0}_d$	The vector of zeros with length d.
$\mathbf{1}_d$	The vector of ones with length d.
c, $c()$	A cost or cost function.
d	The dimensionality of a problem.
$f()$	The function to be optimized.
r	A radius.
s	An environmental variable.
t	A time.
x	A point.
y	The value of a function.
$\mathbf{I}$	The identity matrix (size determined by context).
B	A computation budget.
$D_{\mathbf{KL}}()$	The KL-divergence between distributions.
H	A Hessian matrix.
$H()$	The entropy of a probability distribution.
K_{XY}	The Gram matrix for X and Y.
R	A regret.
$\alpha()$	An acquisition function.
$\delta()$	The Dirac delta function.
θ	A model parameter.

$\kappa(,)$ . . . . . A kernel function.

σ^2 . . . . . A variance.

$\phi()$ . . . . . The standard normal pdf.

Θ . . . . . A model parameter vector.

Σ . . . . . A covariance matrix.

$\Phi()$ . . . . . The standard normal cdf.

List of Abbreviations

BFGS . . . . . Broyden Fletcher Goldfarb Shanno.

BLOSSOM . . Bayesian and Local Optimization Sample-wise Switching Optimization Method.

CMA-ES . . . Covariance Matrix Adaptation Evolution Strategy

DIRECT . . . DIviding RECTangles.

EI . . . . . Expected Improvement.

EnvPES . . . . Environmental variable Predictive Entropy Search

EP . . . . . Expectation Propagation.

ES . . . . . Entropy Search.

FABOLAS . . FAst Bayesian Optimization on LArge data Sets

KL-divergence Kullback-Lieber divergence.

GP . . . . . Gaussian Process.

GRR . . . . . Global Regret Reduction.

LCB . . . . . Lower Confidence Bound.

MAP . . . . . Maximum A Posteriori.

MCMC . . . . Markov Chain Monte Carlo.

MLE . . . . . Maximum Likelihood Estimate.

MLQM . . . . Minimum of Local Quadratic Models.

MTBO . . . . Multi-Task Bayesian Optimization.

PES . . . . . Predictive Entropy Search.

PI . . . . . Probability of Improvement.

UCB . . . . . Upper Confidence Bound.

WLH . . . . . Weighted Local Hessians.

1 Introduction

Optimization is a field of study with centuries of history and numerous sub-fields relevant to widely varying disciplines. In this thesis we are concerned with the comparatively new field of Bayesian optimization using Gaussian processes, which originates from the broader field of response surface based optimization. The fundamental premise of Bayesian optimization is that each evaluation of the function to be optimized incurs substantial cost, prohibiting methods requiring large numbers of evaluations. In order to obtain good optimization performance while taking as few function evaluations as possible, we are willing to expend considerable computational resource on the selection of each location to evaluate. Optimizing the parameters of long running computational tasks, such as evaluating the log-likelihood of a machine learning algorithm conditioned on a set of hyperparameters [Snoek et al., 2012; Swersky et al., 2013; Hernández-Lobato et al., 2014], or the result of a physical simulation given certain design parameters [Forrester et al., 2008], or any task requiring human intervention to run a physical experiment [Lizotte et al., 2007; Tesch et al., 2011b; Calandra et al., 2016], are candidates for Bayesian optimization.

The field of Bayesian optimization has many active lines of research. Many new methods have been produced achieving successively superior performance for simple sequential evaluations [Jones et al., 1998; Hennig and Schuler, 2012; Hernández-Lobato et al., 2014]. However, this is a *greedy* strategy, selecting the

most valuable location to evaluate at each iteration without consideration for longer term consequences, and is not guaranteed to be optimal. Attention has therefore also been given to the more complex problem of obtaining the best possible performance over all (or at least more than one) future steps [Osborne et al., 2009; González et al., 2016b]. These approaches have some similarity to methods designed to exploit the parallel capabilities of modern hardware by undertaking multiple simultaneous evaluations of the objective [Jones, 2001; Ginsbourger et al., 2008; Azimi et al., 2012; González et al., 2016a]. Bayesian optimization has also been extended to multi-objective optimization [Hernández-Lobato et al., 2015] and to variable fidelity objectives [Swersky et al., 2013; Klein et al., 2017], in which discounted evaluations can be obtained at the cost of reduced similarity to the true objective.

The principal focus of this thesis is to improve the practicality of Bayesian optimization, with particular emphasis on the variable fidelity setting, and on reducing overhead costs. Following a review of the necessary background material and relevant literature in Chapter 2 we develop in Chapter 3 BLOSSOM, a new Bayesian optimization routine which uses automatic stopping, and switching between modes, to combine the good global properties of Bayesian optimization with the reliable fast convergence of local optimization. We then consider the variable fidelity setting in Chapter 4 and adapt the state-of-the-art predictive entropy search [Hernández-Lobato et al., 2014] algorithm to provide superior results to existing methods in this context. We also develop a new fast method of generating support point sets, a necessary and time consuming process in this and later chapters. We continue with the variable fidelity setting in Chapter 5, showing that the existing approaches do not provide practical solutions in the special case of unbiased variable fidelity. By taking into account predicted optimization performance and overhead costs, we then demonstrate a potential solution to this problem. In Chapter 6, we consider improvements to the overhead costs of implementation, regardless of the specific Bayesian optimization algorithm in use. We show that deterministic adaptive quadrature for hyperparameter marginalization can provide comparable performance to MCMC sampling at substantially lower costs, due to the use of

$\mathcal{O}(n^2)$ Cholesky updates. We also remark on the potential cost reduction available by using batched evaluations of the acquisition function. Finally we conclude and suggest possible future developments in Chapter 7.

2 Background

Contents

In this chapter we will review the necessary background theory and relevant recent literature in the field of Bayesian Optimization.

2.1 Probability

In any application of machine learning, we must make decisions despite not having perfect knowledge. In the context of optimization we must attempt to identify the

global minimizer of our objective function despite only having access to a finite number of evaluations, which may have been corrupted by noise. We therefore require a set of rules allowing us to express our belief over uncertain quantities and update those beliefs given additional data. In order to reason under uncertainty we make use of probability theory, for a full introduction we recommend Jaynes et al. [2003]. We take a *Bayesian* perspective on probability, allowing us to express our belief about the value of an unknown quantity using probabilities, even if that quantity is fixed. For example, the result of evaluating a function at some location may be completely deterministic, but we nonetheless place a probability distribution over the result to represent our belief, given our only partial information, over the value that would be observed. We then update the distribution to reflect our changing belief whenever we observe the function value at other nearby locations. We now give a brief overview of the key concepts that we will make use of.

2.1.1 Basic Concepts

In probability theory we are concerned with a variable which may take on any value in some sample space Ω. We call any subset of Ω an event, and use a function, p, to assign a probability to the likelihood that that event will occur. For example, if we consider the outcome of rolling an unbiased six sided die, d, the sample space is $\Omega = \{1, 2, 3, 4, 5, 6\}$. The probability of the whole sample space is defined to be unity while the probability of the empty set is defined to be zero. Each of the individual outcomes is an event, for which in this example the probabilities are equal sixths ($p(d = 2) = \frac{1}{6}$), but we can also consider more complex events such as the outcome being even ($p(\texttt{even}) = 0.5$) or greater than two ($p(d > 2) = \frac{2}{3}$).

We will also wish to consider the interactions of the probabilities of multiple events. If we are concerned with the joint probability of two events both occurring, for example the event $d > 3 \cap \texttt{even}$, the outcome of a dice roll being both even and greater than three, we write $p(\texttt{even}, d > 3)$. If we wish to denote the probability of an event conditioned on having already observed some other event, for example

the probability that d is even given that we have been told that the outcome is greater than three, we write $p(\texttt{even} \mid d > 3)$.

For a variable x in some continuous domain $\mathcal{X}$ there may be an infinite number of possible outcomes. We abuse notation by redefining $p(x)$ in this context to be the *probability density function*, pdf, which if integrated over any subdomain of $\mathcal{X}$ provides the probability that x will fall within that region. In the one dimensional setting it is often useful to consider the probability that x will be less than some value a, which we call the *cumulative distribution function*, $\text{cdf}(a) = p(x \leq a) = \int_{x_{\text{lower}}}^{a} p(x)\mathrm{d}x$, where x_{lower} is the lower bound of $\mathcal{X}$.

Calculations using probability are governed by the following basic rules:

The Sum Rule

Necessarily, the probability of all possible outcomes must be unity. Therefore if there are N possible mutually exclusive events, $A_1, A_2 \ldots A_N$, then

$$\sum_{i=1}^{N} p(A_i) = 1\,, \tag{2.1}$$

or in the continuous case

$$\int_{\mathcal{X}} p(x)\mathrm{d}x = 1\,. \tag{2.2}$$

The Product Rule

The probability of two events both occurring is

$$p(A, B) = p(A \cap B) = p(A \mid B)p(B)\,, \tag{2.3}$$

and since $p(A, B) = p(B, A)$ this can also be expressed as

$$p(A, B) = p(B \cap A) = p(B \mid A)p(A)\,. \tag{2.4}$$

From the sum and product rules we can express the probability of either of two events, $A \cup B$, which is given by

$$p(A \cup B) = p(A) + p(B) - p(A, B)\,. \tag{2.5}$$

2.1.2 Operations on Probability

The operations which we will wish to perform frequently are: *marginalization*, in which we express a probability over multiple quantities in terms of only those we are concerned with; *inference*, in which we use Bayes' rule to update an existing belief with new information; and *sampling*, in which we generate a sequence of values drawn from a given distribution.

Marginalization

We often have an expression for the probability of some outcome A conditioned on a partition of the sample space B_i, $p(A \mid B = B_i)$, and would like to find $p(A)$. This is given in the discrete case where there are N mutually exclusive outcomes $B_1, B_2 \dots B_N$ by

$$p(A) = \sum_{i=1}^{N} p(A \mid B = B_i) p(B_i)\,, \tag{2.6}$$

or in the continuous case where we have $p(x \mid y)$, the probability density at x conditioned on the value of some other outcome y, we can find $p(x)$ alone by

$$p(x) = \int p(x \mid y) p(y) \mathrm{d}y\,. \tag{2.7}$$

If the integral in Equation 2.7 does not take a tractable form we must choose an appropriate quadrature routine to approximate the result as a weighted sum over N selected values of y,

$$p(x) = \sum_{i=1}^{N} w_i\, p(x \mid y_i) \tag{2.8}$$

where the w_i and y_i are determined by our choice of method. This choice will depend on the specific properties of the problem under consideration, since the relative performance of different quadrature routines is dependent on both the dimension and shape of the function to be integrated [Schürer, 2003]. In general for relatively easy to evaluate and low dimensional expressions we may use the trapezoid rule, or any other common cubature rule. These schemes have excellent performance for low dimensional problems but, in higher dimensions the curse of dimensionality

requires an exponentially increasing number of function evaluations. We may then prefer quasi-Monte Carlo routines, which make use of carefully chosen evaluation sequences to obtain performance with a much lower dependence on dimensionality (convergence is $\mathcal{O}(\frac{\log^d n}{n})$), or full Monte Carlo, which is completely independent of dimensionality. Taking draws from $p(y)$ and using equal weights is the simplest case of Monte Carlo integration, and obtains $\mathcal{O}(\frac{1}{\sqrt{n}})$ convergence. This final approach is commonly used to marginalize hyperparameters for Bayesian Optimization, although in Chapter 6 we will propose an alternative strategy of adaptive quadrature.

Bayes' Rule

Assuming $p(B) > 0$, we can rearrange the two expressions of Equation 2.4 to give Bayes' rule

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)}\,. \tag{2.9}$$

We now rename the variables such that A is the quantity we are concerned with and B is observed data. Bayes' rule now defines the *posterior* probability of A in terms of the *prior* belief, $p(A)$, the *likelihood* of the data under the prior, $p(\mathrm{B} \mid A)$, and the *marginal* probability of the data

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal}}\,. \tag{2.10}$$

That is, the rule defines how we should update our existing belief on obtaining new data.

We can usually avoid evaluation of the marginal, which as the name suggests requires a marginalization which may not be simple to compute. Since the marginal is a constant its function is to normalize the resulting posterior distribution. If $p(A \mid B)$ takes the form of a known distribution we can find the required parameters from the numerator alone. Alternatively, we can often continue using an unnormalized quantity which is proportional to $p(A \mid B)$ by discarding the marginal entirely: $q(A \mid B) = \text{likelihood} \times \text{prior}$. Bayes' Rule is often applied as a chain over multiple steps. At each iteration we use a single item of data, for example the new objective

function evaluation result in an optimization problem, to update from prior to posterior. This posterior then becomes the prior of the next step.

Sampling

We often desire to generate a sequence of draws from a given distribution $p(x)$. For discrete or one-dimensional distributions we can very quickly generate draws that are fully independent and follow to $p(x)$ to any required precision by using the inverse of the cdf to transform samples drawn from the uniform distribution. We can also combine these to generate draws from more complex or higher dimensional distributions which take convenient forms. For example a sample of any D dimensional multivariate normal distribution can be obtained by a linear transform of D independent draws from the univariate normal distribution.

However, many distributions exist from which we cannot so easily take samples despite being able to evaluate $p(x)$. There are many methods of generating samples in this situation, an introduction to the field is available in, for example, Murphy [2012]. However, these methods do have drawbacks. They require much more computational power, since $p(x)$ is usually evaluated many times for each sample generated. Furthermore, methods based on Markov Chain Monte-Carlo (MCMC), a very common class of sampling algorithm, have both considerable correlation between consecutive samples, and also require some number of initial samples to be discarded since the sequence of samples only converges to $p(x)$ after an initial period.

2.1.3 Entropy

It is often useful to be able to quantify the level of uncertainty about a variable conveyed by a probability distribution. This property is called the *Entropy* of the distribution, and in the case of a distribution over N discrete values this is defined as

$$H(p) = -\sum_{i=1}^{N} p(x_i) \log(p(x_i)) \,. \tag{2.11}$$

If we use the base-2 logarithm and consider the context of encoding a stream of samples from $p(x)$ using binary digits this can be interpreted as minimum expected

number of bits required per sample. For example, consider a distribution with three values such that $p(x_1) = 0.5$, $p(x_2) = 0.25$, $p(x_3) = 0.25$. The entropy of this distribution is 1.5 and if we represent x_1 as `1`, x_2 as `00`, and x_3 as `01` then the expected number of bits in our representation is also 1.5. This interpretation in terms of discrete encoding for communication, and the use of entropy in general, originates from Shannon et al. [1951].

We will make use of the differential entropy of a distribution, (an extension of this concept to continuous variables), defined for some distribution $p(x)$ as

$$H(p) = -\int p(x)\log(p(x))\mathrm{d}x\,. \tag{2.12}$$

Unlike the discrete form this quantity is *not* invariant to a scaling $x' = ax$, $a \in \mathbb{R}$ (The entropy of a discrete distribution is the same whatever values of labels we assign to the x_i) but this will be useful to us, as a distribution of the same shape for which probability mass is more tightly distributed does represent an increased state of knowledge in the context of optimization.

The differential entropy has a useful duality property which we shall make use of. We begin with the definition of the mutual information, I, between $p(x)$ and $p(y)$:

$$\begin{aligned}
I(p(x),p(y)) &= \int\int p(x,y)\log\frac{p(x,y)}{p(x)p(y)}\mathrm{d}x\mathrm{d}y \\
&= \int\int p(x,y)\log\frac{p(x,y)}{p(x)}\mathrm{d}x\mathrm{d}y - \int\int p(x,y)\log p(y)\mathrm{d}x\mathrm{d}y \\
&= \int\int p(y\mid x)p(x)\log\frac{p(x)p(y\mid x)}{p(x)}\mathrm{d}x\mathrm{d}y - \int\log p(y)\int p(x,y)\mathrm{d}x\mathrm{d}y \\
&= \int p(x)\left(\int p(y\mid x)\log p(y\mid x)\mathrm{d}y\right)\mathrm{d}x - \int\Big(\log p(y)\Big)p(y)\mathrm{d}y \\
&= \int p(x')H[p(y\mid x=x')]\mathrm{d}x' + H[p(y)] \\
&= H[p(y)] - \mathbb{E}_{p(x)}\Big[H[p(y\mid x)]\Big]\,.
\end{aligned} \tag{2.13}$$

Since we could also have used the equivalent form $p(x,y) = p(x\mid y)p(y)$ of the product rule on the third line to obtain the same expression with a reversed order of conditioning we have

$$H[p(x)] - \mathbb{E}_{p(y)}\Big[H[p(x\mid y)]\Big] = H[p(y)] - \mathbb{E}_{p(x)}\Big[H[p(y\mid x)]\Big]\,. \tag{2.14}$$

The implication of this equality is that the expected change in the differential entropy of our belief over x that occurs by conditioning on y is equal to the expected change in the differential entropy of our belief over y obtained by conditioning on x.

We will also be using the Relative Entropy, or Kullback-Liebler (KL) divergence, which measures the difference in information between two distributions. The KL divergence is defined from one distribution, $q(x)$, to another $p(x)$ as

$$D_{KL}(p,q) = \int p(x) \log \frac{p(x)}{q(x)} \mathrm{d}x \,. \tag{2.15}$$

We have shown here the continuous definition. In the discrete case we can interpret the KL divergence as the number of additional bits per symbol that are required if we have used an encoding designed for p but the true distribution is in fact q. To illustrate this consider the example of encoding three values used above, but let the distribution instead be $q(x_1) = 0.25$, $q(x_2) = 0.5$, $q(x_3) = 0.25$. If we continue to use the representations designed for p the expected number of bits in our representation rises from 1.5 to 1.75. If we now evaluate $D_{KL}(p,q)$ we find that it is 0.25 as expected.

2.2 The Gaussian Distribution

As a prelude to introducing the Gaussian process we must first define the Gaussian, or normal, distribution which is a particularly useful member of the exponential family of probability distributions. The multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ over a real valued vector has the probability density function

$$p(x) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \tag{2.16}$$

where μ is a mean vector and Σ is some valid (symmetric and positive definite) covariance matrix. We can view this distribution as the combination of d independent zero mean unit normal distributions on each of the dimensions of some $z \in \mathbb{R}^d$, which have undergone an affine transform $x = Az + \mu$, for some A such that

$\Sigma = AA^T$, to achieve the required covariance and mean. The differential entropy of the multivariate Gaussian has a simple closed form:

$$H\Big(\mathcal{N}(\mu, \Sigma)\Big) = \frac{1}{2} \log \Big((2\pi \mathrm{e})^d \det(\Sigma)\Big) \ . \tag{2.17}$$

The Gaussian distribution has a number of particularly useful properties:

Linear combination

The sum of any number of Gaussian distributed variables also takes a Gaussian distribution. In one dimension if x_1 has distribution $\mathcal{N}(\mu_1, \sigma_1^2)$ and x_2 has distribution $\mathcal{N}(\mu_2, \sigma_2^2)$, with covariance ρ, then their sum $z = x_1 + x_2$ is distributed as $\mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2 + 2\rho)$. Extending this to the multivariate setting any affine transform of a Gaussian also remains Gaussian distributed. If we transform some distribution over $x \in \mathbb{R}^{d_1}$ into $z \in \mathbb{R}^{d_2}$ such that $z = Rx + c$, where R is any matrix of size $d_2 \times d_1$ and c is any offset of size d_2, then the resulting distribution over z is $\mathcal{N}(R\mu + c, R\Sigma R^T)$.

Marginal

Any marginal of a Gaussian distribution is also Gaussian. If we partition a d dimensional Euclidean space, $x \in \mathbb{R}^d$, over which we have placed a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ such that

$$x = \left[\frac{x_1}{x_2}\right], \qquad \mu = \left[\frac{\mu_1}{\mu_2}\right], \qquad \Sigma = \left[\begin{array}{c|c} \Sigma_{11} & \Sigma_{12} \\ \hline \Sigma_{12}^T & \Sigma_{22} \end{array}\right] \tag{2.18}$$

then x_1 and x_2 have marginal distributions $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$ respectively. Since as noted any affine transformation of Gaussian distributions remains Gaussian we can also obtain the marginals for any transformed set of axes.

Posterior

The Gaussian distribution is its own conjugate prior. That is, given a Gaussian prior belief, and conditioning on an observation corrupted by Gaussian noise, the posterior distribution is also Gaussian. Using the same partitioning as above, observing $x_1 = a$

under a noise model with zero mean and independent σ^2 variance on each component of x_1 yields a posterior distribution on x_2 of $\mathcal{N}(\mu_{\text{post}}, \Sigma_{\text{post}})$ where

$$\begin{aligned} \mu_{\text{post}} &= \mu_2 + \Sigma_{12}^T(\Sigma_{11} + \sigma^2 I)^{-1}(a - \mu_1) \\ \Sigma_{\text{post}} &= \Sigma_{22} - \Sigma_{12}^T(\Sigma_{11} + \sigma^2 I)^{-1}\Sigma_{12}\,. \end{aligned} \tag{2.19}$$

These properties of the Gaussian distribution are of particular usefulness as they allow us to repeatedly perform inference and marginalization with additional data on any quantities of interest, while our distribution remains Gaussian. Unfortunately, unless each sequential observation is fully independent we must also increment the dimension of the observation covariance matrix Σ_{11} at each iteration. This leads to increasing computational overheads, since evaluating the terms containing Σ_{11}^{-1} in general requires an $\mathcal{O}(n^3)$ Cholesky decomposition. As we shall see this cost is a key contributor to a practical limit on the usefulness of Bayesian optimization with Gaussian processes, introduced in the next section.

2.3 Gaussian Processes

In order to perform optimization in a principled manner we require some method of providing estimates of the value of some function of interest, f, at any location, given observations so far. We make use of the Gaussian process (GP) which is an extremely flexible model with a number of useful properties. For a more complete introduction we recommend Rasmussen and Williams [2006]. The use of Gaussian processes is also sometimes called *kriging*, usually in the context of engineering design applications. The value of this function at any location is modelled as taking a Gaussian distribution. We use a kernel function $\kappa(x_1, x_2) \in \mathbb{R}$, to determine the covariance between the function value at any two locations x_1 and x_2. The required properties of this function and how it controls the nature of predictions made by the GP will be discussed in Section 2.3.1. The prior distribution over the function values at any set of points $X = \{x_0, x_1, \ldots x_n\}$ is then given by a multivariate normal distribution $\mathcal{N}(\mu(X), K_{XX})$, where $K_{XX}[i, j] = \kappa(x_i, x_j)$ and

$\mu(x)$ is a prior mean function. The mean function can take any form, in this work we have always used a constant $\mu(x) = \mu_0$.

Since the Gaussian distribution is its own conjugate prior, the updated belief on our function conditioned on observations of the function value which have been corrupted by normally distributed noise is also a multivariate Gaussian distribution. The posterior, at locations $Z = \{z_0, z_1, \dots z_n\}$ conditioned on observations $Y = \{y_0, y_1, \dots y_n\}$ at locations X, with zero-mean, independent Gaussian distributed noise $\mathcal{N}(0, \sigma^2 I)$ is given by $\mathcal{N}(\mu_Z, \Sigma_{ZZ})$, where

$$\begin{aligned} \mu_Z &= K_{ZX}^T (K_{XX} + \sigma^2 I)^{-1} (Y - \mu_0 \mathbf{1}) + \mu_0 \mathbf{1} \\ \Sigma_{ZZ} &= K_{ZZ} - K_{ZX}^T (K_{XX} + \sigma^2 I)^{-1} K_{ZX} \\ K_{ZX}[i,j] &= \kappa(x_i, z_j) \\ K_{ZZ}[i,j] &= \kappa(z_i, z_j)\,. \end{aligned} \tag{2.20}$$

Since K_{XX} must be a symmetric positive definite matrix these equations are usually solved using the Cholesky decomposition [Rasmussen and Williams, 2006].

Using a Gaussian process model allows us to perform inference on functions values, provides us with an estimate of the uncertainty of those predictions and allows us to draw samples from the posterior over any finite set of points of interest. We also have a closed form expression for the log-likelihood of our observations

$$\begin{aligned} \log(p(Y \mid X, \kappa, \mu_0)) = &-\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(\det(K_{XX})) \\ &- \frac{1}{2}(Y - \mu_0 \mathbf{1})^T (K_{XX} + \sigma^2 I)^{-1} (Y - \mu_0 \mathbf{1}) \end{aligned} \tag{2.21}$$

These facilities will be particularly useful in the optimization setting.

2.3.1 Kernel functions

The predictions made by the GP depend on the properties of the kernel $\kappa(x_0, x_1) \in \mathbb{R}$, $x_0, x_1 \in \mathbb{R}^d$, which determines the covariance between the function values at two locations in our input space. This function is restricted to forms which produce a positive definite covariance matrix K for all possible inputs, but within this restriction we are free to make any choice of kernel which is appropriate to the function we desire to model. Our choice of kernel function encodes our prior belief

about the nature of the function being modeled, and therefore has a crucial effect on the predictions made by a GP. Even considering only Cartesian coordinates a large number of kernels with particular characteristics are available to us. For example we may wish to encode in our prior that the function being modeled must possess characteristics such as a specified level of differentiability, or specific parametric forms. These forms can be combined with each other by summation, multiplication and convolution to suit any particular structure we wish to encode in our GP model. For example by multiplying a kernel encoding a fixed periodicity with a kernel encoding linear behviour we are able to model functions of a single frequency, but with linear variation in amplitude. A useful introduction to this method of combining simple kernels to model more complex functions is provided by Duvenaud [2014]. In this work we restrict ourselves to two particularly common kernels used in Bayesian optimization, in addition to additive white noise.

The Squared Exponential Kernel

The squared exponential kernel (also called Gaussian or radial basis function) is defined in d dimensions as

$$\kappa_{\mathrm{SE}}(x_0, x_1) = A^2 \exp\left(-\frac{1}{2}r^2\right) \tag{2.22}$$

where

$$r^2 = \sum_{i=0}^{d} \left(\frac{x_0^i - x_1^i}{h_i}\right)^2 . \tag{2.23}$$

This is a particularly common choice of kernel in machine learning, and has the property that samples drawn from a zero-mean GP with this kernel will be infinitely differentiable. The shape parameters, h_i, determine the characteristic length of variations in each axis, while the scale parameter, A, determines the magnitude of features.

The Matérn Kernel

The Matérn kernel is defined as

$$\kappa_{\mathrm{M}}(x_0, x_1) = A^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} r\right)^{\nu} K_{\nu}(\sqrt{2\nu} r) , \tag{2.24}$$

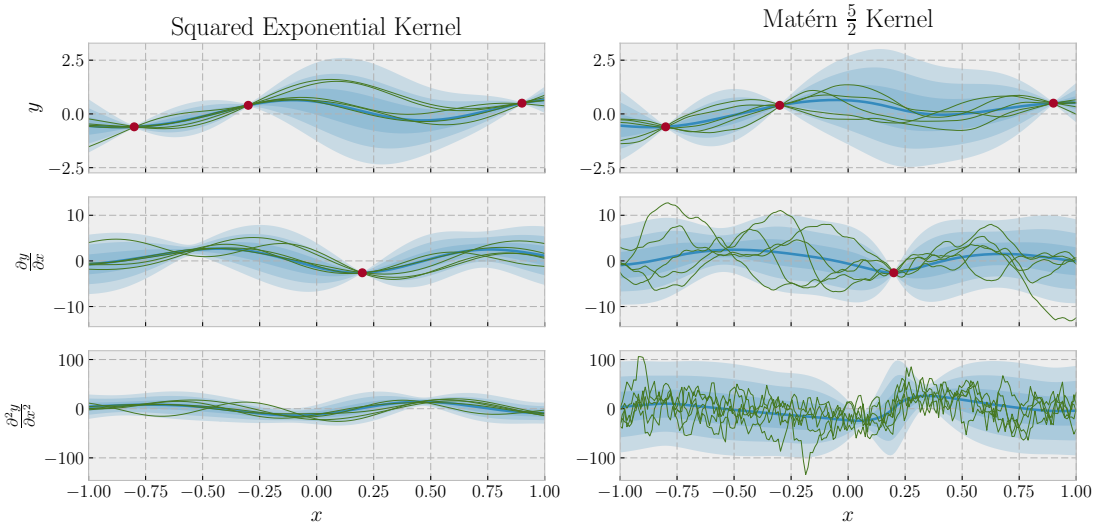

Figure 2.1: Gaussian process posterior prediction and samples using the Squared Exponential and Matérn 5/2 kernels for the value, gradient and Hessian of a function conditioned on observations of both the function values and gradient. Both kernels use a lengthscale of 0.4 and unit output scale. While the mean and standard deviations (shown by shading) of the function values are similar, the predictions and samples of the derivatives clearly show the difference in smoothness between the two kernels.

where r takes the same form as above and K_ν is a modified Bessel function of the second kind [Abramowitz and Stegun, 1964]. The parameters A and h have the same effects as before, while the smoothness of predictions is determined by ν. In the limit $\nu \to \infty$ the Matérn kernel is equal to the squared exponential kernel. For finite values of ν, predictions are $\lfloor \nu \rfloor$ times differentiable. When ν takes a half-integer value, the kernel has a simple form that does not require the evaluation of a Bessel function. We make use of the $\nu = \frac{5}{2}$ form of the Matérn kernel, which takes the form

$$\kappa_{\mathrm{M52}}(x_0, x_1) = A^2 \left(1 + \sqrt{5}r + 5r^2\right) \exp(-\sqrt{5}r)\,, \tag{2.25}$$

and when used provides samples from a zero-mean GP which are twice differentiable. This reduced smoothness in comparison to the Squared Exponential kernel is demonstrated in Figure 2.1.

The White Noise Kernel

A function taking independent Gaussian distributed values at all points in the input space can be represented by the white noise kernel

$$k_{\mathrm{W}}(x_0, x_1) = \sigma^2 \delta(x_0 - x_1)\,. \tag{2.26}$$

This kernel can be summed with any other kernel to represent additive Gaussian observation noise with variance σ^2. However, since we usually wish to infer the underlying function value before observation noise this component is usually shown explicitly in the inference equations, as we have done in Equation 2.20, as the addition of a diagonal component $\sigma^2 I$ added to K_{XX}. If the observation noise has varying magnitude we can replace σ^2 with some positive function of x to represent heteroskedastic noise. For even more flexibility we can replace $\sigma^2 I$ with $\mathrm{diag}(S)$, where each element s_i of S is the variance of the noise associated with the i^{th} observation in our dataset.

2.3.2 Hyperparameter Marginalization

So far we have specified our kernels as $\kappa(x_0, x_1)$, We should more correctly write $\kappa(x_0, x_1 \mid \Theta)$, where Θ is some constant vector. Since these constants are not parameters of the system we are modelling, but parameters of our model, we call them *hyperparameters*. In the kernels described above the hyperparameters are the output scale A, the lengthscales h_i, and the smoothness ν parameters.

Since we do not expect to have exact knowledge of "correct" values for the hyperparameters we must also give these values appropriate Bayesian treatment. We can use Bayes' rule to write the hyperparameter posterior probability in terms of the GP posterior probability and a prior $p(\Theta)$.

$$\begin{aligned} p(\Theta \mid X, Y) &= \frac{p(Y \mid \Theta, X) p(\Theta)}{p(Y \mid X)} \\ &\propto p(Y \mid \Theta, X) p(\Theta) \end{aligned} \tag{2.27}$$

In practice we will use a prior of independent components with broad support and perform calculations using log values of these quantities. The use of log

values is in part due to the simpler expressions often available when using the sum of log-likelihoods rather than the product of true values, and partly to avoid numerical problems, since the likelihood values required tend to have a very high dynamic range and therefore fall outside the range available using standard floating point representation.

One of the simplest approaches is to maximize the hyperparameter log-likelihood. This provides us with a single set of hyperparameter values, so we do not need to make any changes to our model. This is also easy to achieve, since we can usually obtain closed forms for the derivatives of the hyperparameter values. However, this is only a single point approximation, while the operation we truly wish to carry out is marginalization over $p(\Theta \mid X, Y)$,

$$\begin{aligned} p(y \mid x, X, Y) &= \int_{\Theta} p(y \mid x, \Theta, X, Y) p(\Theta \mid X, Y) \mathrm{d}\Theta \\ &= \int_{\Theta} p(y \mid x, \Theta, X, Y) \frac{p(Y \mid \Theta, X) p(\Theta)}{p(Y \mid X)} \mathrm{d}\Theta \,. \end{aligned} \tag{2.28}$$

Unfortunately, this expression does not have a closed form, so we must approximate this result, and any subsequent quantities derived from our model predictions, using quadrature. A common choice in Bayesian optimization literature (e.g. Snoek et al. [2012]; Swersky et al. [2013, 2014]; Hernández-Lobato et al. [2014]) is to use slice sampling[Neal, 2003] to draw a set of hyperparameter samples from $p(\Theta \mid X, Y)$. We have followed convention in this work, except in Chapter 6 where we propose and test an alternative approach with reduced cost by making use of adaptive quadrature.

2.3.3 Derivatives of a Gaussian Process

A property of Gaussian processes that we will make considerable use of is that, in addition to the value of our function, the derivatives and integrals of a Gaussian process are also Gaussian processes.

Consider K_{YY}, the covariance matrix for observations $Y = f(X)$ made at three locations $X = [x_0, x_0 + hu_i, x_1]^T$ where u_i is a unit vector in the i^{th} direction and h is a scalar. This is given by

$$K_{XX} = \begin{bmatrix} \kappa(x_0, x_0) & \kappa(x_0, x_0 + hu_i) & \kappa(x_0, x_1) \\ \kappa(x_0 + hu_i, x_0) & \kappa(x_0 + hu_i, x_0 + hu_i) & \kappa(x_0 + hu_i, x_1) \\ \kappa(x_1, x_0) & \kappa(x_1, x_0 + hu_i) & \kappa(x_1, x_1) \end{bmatrix} . \tag{2.29}$$

We can use the property that an affine transform of a Gaussian distribution is also Gaussian to transform Y by the matrix

$$A = \begin{bmatrix} \frac{-1}{h} & \frac{1}{h} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.30}$$

into

$$\begin{aligned} Z &= AY \\ &= \begin{bmatrix} \frac{-1}{h} & \frac{1}{h} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f(x_0) \\ f(x_0 + hu_i) \\ f(x_1) \end{bmatrix} \\ &= \begin{bmatrix} \frac{f((x_0+hu_i)-x_0)}{h} \\ f(x_1) \end{bmatrix}. \end{aligned} \tag{2.31}$$

Then K_{ZZ}, the covariance matrix for the transformed observations Z, is then given by

$$\begin{aligned} K_{ZZ} &= A^T K_{YY} A \\ &= \begin{bmatrix} \frac{\kappa(x_0,x_0+hu_i)-\kappa(x_0,x_0)+\kappa(x_0+hu_i,x_0+h)-\kappa(x_0+hu_i,x_0)}{h^2} & \frac{\kappa(x_0+hu_i,x_0)-\kappa(x_0,x_0)}{h} \\ \frac{\kappa(x_0,x_0+hu_i)-\kappa(x_0,x_0)}{h} & \kappa(x_1,x_1) \end{bmatrix}. \end{aligned} \tag{2.32}$$

Considering the usual one-sided definition of a derivative

$$\frac{\partial f(x)}{\partial x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.33}$$

we can see that in the limit $h \to 0$ we have

$$Z' = \lim_{h \to 0} Z = \begin{bmatrix} \frac{\partial f(x_0)}{\partial x_0^i} \\ f(x_1) \end{bmatrix} \tag{2.34}$$

and

$$K'_{ZZ} = \lim_{h \to 0} K_{ZZ} = \begin{bmatrix} \frac{\partial^2 \kappa(x_0,x'_0)}{\partial x_0^i \partial x_0'^i} & \frac{\partial \kappa(x_0,x_1)}{\partial x_0^i} \\ \frac{\partial \kappa(x_1,x_0)}{\partial x_0^i} & \kappa(x_1,x_1) \end{bmatrix}, \tag{2.35}$$

where ∂x_0^i denotes differentiation with respect to the i^{th} element of x_0 and the use of $x'_0 = x_0$ denotes that differentiation in the upper left element of K'_{ZZ} is performed once with respect to each argument of κ. This illustrates that GPs and their derivatives are jointly normal, and that their covariance can be computed by differentation of the covariance function.

The above process can be repeated with further differentiation as necessary to find the covariance between derivatives of f of any order, and in any combination of axes, as long as the required derivatives exist and are tractable for the kernel

in use. We can therefore perform inference, draw samples from, and condition on observations of the gradient, Hessian, and higher order derivatives of the function being modelled by our GP. This is illustrated in Figure 2.1 where we have conditioned on observations of both the function value and gradient, and jointly drawn samples of the function value, gradient and Hessian conditioned on these observations. This is a particularly useful facility, which we will make considerable use of.

2.4 Bayesian Optimization

We now define the problem setting for Bayesian optimization and give an overview of current practice. For more detail, a recent review of the subject, including numerous applications, is Shahriari et al. [2016]. Many problems in Bayesian optimization have also been approached in the engineering design literature, we recommend Forrester et al. [2008] for an overview of this area.

We are concerned with optimizing expensive, unconstrained and possibly multimodal objective functions. Given a function $f\colon \mathcal{X} \to \mathbb{R}$, defined for $\mathcal{X} \subset \mathbb{R}^D$, which we will for this work take to be the hypercube $[-1, 1]^D$, we wish to find the minimum value of this function, $y_* = \min_{x \in \mathcal{X}} f(x)$, which we assume to occur at a unique location $x = x_*$ (If there is not a unique global minimizer finding any one is acceptable since we are ultimately only concerned with obtaining a low value, not the location at which this value occurs). Our motivation is the desire to minimize the cost f (or equivalently maximize performance) of some process with D controllable inputs represented by x. If the objective function was cheap to evaluate we could employ a range of established global optimization methods. Rios [2013] provides a recent review of such methods with performance comparisons. However, we are concerned with objectives that are expensive to evaluate. Both Snoek et al. [2012] and Garnett et al. [2010] demonstrate the optimization of problems in machine learning for which each evaluation can take minutes or hours, while Calandra et al. [2016]; Lizotte et al. [2007]; Tesch et al. [2011a] optimize the gait parameters of various robots, where each evaluation requires human intervention

to run an experiment. In such situations the thousands of evaluations commonly used by conventional methods are clearly impractical.

In order to achieve good optimization performance with only a small number of function evaluations we are willing to dedicate additional computational resources to the selection of each sequential evaluation point. At each stage of optimization we train a model $\mathcal{M}(X_i, Y_i, \Theta)$ where Θ represents the model hyperparameters and $X_i = \{x_0, x_1 \dots x_i\}$, $Y_i = \{y_0, y_1 \dots y_i\}$ are the sequence of locations and objective values up to the current iteration. We then define an acquisition function $\alpha(x \mid \mathcal{M})$ which provides some estimate of the utility of performing our next evaluation at x. This function is optimized over the search domain to find the predicted best location to perform the next evaluation $x_{i+1} = \arg\max_{x \in \mathcal{X}} \alpha(x \mid \mathcal{M}(X_i, Y_i, \Theta))$. We design this process such that performing this inner optimization is still cheap in comparison to evaluations of the true objective. Figure 2.2 shows four iterations of Bayesian optimization for a simple 1d function. While earlier response surface methods of optimization used a variety of function models (Jones [2001] reviews the use of some early non-GP models) the overwhelmingly common choice, following Kushner [1964], is the Gaussian process. Two exceptions to this rule, motivated by a desire to avoid the cubic scaling of the GP are Hutter et al. [2011], where a random forest is used, and Snoek et al. [2015] who demonstrate use of a deep neural network. The Gaussian process choice has the useful facility of allowing us to accommodate noisy objectives by including an independent noise component in our kernel.

The model parameters, Θ, must be given appropriate treatment to obtain a final value of the acquisition function. The simplest choice would be to make use of maximum likelihood values of these parameters. However, as noted above, marginalization is preferable. We have chosen throughout to perform marginalization on the GP predictions before then computing acquisition functions using this single posterior prediction. Since the alternative choice of performing marginalization after separately computing the acquisition function using each hyperparameter sample would incur substantial additional overheads, we believe our choice is the preferred option in this setting.

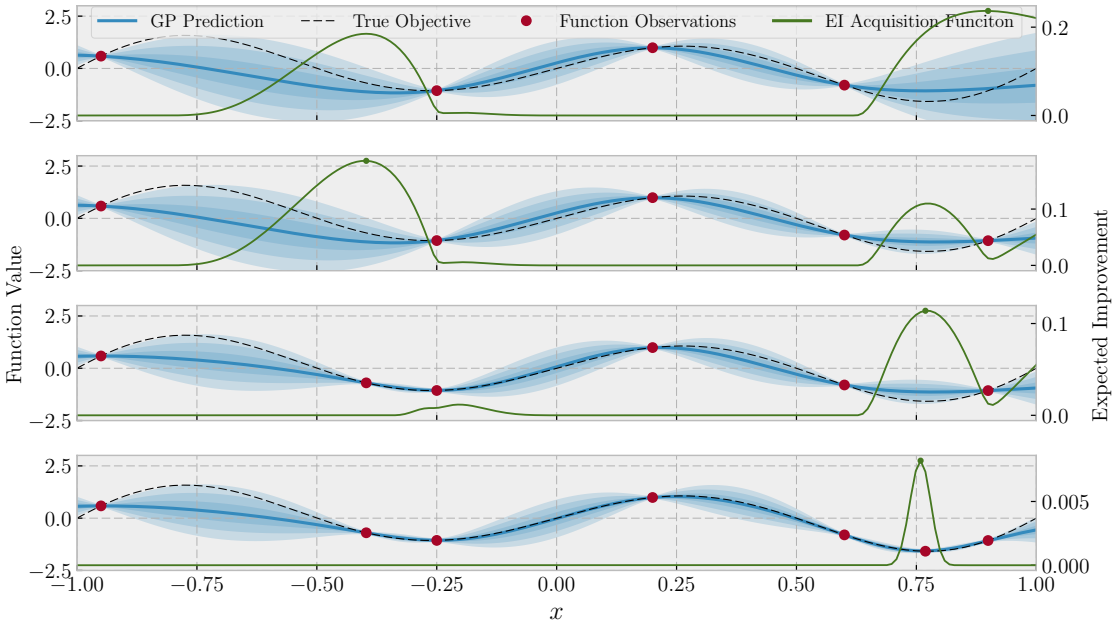

Figure 2.2: Four steps of Bayesian optimization of a one dimensional objective, $y = (1 + x^2)\sin(2\pi x)$, using the Expected Improvement acquisition function and the Matérn 5/2 kernel. At each new step the objective has been evaluated at the location that maximized the acquisition function in the previous step. In this manner the optimization has very quickly identified the global minimum.

It is important to note that the policy of always evaluating the objective at the maximizer of the acquisition function does not necessarily lead to ever evaluating the objective at the minimum of the posterior mean. While this may be correct behavior in an online setting, where we must continue to balance exploitation of the predicted minimum with exploration of areas with greater uncertainty, in many real settings optimization must terminate and provide a final recommendation. In the absence of any restrictions on permissible locations to recommend, and assuming that the objective has been defined as linear with respect to our utility over ouput values, this final recommendation should be the location at which the expected value of the objective is minimized. To show the online performance of Bayesian optimization we have always, unless otherwise noted, reported the immediate regret (IR) (the difference between the true value of the objective at the current predicted posterior mean minimum and the true global minimum, $f(\arg\min_x \mathbb{E}_{p(y|X,Y,x)}[y]) - f(x_*)$) at each iteration.

For many acquisition functions if Bayesian optimization is used from the very first iteration, this can lead to poor behavior, with only the objective only being evaluated at the corners of the domain. It is therefore common in implementations to evaluate the objective at a small number of points according to a initial design before Bayesian optimization is used, providing an initial dataset from which to learn reasonable hyperparameters values. Random samples (preferred by e.g. Snoek et al. [2012]; González et al. [2016b]; Klein et al. [2017]) or a Latin hypercube design (preferred by e.g. Jones et al. [1998]; Osborne et al. [2009]; Tesch et al. [2011a]) are common choices of initial design. We have made use of 10 random samples throughout unless otherwise noted. There is a large body of literature concerned with selection of the acquisition function which attempts to balance *exploration*, evaluating in regions of high uncertainty in the hope of finding new low values, with *exploitation*, evaluating in known low regions in order to refine the exact value of a local minimum. We briefly review some common choices, and more recent developments, which we will make further use of.

2.4.1 Probability of Improvement

One of the simplest acquisition functions is the Probability of Improvement, introduced by Kushner [1964]. It is simply the location with the greatest chance of providing an improvement over some target value, τ, which is lower than the current best observed value $\bar{y}$

$$\begin{aligned} \alpha_{\mathcal{PI}}(x) &= p(f(x) < \tau) \\ &= \Phi\left(\frac{\mu(x) - \tau}{\sigma(x)}\right), \end{aligned} \tag{2.36}$$

where $\mu(x)$ and $\sigma(x)^2$ are the GP posterior mean and variance conditioned on X and Y at x. This method has the disadvantage that we must first choose τ, which can heavily influence performance. The approach preferred by Kushner [1964] is to begin a high value of τ, which provides exploratory behavior, then reduce this value in stages as the maximum value of the acquisition function falls to provide increasingly exploitative behavior. Jones [2001] suggest a policy of optimizing the acquisition

function for a range of values of τ, then identifying clusters from resulting points and performing one evaluation of the objective function for each cluster obtained.

2.4.2 Lower Confidence Bound

Another common method is the Lower Confidence Bound (Upper in the case of maximization)[Auer, 2002]. At each step we choose the location with the minimum value of a specified confidence bound on the function objective value

$$\alpha_{\mathcal{LCB}}(x) = \mu(x) - \beta\sigma(x)\,, \tag{2.37}$$

for some $\beta > 0$. This method is again simple to evaluate, and has the advantage that theoretical bounds on the regret exist [Srinivas et al., 2009]. However, we must again use our prior knowledge to select a suitable value (or schedule) for β in order to obtain a balance between exploration and exploitation.

2.4.3 Expected Improvement

The expected improvement acquisition function [Jones et al., 1998] instead of considering only the probability of observing an improved value y, uses the expected value of that improvement over the incumbent solution a

$$\begin{aligned}
\alpha_{\mathcal{EI}}(x) &= \mathbb{E}_{p(y)}\left[\max\left(0, (a-y)\right)\right] \\
&= \int_{-\infty}^{a} (a-y)p(y)\,\mathrm{d}y \\
&= (a-\mu(x))\Phi\left(\frac{a-\mu(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{a-\mu(x)}{\sigma(x)}\right) \\
&= \mathcal{EI}^{-}(\mu(x), \sigma(x), a)\,.
\end{aligned} \tag{2.38}$$

This acquisition function is frequently used, and has the advantage of providing an automatic exploration-exploitation trade-off without the requirement for a tuning parameter. We make frequent use of this acquisition function, denoting *EI-MAP* when the GP hyperparameters have been optimized, and *EI-HYP* when they have been marginalized over using slice sampling.

Scenario		Expected Improvement	
		Minimization	Maximization
Incumbent:	a	$\mathcal{EI}^-(\mu, \sigma, a)$	$\mathcal{EI}^+(\mu, \sigma, a)$
Proposal:	$\mathcal{N}(\mu, \sigma^2)$		
Incumbent:	$\mathcal{N}(\mu, \sigma^2)$	$\mathcal{EI}^+(\mu, \sigma, a)$	$\mathcal{EI}^-(\mu, \sigma, a)$
Proposal:	a		

Table 2.1: Correct form of expected improvement between an constant and a normal distribution for all combinations of incumbent/proposal and minimization/maximization.

We have defined $\mathcal{EI}^-$ as the expected improvement in terms of minimization. As an aside we also define the expected improvement for maximization

$$\begin{aligned} \mathcal{EI}^+(\mu, \sigma, a) &= (\mu - a)\Phi\left(\frac{\mu - a}{\sigma}\right) + \sigma\phi\left(\frac{\mu - a}{\sigma}\right) \\ &= \mathcal{EI}^-(-\mu, \sigma, -a)\,. \end{aligned} \tag{2.39}$$

These quantities concern the expected improvement of a new observation distributed as $\mathcal{N}(\mu, \sigma)$ over an exact incumbent a. If we are instead concerned with the improvement over an uncertain incumbent that would be obtained by a certain new exact observation we should use $\mathcal{EI}^+$ for minimization and $\mathcal{EI}^-$ for maximization as detailed in Table 2.1.

In both the EI and PI acquisition functions we have relied on the lowest observed value of the objective function as an incumbent value which we seek to improve upon. This approach is valid in a noiseless setting and will still yield reasonable performance in the presence of observation noise with standard deviation significantly smaller than the scale of the objective function, as long as we do not attempt to refine our solution below that noise threshold (we will see this effect in Figure 5.1). However, if observation noise is significant our objective function evaluations may yield values substantially below the true minimum obtainable value, which would lead to poor optimization performance if used without appropriate adjustment. In this case we may wish to either adapt the acquisition function to take this effect into account (Scott et al. [2011] use the expected improvement in the minimizer of the posterior mean to obtain a modified acquisition function that is equal to EI if observation

noise is zero), or use one of the following information based acquisition functions, which do not make use of an incumbent observed value.

2.4.4 Entropy Search

The above acquisition functions have been concerned with the mean and variance of the GP model at a single location. Both the Informational Approach to Global Optimization method[Villemonteix et al., 2009] and Entropy Search [Hennig and Schuler, 2012] are concerned with the global properties of the model, specifically the probability distribution of the location of the global minimizer $p(x_*)$ and choose very similar acquisition functions with the intention of causing this distribution to contract onto the true location of x_*. We detail here the Entropy search method, in which the acquisition function is defined as the expected change in relative entropy of $p(x_*)$ that will be observed on taking the next evaluation at x

$$\alpha_{\mathcal{ES}}(x) = H(p(x_* \mid X, Y)) - \mathbb{E}_{p(y|X,Y,x)}\left[H(p(x_* \mid X, Y, x, y))\right] , \tag{2.40}$$

where H is relative entropy from the uniform distribution.

Unfortunately this acquisition function does not have an analytic form: we can neither evaluate nor take draws from $p(x_*)$ directly, so several stages of approximation are required. First the continuous $p(x_*)$ is approximated by a discrete distribution on a set of support points X_{support}, which are generated from some proposal distribution $u(x)$. We would like this proposal distribution to be as similar as possible to the true $p(x_*)$ in order to gain a good approximation with a reasonable number of points. The authors suggest slice sampling over the distributions proportional to the expected improvement or a lower confidence bound as suitable methods for generating X_{support}. We can then evaluate the full Gaussian process posterior $\mu_{\text{support}}, \Sigma_{\text{support}}$ over this support set, giving us a multivariate Gaussian distribution.

There is again no closed form expression for the distribution of the minimizer of the now discrete distribution $p(x_{\text{support}*})$. The authors present a choice between approximating this using expectation propagation or Monte-Carlo samples, preferring

the former due to the availability of derivatives with respect to $\mu_{\text{support}}, \Sigma_{\text{support}}$, but noting that the latter will have better scaling for a larger number of support points.

Given this discrete distribution $p(x_{\text{support}*})$ we use $u(x)$ to estimate the true density $p(x_*)$, and hence approximate the Relative Entropy at the current iteration. For each evaluation of the acquisition function this process is repeated with an updated x_{i+1} (although we can reuse the support points) and marginalized over the value of y_{i+1} to estimate the change in Relative Entropy expected due to the new observation.

The above process is unfortunately quite time consuming, due to the requirement to calculate expensive numerical approximations to the desired distribution.

2.4.5 Predictive Entropy Search

Predictive entropy search [Hernández-Lobato et al., 2014] uses a similar acquisition function to Entropy Search with an altered form for faster evaluation. Taking advantage of the identity 2.14 we can write

$$\begin{aligned}\alpha_{\mathcal{PES}}(x) &= \alpha_{\mathcal{ES}}(x) \\ &= H(p(x_* \mid X, Y)) - \mathbb{E}_{p(y|X,Y,x)}\left[H(p(x_* \mid X, Y, x, y))\right] \\ &= H(p(y \mid X, Y, x)) - \mathbb{E}_{p(x_*|X,Y)}\left[H(p(y \mid X, Y, x, x_*))\right] .\end{aligned} \tag{2.41}$$

This identity states that the expected change in differential entropy of the distribution of the global minimum obtained by evaluating y at x is equal to the expected change in differential entropy of the distribution of the value of y at x given knowledge of the global minimum. This form of the acquisition function still requires some approximation, but rather than evaluating the differential entropy of a costly discrete approximation for $p(x_*)$ we now only need to evaluate the differential entropy of a univariate Normal distribution over y (which has a simple closed form given by Equation 2.17). This allows better convergence to the ideal acquisition function so performs better than the original Entropy Search in practice. We now only need to approximate the marginalization over $p(x_*)$ which is replaced with a summation over a finite number of draws.

$$\mathbb{E}_{p(x_*|X,Y)}\left[H(p(y \mid X, Y, x, x_*))\right] \approx \frac{1}{N}\sum_{i=1}^{N}\left[H(p(y \mid X, Y, x, x_*^i))\right] . \tag{2.42}$$

To obtain draws from $p(x_*)$ the authors make use of a random spectral parametrization of the kernel, following Rahimi and Recht [2007], to obtain a finite dimensional approximation of the GP model. Taking draws from this parametrization provides approximate samples from $f(x)$ which can be easily optimized to obtain samples of x_*

Given these samples of x_* we are next required to make an update to the GP model by conditioning on x_* This requires three updates to enforce a global minimum:

1. The gradient in each axis is zero at x_*.

2. The Hessian matrix is positive definite at x_*. This is approximated by exact observations of zero on off-diagonal elements and requirements for each diagonal element to be strictly positive.

3. The function value at x_* is less than at all other locations. This is approximated by requiring the value at x_* to be less than all the objective function values observed so far.

The value constraints on various derivatives can simply be appended to the GP data as detailed in 2.3.3. However, the inequality constraints cannot be exactly implemented so instead expectation propagation [Minka, 2001] is used to obtain a set of observations which when included in the GP data induce a posterior that has minimum KL-divergence from the required constraints.

As noted above, our chosen approach to hyperparameters is to use marginalization to provide a single moment-matched GP, which is in turn used to evaluate the acquisition function, and we take the same approach here. Samples of x_* are drawn using this single marginalized GP, and each is used to generate a single term of the summation in Equation 2.42 by updating the GP with the sampled x_*, as detailed above, then marginalizing over the same hyperparameter set to obtain $p(y \mid X, Y, x, x_*^i)$ for any desired x.

3
Principled Stopping

Contents

In this Chapter we introduce and demonstrate the performance of a new algorithm for Bayesian optimization: Bayesian and Local Optimization Sample-wise Switching Optimization Method, BLOSSOM. By developing a global regret criterion we are able to halt optimization in a principled manner, avoiding the

waste of unnecessary additional steps. Further, by making appropriate use of local optimization we can achieve better local convergence properties than are available using Gaussian process methods, allowing considerable improvements in performance.

3.1 Problem Motivation

3.1.1 Requirement for a Stopping Criteria

In the majority of work on Bayesian optimization the results are shown as having either a fixed number of iterations or, less often, up to a fixed computational budget. While this is clearly desirable for averaging over and comparing multiple repetitions of the same optimization, it is not desirable in practice as the number of steps to take is now an additional parameter that must be selected manually. This choice requires expert knowledge of Bayesian optimization to select a number that hopefully will be neither too small resulting in easy improvement being neglected, nor too large costing additional expensive evaluations for minimal gain. We are therefore motivated to seek an automatic stopping criterion.

Early stopping has been considered by Lorenz et al. [2015] in an application of Bayesian optimization to brain imaging experiments. They propose and test early stopping based on the Euclidean distance between consecutive evaluations of the objective, and based on the probability of improvement on the incumbent solution. Both of these criteria provide notable improvement on a fixed number of iterations. However both these criteria are strictly *local* quantities with no consideration of the GP model apart from the incumbent solution and proposed next location. The values must still be selected by the user so that optimization is not terminated undesirably early by an incremental exploitative step while regions of the optimization domain remain unexplored. We would prefer a stopping criterion which takes account of the full model of the objective, and which has a parameter more easily interpreted in terms of the expected difference between the proposed and true solutions.

3.1.2 Convergence Properties

Bayesian Optimization has excellent empirical performance in identifying the minimizer of multi-modal functions with only a small number of evaluations. Bull [2011] has shown that $\mathcal{O}(n^{-\frac{v}{d}})$ convergence, where v is the smoothness of the kernel and d is the dimensionality of the problem, can be achieved with a modified version of expected improvement, however the authors note that this is only applicable for fixed hyperparameters. We are not aware of any estimates on convergence for the empirically better performing PES. Furthermore, even if a theoretical guarantee is available, good convergence is not likely to be achieved in practice. Implementation details ensure that convergence is likely to be severely limited once regret has been reduced to more than a few orders of magnitude lower than the scale of the objective function. Firstly, we are not able to exactly maximize the acquisition function. Constraints placed on the number of evaluations available to the inner optimizer limit our ability to select evaluation points to a high degree of accuracy. This limit is also relevant to minimizing the posterior mean for our final recommendation. Secondly, even in a noiseless setting we must add diagonal noise to our covariance matrix to resolve numerical errors in performing the Cholesky decomposition, $K' = K + \sigma^2_{\text{jitter}} I$. This reduces the rate of convergence available to that of optimizing an objective with the noise level we have now imposed. As we cluster more evaluations closely around the minimum the conditioning of the covariance matrix degrades further since the values of the objective at locations close to each other will be strongly correlated, and we are considering a scenario without observation noise.

This second issue arises in computing the diagonal elements of the Cholesky decomposition, which are computed by subtracting a series of previously calculated terms of the decomposition from the corresponding diagonal element of the original matrix then taking the square root. If the matrix is poorly conditioned then roundoff errors (caused by using finite precision floating point numbers) can lead to attempting to take the square root of a negative number, and the decomposition fails. As illustration consider a matrix A (the covariance matrix of five equally

spaced points in $[0, 0.1]$ using the squared exponential kernel with unit parameters) and its Cholesky decomposition C:

$$A = \begin{bmatrix} 1. & 0.9996 & 0.9987 & 0.9971 & 0.9950 \\ 0.9996 & 1. & 0.9996 & 0.9987 & 0.9971 \\ 0.9987 & 0.9996 & 1. & 0.9996 & 0.9987 \\ 0.9971 & 0.9987 & 0.9996 & 1. & 0.9996 \\ 0.9950 & 0.9971 & 0.9987 & 0.9996 & 1. \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.9996 & 2.499 \times 10^{-2} & 0 & 0 & 0 \\ 0.9987 & 4.996 \times 10^{-2} & 8.834 \times 10^{-4} & 0 & 0 \\ 0.9971 & 7.484 \times 10^{-2} & 2.647 \times 10^{-3} & 3.823 \times 10^{-5} & 0 \\ 0.9950 & 9.961 \times 10^{-2} & 5.287 \times 10^{-3} & 1.527 \times 10^{-4} & 1.911 \times 10^{-6} \end{bmatrix}. \tag{3.1}$$

The matrix being decomposed is Toeplitz. However, as the Cholesky decomposition is computed from the upper corner down its elements have rapidly decreasing magnitude, and therefore become increasingly susceptible to numerical errors. An increased number of points in a given interval (or equivalently a decreased distance between the same number of points) worsens conditioning, while the addition of a white nose kernel to the diagonal of the matrix being decomposed improves conditioning.

The potential loss in performance due to diagonal noise is illustrated in Figure 3.1 where we can see that the minimum required numerical conditioning roughly corresponds to a plateau in optimization performance. We shall see in Figure 5.1 that the observed profile of a fast initial drop in log-regret followed by a substantially shallower section is characteristic of Bayesian optimization with observation noise. Furthermore, in Figure 3.8 we shall see that many orders of magnitude lower regret can be achieved with our implementation of the Hartmann-4D objective function. We therefore also desire to create an optimization routine which does not suffer from this ineffective exploitation property.

This convergence issue has been addressed by Mohammadi et al. [2015] who switch from Bayesian optimization to the covariance matrix adaptation evolution strategy (CMA-ES) algorithm [Hansen and Ostermeier, 2001] once a criteria on probability of improvement has been achieved. This provides excellent convergence on the objectives tested. However, the switching relies on a heuristic based on a

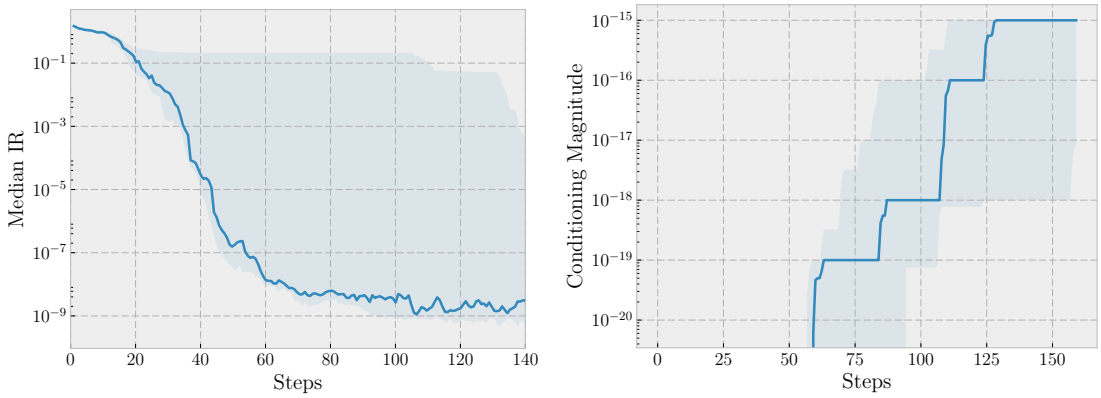

Figure 3.1: True immediate regret and numerical conditioning magnitude, σ^2_{jitter}, under optimization of the Hartman-4D function (defined in Appendix A) using the predictive entropy search acquisition function. The median and quartiles of 16 repetitions of the optimization are shown. Rather than adding a fixed diagonal component to the GP kernel matrix to ensure stability we have added the minimum value which still allows the Cholesky decomposition to succeed, in decade increments starting from $\sigma^2_{\text{jitter}} = 10^{-20}$. The optimization achieves a good result quickly, but then as jitter is added there is no further improvement beyond roughly σ_{jitter}. The reason that conditioning values significantly less than machine epsilon (the least value that can be added to 1, roughly 10^{-16} for 64-bit numbers) are observed is that although the values of Y have been normalized to have unit variance, we are using sampled hyperparameters. The sample with the smallest output scale parameter will therefore have much smaller values on the leading diagonal of the covariance matrix (if the smallest sample is 0.1 values on the leading diagonal will be 0.01) and therefore a smaller value than machine epsilon will still have an effect on conditioning. This effect allows lower than expected values to be present when jitter is first required, but is not expected to have any effect in later iterations when the learned distribution of hyperparameters is much tighter.

combination of previous values of the acquisition function and the number of steps without improvement of the incumbent. We desire to achieve a more Bayesian criterion for any changes in behaviour, to avoid the need for additional parameters which must be manually chosen by an expert. By using a non-stationary kernel which replaces the squared exponential form with a quadratic kernel in regions around local minima Wabersich and Toussaint [2016] also aim to achieve superior convergence. However they do not show the final value achieved by their method in most experiments, and the use of fixed pre-trained hyperparameter samples makes their implementation unsuitable for an online setting.

3.2 The Algorithm

3.2.1 Separating Global and Local Regret

In Bayesian optimization we aim to minimize the difference between the function value at the final recommended point, $y = f(\hat{x})$ and the value at the true global minimizer $y_* = f(x_*)$. This is the *regret* of selecting the current recommendation as the final solution. We shall now separate this concept into two distinct components which we treat separately. Let S be some region of note containing the incumbent solution. We define y_{in} as the minimum value of the objective within S and y_{out} as the minimum value outside S. We can then write

$$\begin{aligned}
\text{Regret}(x) &= \mathbb{E}_{p(y,y_*|x,X,Y)}(y - y_*) \\
&= \mathbb{E}_{p(y,y_*,y_{\text{in}}|x,X,Y)}(y - y_{\text{in}} + y_{\text{in}} - y_*) \\
&= \mathbb{E}_{p(y,y_{\text{in}}|x,X,Y)}(y - y_{\text{in}}) + \mathbb{E}_{p(y_*,y_{\text{in}}|x,X,Y)}(y_{\text{in}} - y_*) \\
&= \mathbb{E}_{p(y,y_{\text{in}}|x,X,Y)}(y - y_{\text{in}}) + \mathbb{E}_{p(y_{\text{out}},y_{\text{in}}|x,X,Y)} \max(y_{\text{in}} - y_{\text{out}}, 0) \\
&= R_{\text{local}} + R_{\text{global}} ,
\end{aligned} \tag{3.2}$$

where we have split the full regret into a local component, due to the difference between out candidate point and the associated local minimum, and a global component, due to the difference between the local and global minima. In the penultimate line we have replaced $y_{\text{in}} - y_*$ with $\max(y_{\text{in}} - y_{\text{out}}, 0)$, which will be of use later. The use of max is required when replacing y_* with y_{out}, since $y_* \leq y_{\text{in}}$ by definition but this inequality is not true for y_{out} . Both regret components are non-negative by definition, and we expect both to decrease as we learn about the objective. The local component represents the difference between our incumbent and the minimum within S. It is reduced by exploitation of the objective. The global component represents the difference between the local minimum and the global minimum. It will be reduced by exploration and also by exploitation of other local minima. There is a non-zero probability that $R_{global} = 0$ corresponding to the probability that the global minimum is in fact the minimum of the basin containing our incumbent.

3.2.2 Multiple Acquisition Functions

To address the issues identified above we split our optimization into four distinct modes intending to use the most effective at each iteration. The four modes are Random Initialization, Bayesian Optimization, Global Regret Reduction and Local Optimization.

Random initialization is as usual only required in the first few iterations, while Bayesian optimization using predictive entropy search is our default acquisition function if no relevant conditions to change behaviour have been satisfied. This provides the usual balance between exploration and exploitation.

In steps when a distinct candidate minimum can be identified we switch to the predominantly exploration strategy of Global Regret Reduction intended to reduce the global regret. By making this change we avoid the inefficient convergence of exploitation due to poor conditioning in Gaussian processes model when used for Bayesian Optimization. To identify a candidate minimum we require the existence of a region surrounding the minimum of the GP posterior mean within which the objective function has a high probability of being locally convex at all points..

Once the predicted global regret has fallen below some target value we use a purely exploitative Local Optimization algorithm. In this chapter we assume we have access to noiseless evaluations of the objective functions so we can employ a quasi-Newton local optimization routine, such as BFGS [Nocedal and Wright, 2006], which delivers super-linear convergence to the local minimum and is free of the numerical conditioning problems present in Gaussian processes. This comes at the cost of taking approximate gradient evaluations, each requiring D separate evaluations of the objective function. However, we note that since we are starting our local optimization very close to the minimum, (in fact we will detail in Section 3.3 that we choose to start only when the GP model predicts a region within which the objective is locally convex), only a small number of steps should be needed to achieve any required local regret target. We are then able to stop optimization, having utilized the less evaluation efficient local optimization algorithm to achieve

a target total expected regret lower than that which could have been achieved with a GP approach alone..

We now give further detail on the two new modes of optimization used by BLOSSOM. The methods used share many expensive computations with PES, so by reusing these results we do not incur too large an additional overhead.

Global Regret Reduction

Once a region around the posterior mean minimum has been identified within which local optimizations are likely to converge to the same location we do not wish to perform exploitation within this region with Gaussian processes, as this leads to numerical conditioning issues and therefore does not use evaluations efficiently. Instead we wish to set this region aside for pure exploitation under a local search strategy at a later stage. We therefore direct our efforts towards reducing the probability of any other local minima which might take lower values than our incumbent solution existing, reduction of the global regret. We use a modified form of expected improvement to achieve this, where instead of taking improvement with respect to the lowest observed objective value we compare to the estimated value of y_{in} that would be obtained by starting a local optimization from the posterior mean minimum. The acquisition function used is therefore

$$\alpha_{GRR} = \mathcal{EI}^{-}(\mu, \sigma, \mathbb{E}(y_{\text{in}})) \tag{3.3}$$

where μ and σ^2 are as usual the GP posterior mean and variance, and y_{in} is the minimum value within a defined region around the posterior mean minimum.

Local Optimization

Once we have a sufficiently high certainty that the GP posterior mean minimum is close to a local minimum of the objective, and that the expected difference between the value of this minimum and that of the global minimum falls below an acceptable threshold, we wish to exploit fully.

We use the BFGS algorithm for local optimization. This is a second order algorithm which updates an estimate of the Hessian at each step. By using our

estimate of the Hessian available from the GP model as the initial estimate in BFGS we hope to achieve convergence with fewer evaluations of the objective than otherwise. Rather than modifying the BFGS algorithm to use this estimate we rescale the problem so that the expectation of the Hessian is the identity matrix. With a local function model

$$f(x) = \frac{1}{2}x^T Hx + x^T g + c \tag{3.4}$$

we define $z = Ux$ where $U^{-T} = C$, $H = CC^T$ the Cholesky decomposition of H. This gives us a modified function

$$g(z) = \frac{1}{2}z^T z + z^T U^T g + c \tag{3.5}$$

as required. Once we have started this process we are no longer performing Bayesian optimization and can continue using BFGS until convergence. By selecting an appropriate stopping condition for local optimization we can ensure R_{local} falls below any desired target. We have selected a gradient estimate of less that 10^{-6} as our stopping condition, but any other method could be used.

3.2.3 Switching Between Acquisition Functions

We have specified that we wish to make decisions on the basis of the existence of a candidate minimum, and on the value of the global regret. In Figure 3.2 we show the decision making process used. However, we have not yet specified these criteria exactly. We choose to consider the existence of a sphere with non-zero radius, centred on the minimum of the GP posterior mean, $x_{\min}$, within which the objective function has a high probability of being convex at all points. Local optimization routines have excellent performance on convex functions, so if our model predicts a convex region surrounding a local minimum with high confidence we no longer desire our Bayesian optimization to recommend inefficient exploitative evaluations in this region. We therefore switch to the Global Regret Reduction acquisition function, which will place a high weight on exploration.

We have defined the global regret as the difference between the objective value at the local minimizer in some region and the true global minimum. We choose

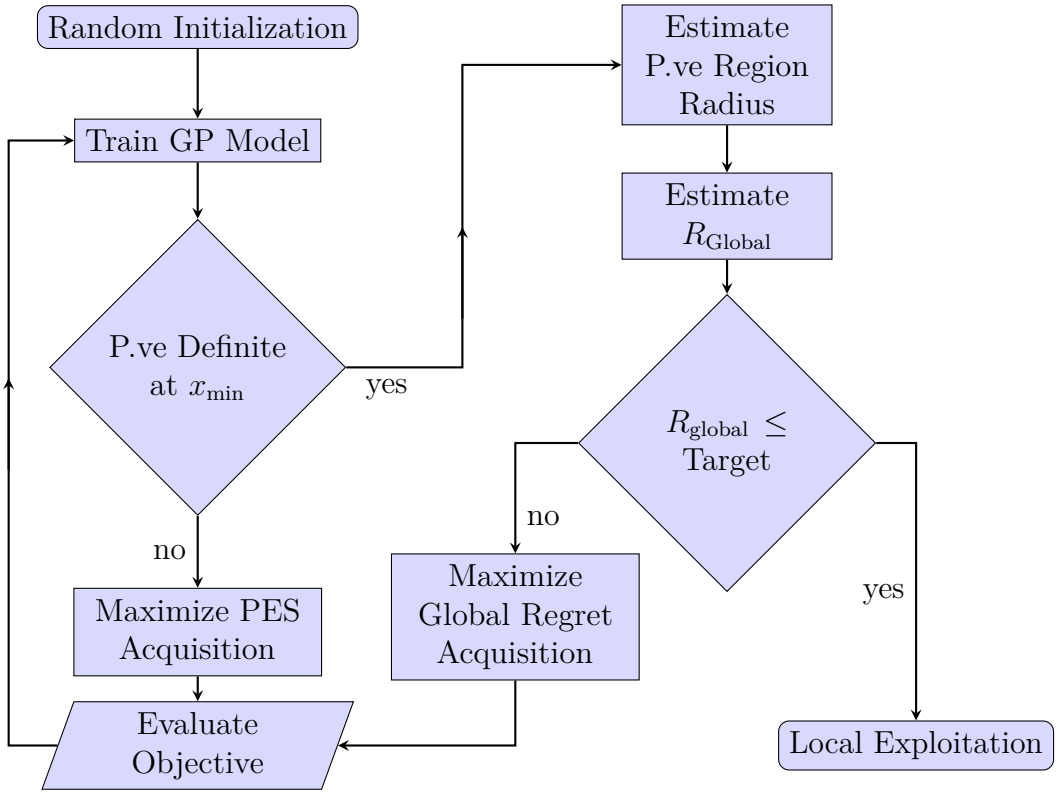

Figure 3.2: Flowchart for acquisition function switching behavior. Following initialization the acquisition function may switch between PES and Regret Reduction until R_{global} achieves a sufficiently low value. The final steps are then under the local exploitation strategy.

to use the same largest sphere centered on the posterior mean minimum within which the objective is believed to be convex used above to define this region. We can then obtain an estimate of Global Regret. If this estimate falls below our target value we move to the final stage of optimization and use Local Exploitation, otherwise we continue with the Global Regret Reduction. This process is illustrated in Figure 3.3 and detailed in Algorithms 1 to 4.

3.3 Estimating Required Quantities

We now detail the procedures used to estimate the quantities required in our switching criteria. We first detail our method of determining the size of the sphere within which the objective is believed to be positive definite, then provide a method for estimating the global regret and expected local minimum value.

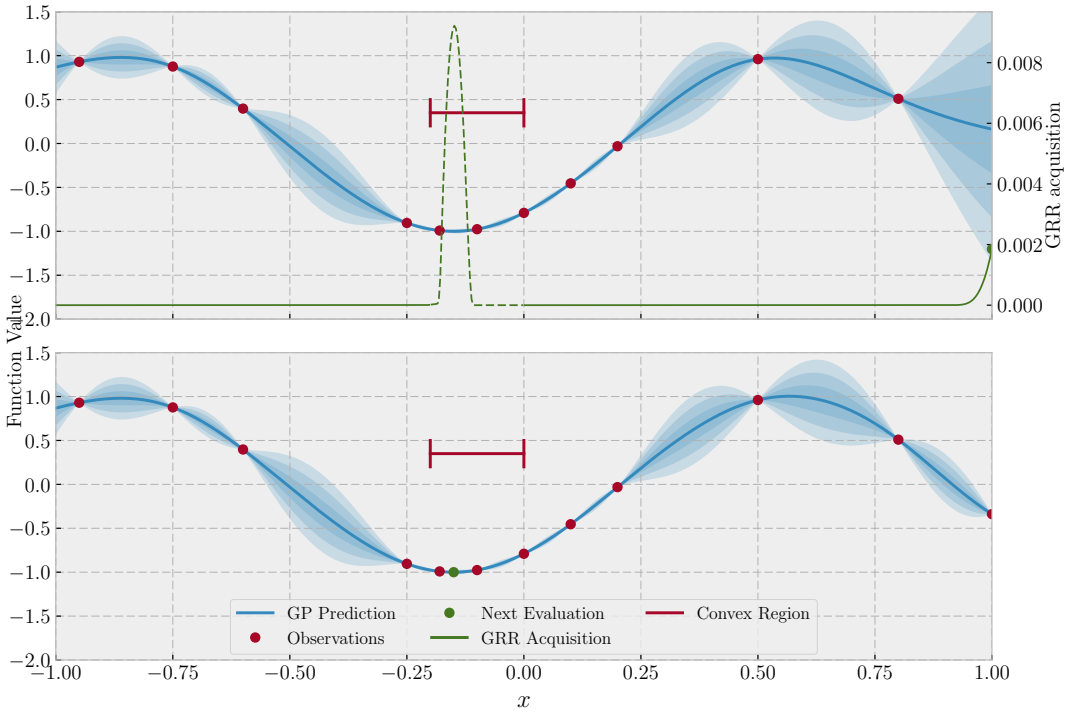

Figure 3.3: Two steps of BLOSSOM in an example problem. In the upper plot the GRR acquisition has a maximum at -0.15. However, since this falls within the convex region around the minimum of the posterior mean the next function evaluation is taken at the highest location outside this region. In the lower plot the estimated Global Regret is sufficiently low that no further Bayesian iterations are required. The next evaluation is the start of a local optimization from -0.15.

3.3.1 Identifying a Convex Region

Convexity is characterized by the Hessian matrix of the objective being positive definite at all points. For a matrix to be positive definite we require

$$x^T H x > 0 \tag{3.6}$$

for all x. Given a Gaussian process model of the objective we would like to construct:

1. a method to determine, using our GP model, the probability that the Hessian is positive definite at any point; and

2. a method to determine, using our GP model, the largest region centred on the current posterior mean minimum with a required probability of being convex at all points within that region.

Convexity at a Point

We make use of the Cholesky decomposition to determine if a matrix is positive definite. A unique real solution to the Cholesky decomposition of a matrix only exists if the matrix is positive definite. Implementations of the routine commonly return an error rather than computing the complex solution. We can make use of this behaviour to provide a test for positive definiteness returning a binary result, $PD(H) : \mathbb{R}^{\frac{d(d+1)}{2}} \to \{0, 1\}$, where d is the dimensionality of the problem.

Since under a Gaussian process model there will always be non-zero probability over all real values of inferred quantities we can never have certainty of positive definiteness. We therefore wish to determine the probability that the Hessian of our objective at some point x is positive definite (or if the point is on the boundary of the domain the Hessian for the remaining dimensions) under our GP model. All elements of the Hessian, H, have a joint normal distribution $p(H \mid x, M) = p(H_{\text{flat}} \mid x, M) \sim \mathcal{N}(\mu_H(x), \Sigma_H(x))$ where H_{flat} is the vectorized elements of the upper triangle of the Hessian matrix (the lower triangle is not needed since the Hessian is symmetric), and $\mu_H(x), \Sigma_H(x)$ are determined by the full GP posterior over H_{flat} at x. The probability of positive definiteness at x is then

$$\begin{aligned} p(PD(H) = 1 \mid x, M) &= \int p(PD(H) = 1 \mid H) p(H \mid x, M) dH \\ &= \int PD(H) p(H \mid x, M) dH \\ &= \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} PD(h_i) \end{aligned} \tag{3.7}$$

where M is our GP model and the h_i have been drawn from $p(H \mid x, M)$.

As our test of positive definiteness at a point, we require all of some n samples of H to be positive definite. We treat the positive definiteness of samples from $p(H \mid x, M)$ as Bernoulli distributed with parameter θ, since it is a deterministic binary output function of H. Conditioned on θ, the probability of k out of n samples of H being positive definite is

$$p(k \mid n, \theta) = \binom{n}{k} \theta^k (1 - \theta)^{n-k} . \tag{3.8}$$

Taking a uniform prior over θ the posterior distribution conditioned on n draws which are all positive definite is

$$p(\theta \mid n, k = n) = (n + 1)\theta^n \tag{3.9}$$

and the expected value of θ is

$$\mathbb{E}[\theta \mid n, k = n] = \int_0^1 \theta p(\theta) d\theta = \frac{n+1}{n+2}. \tag{3.10}$$

Passing our test for positive definiteness at a point, as described in Algorithm 1, can therefore be interpreted as determining that $\mathbb{E}[\theta] = 1 - \epsilon$ where $\epsilon = \frac{1}{n+2}$ while failure implies $\mathbb{E}[\theta] < 1 - \epsilon$.

Algorithm 1 Positive Definite Test

```
function PDatX(x, ε)
    G ← GP_model
    Hmean, Hvar ← G.infer_Hessian(x)
    PVEcount ← 0
    n ← ⌈1/ε − 2⌉
    for i = 1...n do
        h ← draw_Gaussian(Hmean, Hvar)
        h* ← Remove_Boundary_Elements(h, x)
        if Cholesky(h*) ≠ FAIL then
            PVEcount ← PVEcount + 1
        end if
    end for
    return PVEcount = n
end function


function remove_boundary_elements(h, x)
    # Utility function to drop rows and columns of the Hessian corresponding
    to axes where we are on the boundary of the optimization domain. (Positive
    definiteness is not required in these directions)
    Keep_Axes ← []
    for i = 1...D do
        if lower_bound[i] < x[i] < upper_bound[i] then
            Keep_Axes.append(i)
        end if
    end for
    return h[Keep_Axes][Keep_Axes]
end function
```

As an aside to this section we note that even at a minimum of our GP posterior, the probability that the Hessian is positive definite can be arbitrarily small, even though the mean function is convex. Consider a Hessian taking the form

$$H = \begin{bmatrix} 1 & a \\ a & 1 \end{bmatrix} \tag{3.11}$$

where a is Gaussian distibuted $\mathcal{N}(\mu_a, \sigma_a^2)$. Let $x_0 = [1, 1]^T$ and $x_1 = [1, -1]^T$. Substituting x_0 into Equation 3.6 we require $-1 < a$ while from x_1 we require $a < 1$. We can therefore pick any mean for a within $-1 < \mu_a < 1$ and the mean of H will still be positive definite. However, we have the freedom to select off-diagonal variance, σ_a^2, so can obtain any non-zero value we desire for the probability that $-1 < a < 1$. Since this is a necessary condition for positive definiteness we can force $p(PD(H))$ to be arbitrarily small.

Radius of a Convex Region

The method above allows us to effectively test a point for convexity. We now wish to use this function to find a convex region centred around the posterior mean minimum (again we exclude any axes on the boundary of the search domain). We choose to find the hypersphere centred at $\hat{x}$ with the greatest possible radius . As before we cannot obtain a certain value. Instead we find an estimate $\hat{r}_{max}$ which is the minimum distance over a finite set of test directions, u, to a point where our test for positive definiteness does not succeed.

We draw unit vectors, u, uniformly at random, by normalizing draws from the multivariate normal distribution $u = \frac{v}{|v|}$ where $v \sim \mathcal{N}(0, I_d)$. For each direction we obtain the positive definite radius

$$r_u(u) = \max_{PD\text{AT}X(\hat{x}+ru,\epsilon)=1} r \tag{3.12}$$

by performing a binary linesearch on r down to a resolution h_r. The first search is performed with the radius of the search domain as the upper limit, subsequent directions use the previous value of $R(u)$ as the upper limit and test the outer point

first, moving on to the next direction if this point returns a convex result. We thus obtain as our estimate of the radius of a convex spherical region centred $x_{\min}$

$$\hat{r}_{max} = \min_{u \in U} r_u(u)\,, \tag{3.13}$$

where U is our set of randomly drawn directions, which is in the minimum distance from $\hat{x}$ to the edge of the positive definite region out of $n_u = |U|$ random directions.

Algorithm 2 Positive Definite Sphere Radius

function PDBALL($\hat{x}, n_u, \epsilon$)
  $u \leftarrow$ `random_unit_vector`
  $x_{\text{edge}} \leftarrow$ `dist_to_domain_boundary`
  $\hat{r} \leftarrow \|x_{\min} - x_{\text{edge}}\|$
  for $i = 1...n_u$ **do**
    if $PDATX(x + \hat{R}u, \epsilon) = 0$ **then**
      $\hat{r} \leftarrow$ `binarysearch`$(u, \hat{r})$
    end if
    $u \leftarrow$ `random_unit_vector`
  end for
  return $\hat{r}$
end function

Algorithm 3 Global Regret Estimation

function GLOBALREGRETEST(GP, $\hat{x}, r$)
  $Y_{\text{in}}, Y_{\text{out}} = [], []$
  $Z \leftarrow$ `generate_support_points`(GP)
  for $i = 1...n$ **do**
    $Y_s =$ GP.`sample`(Z)
    Y_{in}.`append`$(\min_i Y_s[i]$ such that $|\hat{x} - Z[i]| < r)$
    Y_{out}.`append`$(\min_i Y_s[i]$ such that $|\hat{x} - Z[i]| \geq r)$
  end for
  $\mu_{\text{in}}, \sigma^2_{\text{in}} \leftarrow$ `mean`(Y_{in}), `var`(Y_{in})
  $R_{\text{global}} \leftarrow \sum_{i=1}^{N} \mathcal{EI}^+(\mu_{\text{in}}, \sigma_{\text{in}}, Y_{\text{out}}[i])$
  return R_{global}
end function

To obtain an estimate of the global regret we are required to marginalize over the values of the local and global minima, y_{out} and y_{in}, for this we will assume independence between these quantities. This assumption is not unreasonable, since the separation between local minima will be of similar magnitude to the characteristic

Algorithm 4 BLOSSOM

function BLOSSOM(f, R_{target})
  $X, Y \leftarrow$ `RandomInit`(f)
  while True **do**
    GP $\leftarrow$ `TrainGP`(X, Y)
    $\hat{x} \leftarrow$ `Mimimize`(GP.`posterior_Mean`)
    if PDATX($\hat{x}, \epsilon$) **then**
      $r \leftarrow$ PDBALL($\hat{x}, n_u, \epsilon$)
      $R_{\text{global}} \leftarrow$ GLOBALREGRETEST(GP, $\hat{x}, r$)
      if $R_{\text{global}} < R_{\text{target}}$ **then**
        break
      else
        $x_{\text{next}} \leftarrow \arg\max_x \alpha_{GRR}(x \mid \text{GP})$
      end if
    else
      $x_{\text{next}} \leftarrow \arg\max_x \alpha_{PES}(x \mid \text{GP})$
    end if
    $y_{\text{next}} \leftarrow f(x_{\text{next}})$
    $X, Y \leftarrow [X, x_{\text{next}}], [Y, y_{\text{next}}]$
  end while
  $\hat{x}_{\text{final}} \leftarrow \text{BFGS}(f, \hat{x})$
  return $\hat{R}$
end function

length of our GP kernel, significantly limiting their correlation (provided we have chosen a kernel function which is strictly decreasing with distance). Furthermore, in the context of a strictly positive valued kernel, such as the Matérn 5/2, any dependence between y_o and y_i will lead to a lower variance for $y_i - y_o$. Since this has a negative mean by definition, the lower variance will result a reduced probability of observing any positive value for $y_i - y_o$, and hence the expected global regret is reduced compared to the independent case. This makes our independence assumption an overestimate of R_{global} and thus not at risk of causing premature termination of optimization. Our estimate is therefore

$$R_{\text{global}} \approx \iint_{y_{\text{in}} \times y_{\text{out}}} \max(y_{\text{in}} - y_{\text{out}}, 0)\, p(y_{\text{in}}) p(y_{\text{out}})\, dy_{\text{in}}\, dy_{\text{out}}\,. \tag{3.14}$$

If we consider y_{in} to be well approximated by a normal distribution $\mathcal{N}(\mu_{\text{in}}, \sigma^2_{\text{in}})$ then we can perform the integral over y_{in}. This is simply the positive expected

improvement from y_{out} to y_{in}

$$\begin{aligned} R_{\text{global}} &\approx \int_{y_{\text{out}}} \int_{y_{\text{in}}=y_{\text{out}}}^{+\infty} \max(y_{\text{in}} - y_{\text{out}}, 0) p(y_{\text{in}})\, dy_{\text{in}}\, p(y_{\text{out}}) dy_{\text{out}} \\ &= \int_{y_{\text{out}}} (\mu_{\text{in}} - y_{\text{out}}) \Phi\left(\frac{\mu_{\text{in}} - y_{\text{out}}}{\sigma_{\text{in}}}\right) + \sigma_{\text{in}} \phi\left(\frac{\mu_{\text{in}} - y_{\text{out}}}{\sigma_{\text{in}}}\right) dy_{\text{out}} \\ &= \int_{y_{\text{out}}} \mathcal{EI}^{+}(\mu_{\text{in}}, \sigma_{\text{in}}, y_{\text{out}})\, p(y_{\text{out}}) dy_{\text{out}} \,. \end{aligned} \tag{3.15}$$

Since we do not have an analytic form for $p(y_{\text{out}})$ we cannot perform this integral. We instead approximate the marginalization over global regret as a summation

$$R_{\text{global}} \approx \sum_{j=1}^{N} \mathcal{EI}^{+}(\mu_{\text{in}}, \sigma_{\text{in}}, y_{\text{out}}^{(j)}) \,. \tag{3.16}$$

To evaluate this expression we can draw N samples from out GP model and find the value of y_{out} in each case. The process for this follows the same form as drawing global minima from the GP model used in §4.3, except that we must exclude support points within the convex region when we find the minimum of each draw. For this reason we draw half our support points using rejection sampling with the GP posterior variance as an unnormalized distribution. This ensures we still include high variance points outside the convex region even if no other minima are predicted by our model. The other support points are drawn using the more complex method in §4.3.3. To evaluate μ_{in} and σ_{in} we can use the same set of draws, considering this time only points within the convex region, to obtain a sequence of samples of y_{in}. Since we can easily generate a large number of samples of y_{in} directly from the GP posterior it is sufficient to use a maximum likelihood estimate of the mean and variance of these samples for μ_{in} and σ_{in}^2. As validation for out assumption of normality for y_{in} we show in Figure 3.4 the results of the Jarque-Bera test of normality [Jarque and Bera, 1980] using the actual samples of y_{in} when optimizing selected objective functions using PES. Since we are only claiming that $p(y_{\text{in}})$ is reasonably approximated by a normal distribution (and is expected to approach normal as our increased certainty about the location of x_* shrinks the region within which the y_{in} are located) we do not expect to pass the Jarque-Bera test, since even a small difference in skew and kurtosis will be detected with a reasonably large number of samples. However, a majority of test results fall below that of the normal

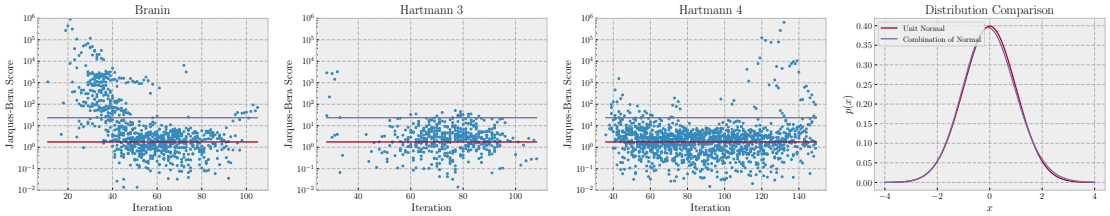

Figure 3.4: Jarques-Bera test score for each iteration where a local optimum has been identified over 16 optimizations of selected objective functions using PES. For each point shown 10000 samples of y_{in} have been used. Since the Jarques-Bera test correctly identifies that the distribution of y_{in} is not exactly normal for most points shown we have shown for comparison the score of the same number of samples (averaged over 20 repetitions) from two distributions: (1) the normal distribution with zero mean and unit standard deviation. (2) an equal mixture of two normal distributions with means $0.2, -0.2$ and standard deviations $1.1, 0.9$ respectively. As shown in the rightmost plot the pdf of these distributions differ by very little. Since the large majority of sample sets of y_{in} obtain scores falling below that of this second distribution, particularly if we exclude those from low numbered iterations, a normal approximation of y_{in} seems reasonable.

mixture distribution $p(x) = 0.5p(x \mid \mathcal{N}(-0.2, 0.9)) + 0.5p(x \mid \mathcal{N}(0.2, 1.1))$ which we have also shown. This distribution deviates from the zero mean normal distribution with unit variance by a maximum of ≈ 0.014, as illustrated in Figure 3.4 where the pdf of this distribution is separated from that of the unit normal by no more than twice the thickness of the plotted line, so we believe on this basis that our use of a normal approximation is reasonably justified, although further investigation into alternative distributions could very well yield some improvement.

3.4 Additional Detail

3.4.1 Alignment with the Local Principal Axes

Using the method described so far gives good performance on objectives with well conditioned Hessian at the global minimizer. However, identifying positive definiteness in narrow valleys not aligned with the axes performs poorly. Although each individual axis is quickly learned to have positive curvature the off-diagonal elements remain uncertain. We believe this behavior is due to the use of PES acquisition function, in which we wish to use the information gained by conditioning on positive definiteness of the Hessian at x_*, but instead approximate this impractical requirement by setting off-diagonal elements of the Hessian to zero while expectation

propagation is used to approximate a positive constraint on the diagonal elements. This different approximate treatment of diagonal and off-diagonal elements will result in a slightly different computed acquisition function than the ideal one, and may lead to the observed effect of slower learning of the value of the off-diagonal elements of the Hessian.

Since we have made no assumptions about the alignment of our objective we can transform our data (and the boundary of the optimization domain) by any rotation $X' = QX$ and evaluate our acquisition function on this altered dataset. For a rotation invariant kernel we can do this without loss of generality, in the more general case this will result in a change to the GP predictions, which we justify on the basis that we have at no point specified that the individual components of x are the principal directions of our objective, so any other choice of alignment is equally valid. In this case we choose at each step of optimization a rotation such that Q is the stacked eigenvectors in the eigendecomposition $H_{n-1} = Q^T \Lambda Q$, where Λ is diagonal and H_{n-1} is the Hessian of the posterior mean minimum from the previous step. Assuming the next step provides only a small update the Hessian at the minimizer will therefore be close to diagonal, which allows more effective optimization.

3.4.2 Approximating Small Values of Global Regret

By design, as the optimization progresses the distributions of y_{out} and y_{in} will contract and we become more certain that the minimum of the posterior mean corresponds to the true global minimum. Most terms in Equation 3.16 will be negligible, with only the few samples in which $y_{\text{out}} < y_{\text{in}}$ having a significant contribution. In the results shown in §3.5 the smallest R_{global} target used is 10^{-4} so this is not a problem. However, if we wish to continue to much smaller target regret values we shall require a better method of approximating the marginalization over y_{out}.

No closed form exists for the minimum distribution of a GP. Upper bounds on the tail of the minimum distribution do exist, and bounds on the tail distribution for a GP with non-constant variance are also available [Berman and Kono, 1989]. However, they are not particularly useful in this case as they apply to a GP with a

constant mean. To obtain a bound valid in our case we would therefore have to consider this constant mean to be the minimum of our posterior mean. The resulting bound would be on the minimum value of a GP for which every location has a 50% probability of being less than our posterior mean minimum. This will clearly give a bound far too wide to be useful in our application in which we have actively obtained new observations in order to eliminate locations with a significant probability of improving on our predicted minimum. Instead we use the existing sequence of draws of y_{out} as an empirical estimate of the cdf which is valid down to the minimum observed value, and extrapolate an upper bound on the tail of the distribution. We can then use numerical quadrature to perform the required marginalization.

Let σ^2_{Vmax} be the maximum posterior variance found in our GP model, and μ_{Vmax} be the posterior mean at that location, x_{Vmax}. The cumulative distribution function of the minimum value of a GP must be greater than the cdf of the value at any single point. The cdf of the objective function value at x_{Vmax} therefore provides a lower bound on the cdf of the global minimum. Further, since the cdf of a Gaussian distribution has a decay rate governed by its variance, and σ^2_{Vmax} is the maximum variance present, the cdf at all other locations must be less than that at x_{Vmax} for all y less than some value α. The cdf of the minimum value of the GP must therefore approach $\mathcal{N}(\mu_{Vmax}, \sigma^2_{Vmax})$ for small values. The cdf of any normal distribution with the variance σ^2_{Vmax} but mean value μ_{bound} which is less than μ_{Vmax} will therefore become an upper bound on the tail of the minimum distribution. We choose μ_{bound} such that the cdf of $\mathcal{N}(\mu_{bound}, \sigma^2_{Vmax})$ passes through the end of the empirical cdf found using draws of y_o

$$\mu_{bound} = \hat{y} - \sqrt{2\sigma_{Vmax}}\,\text{erf}(\frac{1}{N} - 1)\,. \tag{3.17}$$

This provides an estimated upper bound on the tail of the cumulative distribution, as illustrated in Figure 3.5. We can then approximate the global regret as the sum of contributions from samples and a numerical integral over the approximate tail

$$R_{global} \approx \sum_j^N \mathcal{EI}^+(\mu_{\text{in}}, \sigma_{\text{in}}, y^{(j)}_{\text{out}}) + \int_{-\infty}^{\hat{y}} \mathcal{EI}^+(\mu_{\text{in}}, \sigma_{\text{in}}, y) p(y \mid \mu_{bound}, \sigma^2_{Vmax}) \text{d}y\,. \tag{3.18}$$

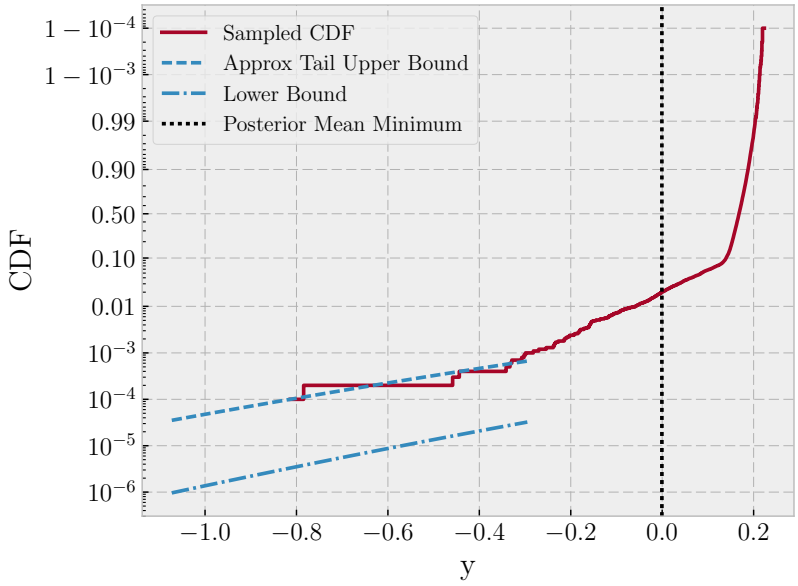

Figure 3.5: The CDF of the minimum value outside the positive definite sphere and bounds after 70 iterations on the Branin function. Since expected improvement takes negligible values for inputs greater than the incumbent a few tail samples contribute most of the expected global regret. The global regret estimated by samples is 8.00×10^{-4}, the additional component estimated by the approximate upper bound on the tail to the left of the least sample is 1.80×10^{-4}.

3.4.3 Incurred Overhead Costs

It might be expected that the addition of considerable additional complexity to already expensive Bayesian optimization would result in significantly increased overheads. The overheads in iterations requiring the evaluation of the expected global regret is indeed substantially increased. However, the additional work is often not necessary.

In each step we first find the posterior mean minimum, which is achieved using the same GP model that is already required for the acquisition function. This search requires only evaluations of the posterior mean, in contrast to the search of the PES acquisition function, which requires both the mean and variance from multiple models each conditioned on a different sample of x_*, so is comparatively cheap. We then test for positive definiteness at that location. If the test fails no further operations are necessary in this step so the additional overhead is negligible, as shown in steps 10 to 22 in Figure 3.6.

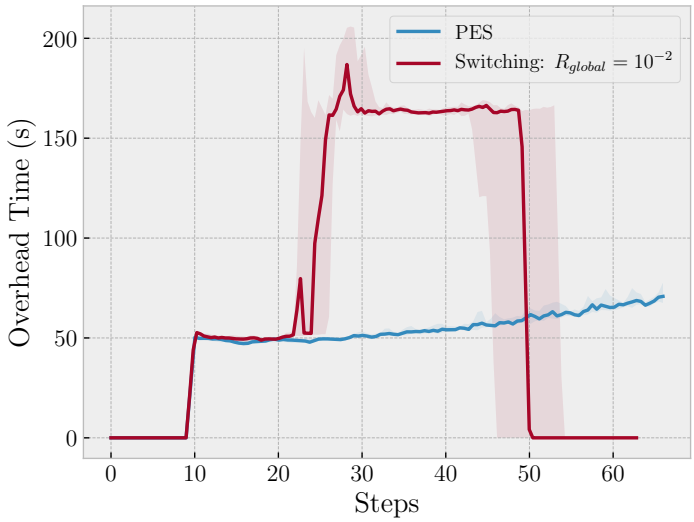

Figure 3.6: Per step overhead costs of our method in comparison to PES, measured single threaded using an Intel® Xeon™ E5 processor. The costs illustrated are for the Branin objective function and the median and quartiles of 16 runs with a global regret target of 0.01 are shown. While the steps in which an estimate of global regret is required incur significant additional costs those before do not, and the overhead in the final exploitative steps is negligible.

Furthermore, in the final steps before termination, which would otherwise be the most expensive due to the rising $\mathcal{O}(n^3)$ GP training cost, we perform pure exploitation using a local search, which incurs negligible overhead as illustrated from step 50 in Figure 3.6. The overall contribution is shown in Table 3.1, which displays the mean ratio of our optimization overhead to the mean cost the PES for the same number of iterations

$$\text{Overhead Ratio} = \frac{1}{M}\sum_{j=1}^{M}\frac{\sum_{i=1}^{N} t^{i,j}_{\text{stopping}}}{\sum_{i=1}^{N} t^{i,j}_{PES}}, \tag{3.19}$$

where $t^{i,j}_{\text{stopping}}$ and $t^{i,j}_{\text{PES}}$ are the overhead costs of the i^{th} iteration of the j^{th} run for BLOSSSOM and PES respectively. At worst, the overhead has been doubled, while eight of the twelve tests show a reduction in overhead costs. The overhead cost of our method therefore compares well to that of existing methods.

	Overhead Ratio	
	BLOSSOM 10^{-2}	BLOSSOM 10^{-4}
Branin	1.132	2.156
Camel 3hump	0.481	0.611
Camel 6hump	1.042	1.380
Hartmann 3D	0.769	0.961
Hartmann 4D	0.373	0.477
Hartmann 6D	0.654	0.750

Table 3.1: Mean relative overhead of our stopping implementation with PES taking the same number of steps in each indiviual optimization

3.4.4 Alternative Approaches to Determining Local Convexity

We have used the Cholesky decomposition of samples from our GP posterior over the Hessian of the objective function to determine local convexity, as this approach is trivial to implement and the cost of sampling and testing Hessians is determined only by the problem dimensionality so, in the low dimensional problems where Bayesian optimization is commonly used, is negligible compared to the Gaussian Process inference overheads already present. However, for two reasons it is worthwhile to consider alternative approaches.

- The Cholesky decomposition does, as discussed above, often fail to complete for matrices which are numerically very close to non-positive definite. This is not likely to be a problem in terms of reducing the radius of our positive definite sphere around a minimum at which positive definiteness is not marginal, but may be significant when considering an objective function which has a very poorly conditioned Hessian at the global minimizer.

- If we do wish to make use of our method in much higher dimension the computational overhead of sampling and testing a large number of Hessian matrices may become an issue. We are then motivated to seek a more efficient method of determining positive definiteness.

A natural alternative to the Cholesky decomposition for checking positive definiteness is to consider the eigenvalues of a matrix, which will all be positive if the matrix is positive definite. There is potentially more information available to us when considering eigenvalues since their values inform us how close to positive definite a matrix is, which cannot be achieved using the binary result of attempting a Cholesky decomposition. We might attempt to either directly consider the eigenvalue distribution given our GP prediction, or use the eigenvalues of sampled Hessians.

Unfortunately, while results on the distribution of eigenvalues are available, these tend to be in terms of the limiting distribution for large matrices, so are unsuitable for our low dimensional use case. Furthermore, the nature of our problem means we are concerned with matrices with elements each have individual normal distributions with full covariance between elements. The closest result we are aware of is that of Simon et al. [2006], which provides results for a positive definite matrix GG^T, where the elements of G are permitted to correlated and have non-zero mean.

Given our concern with low dimensional problems we might consider taking two and three dimensions as as a special case. We then have a closed form expressions for the eigenvalues of the Hessian using the quadratic and cubic formulas. However, since these formulas represent non-linear transforms of the coefficients of the characteristic polynomial of the Hessian, which in turn has coefficients determined by the sum of products of correlated Gaussian variables, we would only be able to obtain an approximate pdf for the eigenvalues. Since in these low dimensions our existing method remains very cheap we do not believe there is any advantage to be gained from this approach.

Moving on from considering the distribution of eigenvalues directly, we consider using eigenvalues obtained from samples of the Hessian. Evaluation of all eigenvalues for a matrix of size n requires $\mathcal{O}(n^3)$ operations, but we only require the sign of the minimum eigenvalue in order to check positive definiteness. This can be obtained in only $\mathcal{O}(n^2)$ via two uses of the power method, so there is some saving to be obtained for higher dimensional problems by directly replacing our check of the Cholesky decomposition with a check of the smallest eigenvalue. If we replace our

simple check of the number of samples which are not positive definite with some learned model for the distribution of the minimum eigenvalue we may also be able to obtain a given level of certainty with a reduced number of samples. However, since the power method is an iterative algorithm our reduced number of Hessian samples is obtained by replacing a fixed cost $\mathcal{O}(n^3)$ algorithm with an iterative one requiring $\mathcal{O}(n^2)$ per iteration. We do not expect that this scaling advantage compared to a single use of the Cholesky decomposition will be significant for problems of the dimensionality we are considering here.

One potential advantage of using eigenvalues in future work is that we obtain a numerical value for each dimension, rather than binary result if the matrix being tested is not numerically positive definite (By testing the Cholesky decomposition and eigenvalues of the covariance matrix of n equally spaced locations on $[0, 1]$ using the squared exponential kernel with unit parameters we find that the minimum calculated eigenvalue becomes negative at $n = 11$, the same value at which the Cholesky decomposition fails to complete). We are therefore able to set a small negative threshold allowing non-convexity in one and only one principal direction, while the equivalent approach of adding a diagonal component before the Cholesky decomposition would affect all principal directions. This provides a more nuanced approach with additional information compared to determining the positive definiteness of our model than the simple binary result of the Cholesky based test. We could also use eigenvalue and eigenvector information to identify directions in which the objective function has very little variation at the posterior mean minimum in order to inform optimization techniques such as Wang et al. [2016], which seek low dimensional objectives embedded in high dimensions.

3.5 Results

We compare BLOSSOM to expected improvement with the PI stopping criteria of Lorenz et al. [2015], and to PES using the acquisition function value as the stopping criterion. For each algorithm we test multiple values of the stopping criteria, shown in the legend as appropriate. Full detail of the objectives used is given in Appendix A.

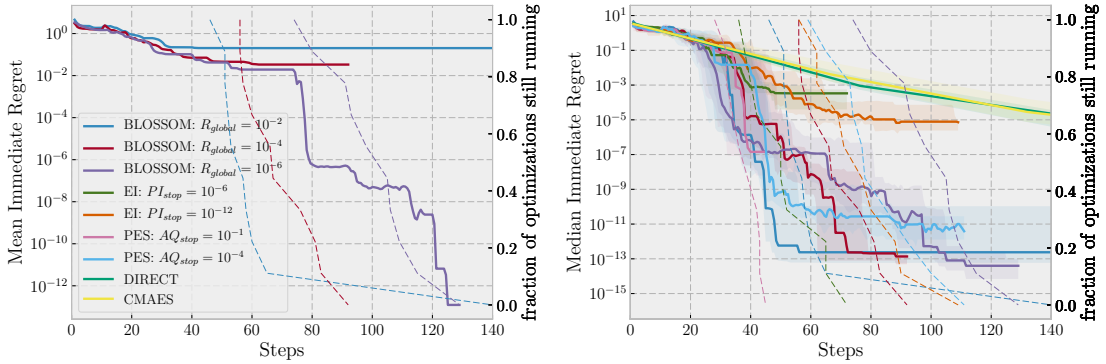

Figure 3.7: Comparison of median regret on draws from a GP corresponding to the Matérn 5/2 kernel in 2D (right), and illustration of the effect of changing our stopping parameter on the mean expected regret (left). Mean regret is heavily influenced by a few large values. Out of 35 runs there are those targeting 10^{-2} regret had 6 non-trivial results, targeting 10^{-4} yielded 4 non-trivial results, targeting 10^{-6} achieved the global minimum on every occasion while with no stopping condition only one failure occurred. Although with a limited number of the mean regret is not particularly close to the target values the decrease with more stringent stopping conditions is clear.

3.5.1 In-Model Objectives

To demonstrate the effect of changing the target value of global regret we make use of objective drawn from a GP, since the effect may not be observable using any single fixed objective. For example, the Branin function has multiple equal valued global minima. We will always achieve the global minimum, and the target regret only alters the number of steps required to terminate. We show in Fig 3.7 the mean regret over objectives drawn from a GP corresponding to the Matérn 5/2 kernel and note that we have achieved roughly the values we requested for expected regret.

3.5.2 Common Benchmark Functions

We now give results of our method on several common test objectives for global optimization, illustrated in Figure 3.8. In these tests we have transformed the objectives by $y' = \log(y - y_* + 1)$.This is unrelated to our contributions, and is done as many of the functions used take the form of a flat plain surrounded by steep sides many of orders of magnitude greater than the plain. This shape is extremely dissimilar to functions sampled from our GP prior using the Matérn 5/2 kernel, so yields very poor results. This is an ad-hoc transformation, and it would

be preferable to either use a kernel more appropriate to the objective or learn a transform of the output space online as suggested by Snelson et al. [2004].

Neither the number of steps taken nor the regret achieved is alone a useful metric for the effectiveness of a stopping condition (few steps with high regret are obviously undesirable, but also a small decrease in regret may not be worth a much increased number of steps), so in Table 3.2 we have also shown the mean product of steps and regret, $\mathbb{E}[nR]$. Equal contours of this metric take the form of $y = \frac{a}{x}$, so low values indicate improved performance in the sense that a halving of regret only represents an improved performance if this can be achieved using less than half the number of iterations. This metric has not been normalized to the objective function in any way, so comparisons across the rows of Table 3.2 ,between different methods operating on the same objective, are valid, but comparisons over the columns, between objectives that may have substantially different ranges, are not.

As is clear from the median curves in Figure 3.8, and mean final values in Table 3.2, we have been successful in our aim of achieving superior local convergence and early stopping. BLOSSOM achieves the lowest mean terminal regret, and mean product of regret and iterations, for five of the six test objectives. There is considerable disparity between the plotted and tabulated results for the Hartman 3D and 4D functions, particularly there are many orders of magnitude difference between the results for our method at different target regret values. However, we argue that this is in fact correct behaviour. These objectives are characterized by having multiple local minima with values greater than the global minimum. Usually Bayesian optimization will identify the correct basin as the global minimum and our local optimization converges to the correct value, as is evident in Figure 3.8. However, with some non-zero probability the GP will predict the global minimum and its surrounding positive definite region in the wrong location. If the estimated global regret is less than our target value when this occurs the solution is accepted, leading to exploitation of a local minimum and a high final regret. This occurs on several runs of our algorithm when using a value of 10^{-2} as the target global regret. When the lower target value of 10^{-4} is used additional

exploration steps are required to reduce the global regret estimate. These provide additional opportunities to correctly identify the basin of the global minimum. This leads to the much greater reliability observed in Table 3.2 at the cost of an increased number of objective evaluations.

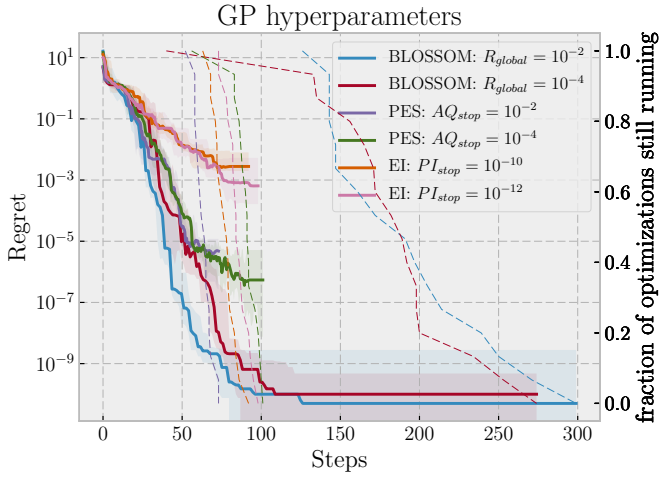

Figure 3.9: Comparison of stopping criteria optimizing the hyperparameter log-likelihood of a Gaussian process. Regret is shown with respect to the best value achieved by any single optimization, the median and quartiles of 16 repetitions of each method are shown. Our method consistently obtains several orders of magnitude better convergence than PES or EI at termination.

3.5.3 GP Hyperparameter Optimization

Optimizing model hyperparameters is a common problem in machine learning. We use BLOSSOM to optimize the input and output scale hyperparameters of a Gaussian process using 6 months of half hourly measurements of UK electricity demand during 2015.[1] As shown in Figure 3.9 we are able to obtain the best absolute result while terminating within a reasonable number of iterations, avoiding taking unnecessary further evaluations once the optimum has been achieved.

[1] www2.nationalgrid.com/UK/Industry-information/Electricity-transmission-operational-data/Data-explorer

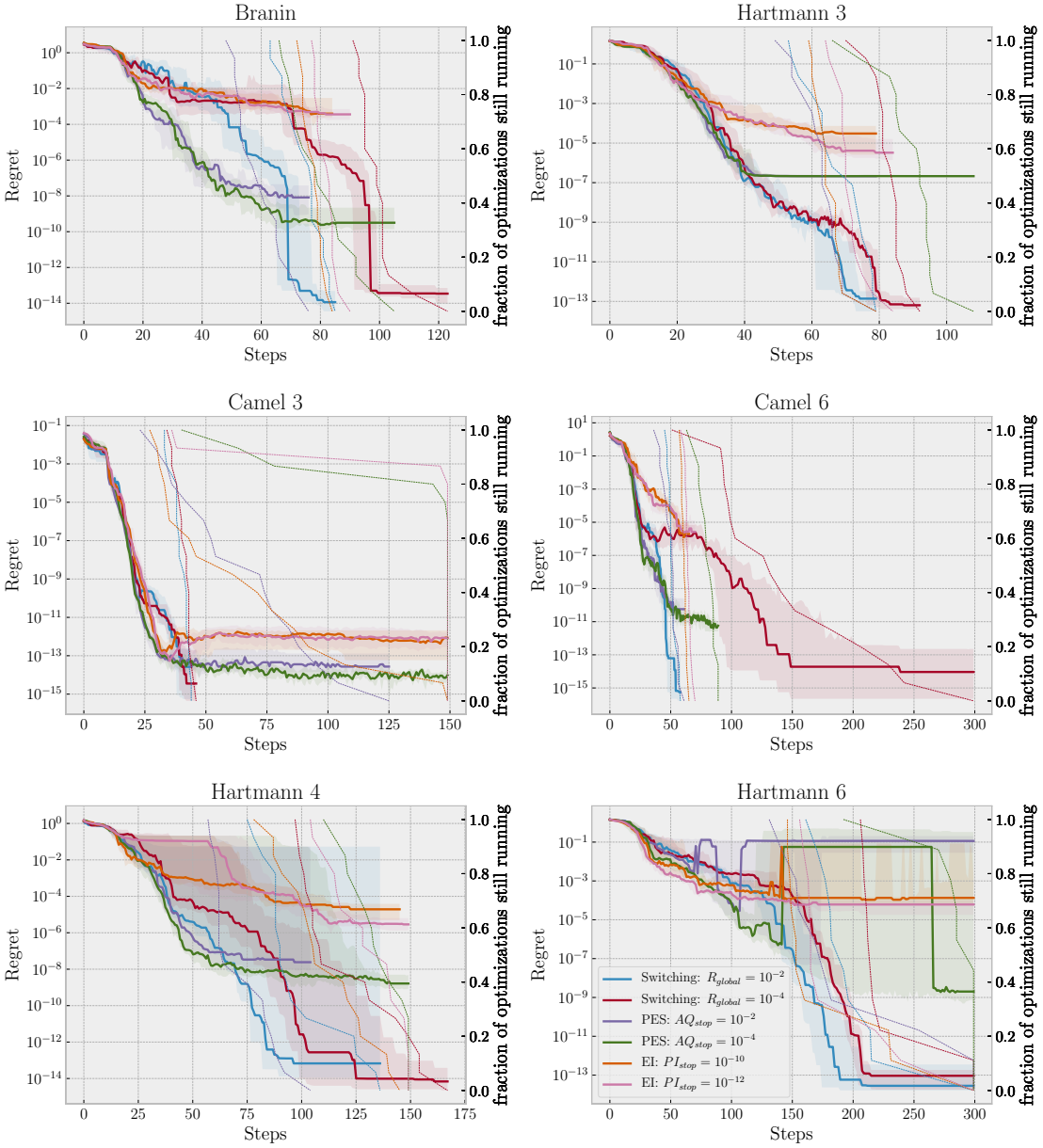

Figure 3.8: Comparison of various stopping methods. In order clockwise from top right: Hartmann 3D function, Camel 6-hump function, Hartmann 6D function, Hartmann 4D function, Camel 3-hump function and the Branin function. All stopping criteria allow an average saving compared to continuing to run optimization for many steps past convergence, but our method reliably achieves a low value before stopping. The median and quartiles of regret are shown. Fraction of optimizations still running after n steps are shown in thin dashed lines.

Objective	BLOSSOM 10^{-2}	BLOSSOM 10^{-4}	PES 10^{-8}	PES 10^{-10}	EI 10^{-10}	EI 10^{-12}
			Regret			
Branin	**3.32e-14**	5.2e-07	1.09e-07	2.9e-09	0.00221	0.00125
Camel 3hump	2.26e-13	1.79e-13	4.14e-13	**2.44e-14**	2.12e-12	1.57e-12
Camel 6hump	**2.28e-14**	7.95e-13	9.41e-11	1.62e-11	2.32e-05	2.35e-05
Hartmann 3D	0.107	**1.14e-13**	2.16e-07	2.16e-07	6.72e-05	0.000116
Hartmann 4D	0.0534	**5.21e-14**	0.0534	0.0133	6.06e-05	6.44e-06
Hartmann 6D	**0.00371**	0.0638	0.196	0.157	0.0229	0.0669
			Steps			
Branin	74.6	99.8	**59.6**	80.4	77.4	81.9
Camel 3hump	**39.6**	40.9	64.9	132	66.8	135
Camel 6hump	51.7	139	**49.8**	78.6	61.2	64.7
Hartmann 3D	67.8	82.6	**62.8**	89.4	65.1	71.1
Hartmann 4D	98.5	122	**72.3**	134	111	130
Hartmann 6D	199	230	196	281	**181**	190
			Steps × Regret			
Branin	**2.39e-12**	5.15e-05	5.69e-06	2.22e-07	0.167	0.103
Camel 3hump	8.56e-12	7.02e-12	1.18e-11	**2.73e-12**	1.07e-10	2.31e-10
Camel 6hump	**1.14e-12**	2.21e-10	4.84e-09	1.33e-09	0.00138	0.00157
Hartmann 3D	5.98	**9.41e-12**	1.35e-05	1.93e-05	0.00422	0.00746
Hartmann 4D	6.93	**5.89e-12**	4.36	1.95	0.00527	0.000853
Hartmann 6D	**1.11**	19.1	37.6	46.2	5.28	17.1

Table 3.2: Performance of selected stopping methods on various common objective functions. For the majority of objectives shown the number of iterations used by BLOSSOM is not substantially different then that used by other methods, so only a small difference in relative performance of the methods under comparison is exhibited between the `Regret` and `Steps` × `Regret` sections of the table. However, when optimizing the Camel 3-hump function BLOSSOM terminates after using only a fraction of the number of iterations used by other methods, leading to an improvement of approximately a factor of three compared to other methods when considering `Steps` × `Regret` rather than simply `Regret`.

4
Biased Variable Fidelity

Contents

In this Chapter we propose EnvPES, a new algorithm for Bayesian optimization, by adapting PES for use in the environmental variable setting. This new method has improved performance with respect to existing approaches to the problem, and more general application. Furthermore, we introduce and validate two variations of a new method for generating support points to be used in sampling the minimum of a Gaussian process. These methods have substantially lower costs than existing techniques and are of higher quality, drawing points from a distribution with greater similarity to the true distribution of the minimizer.

4.1 Introduction

In many applications there is an opportunity to make a coarse approximation to the true objective at a reduced cost. For example, in a setting where in each evaluation we undertake a number of physical experiments and use the mean result we could take a reduced batch size to obtain a quick but higher noise result. Similarly, any process using sampling to perform quadrature for some non-analytic integral could be run with a higher tolerance. In a physical modelling setting, such as using the finite element method to solve an ordinary or partial differential equation we could use a coarser mesh and larger time steps to obtain an approximate result. If our application is a machine learning process with a large volume of training data we could train with a reduced dataset which we then expect to result in a model with poorer performance. These cheap evaluations are hopefully at least reasonably well correlated to a full-cost evaluation with the same inputs, and so could be used to inform us about our true objective at a reduced cost if we make an intelligent choice of this parameter at each iteration. In many of the above examples the true objective is in fact defined as the limit case of infinite cost, for example MCMC quadrature is only ever an approximation to the true value of the integral regardless of the number of samples taken. In this case an appropriate judgement of the cost that is necessary to obtain a useful evaluation is even more important. We have followed Williams et al. [2000] in calling the variable controlling this choice an environmental variable.

This problem has been considered with an environmental variable taking two discrete values by Forrester et al. [2007]. In this work the second cheaper form of the objective is considered to be very cheap, so space filling methods taking large numbers of evaluations are proposed for this setting. Swersky et al. [2013] make use of Entropy search to accommodate any finite number of environmental variable settings, while Klein et al. [2017] also extend Entropy search to allow a continuous environmental variable. This second method, FABOLAS, is targeted towards an environmental variable controlling the use of a subset of training data in Machine Learning applications. The authors also employ a kernel specifically encoding the decay of performance in the axis of the environmental variable. Applications in

which performance is improved over multiple passes through the training data are considered by Swersky et al. [2014]. The authors use a decay kernel to predict the final performance of the objective given performance after only a limited number of training epochs. An entropy based acquisition function is used to choose, at each iteration, between performing additional training for an existing model or starting afresh at a new location.

We build on these existing works by adapting predictive entropy search to the continuous environmental variable setting. In addition to the basic modification we also construct a more efficient method for drawing the support points used to sample the global minimum, reducing overhead costs. Furthermore, since with variable fidelity evaluations the number of iterations is flexible we can extrapolate the growth of the overhead cost. This prediction is then incorporated into our acquisition function. These changes allow us to make a significant improvement in optimization overheads compared to methods based on ES, yielding much better performance.

4.2 The Algorithm

4.2.1 Variable Fidelity Problem Definition

As usual, we desire to find the global minimum of some expensive to evaluate $f(x)$ with x defined in the hyper-rectangle $[-1, 1]^d$. However, we now have access to a more complex function with an environmental variable $f_v(x, s), s \in [0, 1]$ such that $f_v(x, 0) = f(x)$ and evaluations incur a cost defined by some function $c_{\text{ev}}(x, s)$ which is monotonically decreasing with s for any fixed x. Evaluations of the true desired objective thus incur maximum cost. We must therefore define an acquisition function over both x and s.

Following Klein et al. [2017], we treat $f_v(x, s)$ and $c_{\text{ev}}(x, s)$ as independent $d+1$ dimensional black box functions which we model using a GP (in fact since cost is strictly positive we model log-cost). Or acquisition function then takes the form

$$\alpha(x, s) = \frac{\text{Expected Information Gain}}{\text{Expected Cost}} \tag{4.1}$$

in order to obtain the most cost efficient information gain at each step.

4.2.2 Predicting Information Gain

We choose predictive entropy search as the basis for estimating the expected information gain. We require three modifications to the core algorithm in order to employ this method of estimation in the variable fidelity context:

- samples of x_* are generated only in the plane $s = 0$;
- the gradient and Hessian constraints (1 and 2 as described in §2.4.5) are employed only on the d directions of the true objective, not the s direction of the environmental variable;
- the constraint that $f(x_*)$ must be less than existing observations (condition 3 as described in §2.4.5) is not included.

In all three cases these changes are because we are only concerned with finding the minimum within the $s = 0$ plane of the true objective. The second statement is justified by considering a setting where the environmental variable controls dataset size. Performance is then expected to always be increasing with larger datasets, and the gradient of f_v in the s direction is never zero. The third change is valid because $f(x_*)$ is not required to be a global minimum over the full variable fidelity function since a low cost evaluation may return a result far lower than the true global minimum. In fact it is not even required to be a minimum at all as the gradient $\frac{\partial f}{\partial s}$ may be negative at x_*. Furthermore, we may not even have made any evaluations with $s = 0$, which we could constrain $f(x_*)$ to be less than. We do not believe that this ommision causes a significant loss of performance, since the contraints placed on the Hessian at x_* still place considerable emphasis on learning the value of $f(x_*)$. However, it would nonetheless be useful to consider other methods for enforcing $f(x_*)$ to be the minimum in $s = 0$. We could require $f(x_*)$ to be less than the posterior mean at x_*, or less than the objective function value at any other local minima. It is also because $f(x_*, 0)$ is not required to be a minimum with respect to s that we have not employed the decay kernel approaches of Klein et al. [2017] and Swersky et al. [2014].

4.2.3 Estimating Overhead Cost

In Bayesian optimization we usually consider that the overhead cost of optimizing the acquisition is negligible in comparison to the cost of evaluations of the objective function. In the variable fidelity setting this assumption may not always be valid, since the evaluation cost of a low fidelity observation may be arbitrarily small. We use a GP to learn the log-cost from observed values online.

However, if we consider only the evaluation cost in Equation 4.1 we may find that our acquisition functions selects only very low cost evaluations at each iteration. Although this may yield the most cost efficient information gain with respect to evaluation cost, taking a large number of low cost evaluations will cause our overheads to grow quickly, due to the cubic scaling of the GP model. We therefore include not the current overhead, but the predicted mean overhead cost from the current to final iterations in our acquisition function.

We model the overhead costs as a function of iteration number, $i = 1, \ldots, n$, independently of the objective and evaluation cost functions using the parametric form

$$c_i = c(i, \theta) = \theta_0 + \theta_1 i^{\theta_2} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \theta_3^2), \tag{4.2}$$

where each parameter θ_i is assigned broad independent Gamma priors. Given the sequence of overhead costs up to the nth iteration we can use Bayes rule to write

$$p(\theta \mid c_1, c_2 \ldots c_n) = \frac{\prod_{i=1}^{n} p(c_i \mid \theta) p(\theta)}{p(c_1, c_2 \ldots c_n)} \tag{4.3}$$

and then optimize the log-likelihood using a local second order optimizer (we use BFGS) to obtain the maximum a-posteriori estimate of the optimization overhead model

$$\theta_{\text{MAP}} = \arg\max_{\theta} \left(\prod_{i=1}^{n} p(c_i \mid \theta) \right) p(\theta) \,. \tag{4.4}$$

To estimate the expected mean overhead we also require knowledge of B_n, our total computational budget remaining at iteration n. We can then find

$$N^* = \max_{N \in \mathbb{N}} N \text{ s.t. } B_n \geq \sum_{i=n}^{N} \left(c(i, \theta_{\text{MAP}}) + c_{\text{ev}} \right) , \tag{4.5}$$

the number of iterations that we will be able to take given the proposed evaluation cost, and hence

$$\bar{c} = \frac{1}{N^* - n} \sum_{i=n}^{N^*} c(i, \theta_{\text{MAP}}) \,, \tag{4.6}$$

the mean overhead cost over the $N^* - n$ remaining iterations.

The expected cost term in Equation 4.1 is therefore $\bar{c} + c_{\text{ev}}(x, s)$. This choice of overhead estimate causes the algorithm to prefer slightly more expensive evaluations than using just the most recently observed overhead, particularly when a large number of evaluations remain. We find that this relatively simple online estimate of future overhead costs yields improved performance compared to using the most recent value. A more complex treatment of the impact of future overhead costs in the variable fidelity setting will be undertaken in Chapter 5.

4.3 Fast Draws from Pmin

As part of any Entropy based implementation we are required to make draws of the global minimizer x_*. In Hennig and Schuler [2012] a choice is presented between an expectation propagation method and taking draws from the GP posterior on support points produced by MCMC under a proposal distribution that is reasonably similar to $p(x_*)$with EI being the preferred proposal. The distribution implied by the GP expected improvement or by a lower confidence bound are the suggested proposals of Hennig and Schuler [2012] , while Villemonteix et al. [2009] make use of samples from the previous iteration to train a density estimator from which samples can be generated. An alternative method of optimizing samples drawn from a spectral approximation to the GP model is used by Hernández-Lobato et al. [2014]. We now specify two variations of a new method for drawing support points to be used in the second method of Hennig and Schuler [2012], which tends to have better similarity to $p(x_*)$ than EI or LCB (Since the LCB itself may be negative the proposal distribution is in fact exp(LCB)) and can generate support points with a much lower per-point overhead cost.

4.3.1 Motivation for a Faster Method

The method of drawing support points by slice sampling [Neal, 2003] on either the EI or LCB of the GP posterior is effective because these heuristics are considered likely to be high when $p(x_*)$ is high, so provide support points sufficiently similar to the sample from $p(x_*)$. However, evaluation of EI and LCB requires an $\mathcal{O}(n^2)$ matrix multiplication to perform inference on the GP posterior. Many sequential evaluations are required for each point generated by slice sampling. This number is increased when we subsample the generated samples due to correlation between consecutive values and discard a burn-in period as is common practice when using MCMC. This process is therefore a significant contributor to the overhead growth, which we would like to minimize.

We can reduce interaction with the GP model by only using it to determine the parameters of a fixed approximation to $p(x_*)$ from which we can take samples at a much lower cost. We propose two approximations to $p(x_*)$, and provide measures of their performance in comparison to the existing methods for both time and quality of approximation. Both are a combination of unimodal approximations fitted to the local maxima of $p(x_*)$. Our weighted Local Hessians (WLH) method is a simple Laplace approximation with approximate weightings and clipped within the optimization domain. This provides a very low overhead which is almost constant with respect to the number of points required, but at the cost of being a relatively poor approximation unless the uncertainty around the minimum is small, making the Laplace approximation valid. Our Minimum of Local Quadratic Models (MLQM) method takes a more complex approach by taking independent draws of a local Taylor approximation for each local maximum of $p(x_*)$ and properly handling the boundary constraints. This provides an approximation of similar quality to EI or LCB slice sampling but with substantially lower cost which is effectively fixed with respect to the number of sample points desired. Only the fixed initial cost is influenced by the size of the GP model.

4.3.2 The Method

We now detail our new methods for drawing support points. Rather than attempt to directly approximate the intractable $p(x_*)$ we construct a set of local models. We make these approximations at each of the local minima of our GP posterior mean, where the probability of the point being a minimum is at a local maximum.We then combine the draws from our local models to construct the full global approximation.

In both methods we first identify the unique local minima in the posterior mean by use of multi-start local optimizations. At each minimum we evaluate the mean and covariance of the posterior mean and derivatives. This initial step is the only interaction required with the GP model. For each of N local minima x_i^l where $i = 1 \dots N$ we then consider a quadratic Taylor series centred at the posterior mean minimum

$$f = \frac{1}{2} z^T H z + z^T g + c \tag{4.7}$$

where $z = x - x_i^l$. As optimization progresses new observations are made so that our uncertainty about the minimum decreases, therefore this second order approximation becomes more valid as the region believed to contain a minimum contracts.

In the WLH method we make the further assumption that $\mu_H \gg \text{diag}(\Sigma_H)$ for all elements, where μ_H and Σ_H are respectively the mean vector and covariance matrix of the stacked non-repeated elements of H. The Hessian is therefore treated as constant with all elements equal to the appropriate element of μ_H. The local minimum under this assumption is located at

$$\begin{aligned} \frac{\partial f}{\partial z} &= \underline{0} \\ z &= H^{-1} g. \end{aligned} \tag{4.8}$$

Since our belief on g is a Gaussian given by the GP posterior, and zero at x_i^l by definition, the posterior distribution for the local minimum under the Taylor approximation is

$$p(x_*^l) = \mathcal{N}\Big(x_i^l, H^{-1} \Sigma_g H^{-T}\Big), \tag{4.9}$$

where we have used the property that any linear transformation of a Gaussian, in this case by H^{-1}, is also Gaussian. We then combine draws from each separate distribution according to a weight vector w. The ideal w would be the respective probabilities of each local minimum being the global minimum. Unfortunately, the correct weights are intractable, even if we were to consider only the N posterior mean minima. To obtain a very rough approximation we consider the posterior distribution $y_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$ of our GP model at each x_i^l and set the unnormalized weight, $\bar{w}_i$, equal to

$$\begin{aligned} \bar{w}_i &= p(y_i < y_*) \\ &= p(y_i - y_* < 0)\,, \end{aligned} \tag{4.10}$$

where μ_*, σ_*^2 are the μ_i, σ_i^2 pair with the lowest value of μ_i. $y_i - y_*$ is therefore distributed as $\mathcal{N}(\mu_i - \mu_*, \sigma_i^2 + \sigma_*^2)$ and $\bar{w}_i$ can be evaluated using the normal cdf. We then obtain normalized weights using

$$w_i = \frac{\bar{w}_i}{\sum_i \bar{w}_i}\,. \tag{4.11}$$

This weighting has the desirable properties that the local minimum with the lowest expected value has the greatest weight, while for all others the weight is monotonically increasing with the expected value, and monotonically decreasing with the variance. We therefore can express the distribution we use to draw support points, $q(x)$, as

$$q = Z\left(\sum_{i=1}^{M} w_i \mathcal{N}_{\text{bounded}}(x_i^c, H_i^{-1}\Sigma_{gi}H_i^{-T})\right)\,, \tag{4.12}$$

where $\mathcal{N}_{\text{bounded}}$ is a Normal distribution clipped within the search domain and Z is the appropriate normalizing constant.

4.3.3 Further Improvements

In the more complex MLQM approach we aim to achieve a more accurate approximation to the local distributions and their relative weights by making use of the full covariance matrix between the function value and the elements of the Hessian matrix and gradient vector at each local minimum. This improved approximation comes at

the cost of a more involved process for generating draws, so incurs a slightly higher overhead than the simpler method. To generate draws from each local approximation we sample from the full GP posterior of the components of our Taylor approximation

$$\begin{aligned} a &= [c, g_0, g_1, \ldots, g_d, h_{00}, h_{01}, \ldots, h_{0d}, h_{11}, h_{12}, \ldots, h_{1d}, \ldots, h_{dd}]^T \\ &\sim \mathcal{N}(\mu, \Sigma)\,. \end{aligned} \tag{4.13}$$

We then find $x_i = x_i^l - H^{-1}g$. If this falls within the optimization domain and the sampled H is positive definite we use this as our draw. Otherwise we use a quadratic programming implementation to solve the more complex inequality constrained problem, where f must now be minimized subject to the constraints

$$\left[\frac{\mathbf{I}_d}{-\mathbf{I}_d}\right] x \leq \left[\frac{\mathbf{1}_d - x_i^l}{x_i^l - \mathbf{1}_d}\right] \tag{4.14}$$

which ensure the sample remains within the optimization domain. We then find the value of the objective predicted at x by our Taylor expansion $y_i = f(x_i)$. To obtain m support points we take m sample-value pairs from each of the N local approximations, and use that with the lowest predicted value in each case.

This process has effectively replaced our GP model with a minimum of quadratics model with each independent quadratic being matched with full covariance to the local minima of the GP posterior. We can then take exact draws from this model.

4.3.4 Validation of Support Point Methods

Since we do not have access to the true distribution $p(x_*)$ we compare our methods of generating support points by evaluating a set of quality metrics offline using a selection of optimizations of sample objectives. For each step of optimization we generate 1000 support points using each method and then take 10000 samples from the $\mathcal{GP}$ posterior on these points. We then evaluate four quality metrics:

KL-divergence The distribution of the posterior minimum when confined to m support points is an m-dimensional multinomial distribution, which, if the support points have been drawn from $p(x_*)$, will be uniform. Using a Dirichlet prior we can find the parameters which maximize the posterior likelihood of

this distribution conditioned on the observed draws. The KL-divergence from the uniform distribution to the multinomial with these parameters provides a measure of how similar our approximation distribution is to $p(x_*)$.

Unused points If we take 10000 samples from a discrete uniform distribution with 1000 values the expected number of values not occurring is less than one. If a method has a significantly higher number of unused points this indicates a large number of our support points have a very low value for $p(x_*)$, so are costing us additional computation time for no benefit.

Time Reducing execution time is of course our primary objective, and is measured single threaded using an Intel® Xeon™ E5 processor.

Rate of useful point production We define a useful point in a set of support points as one that is selected as the minimum at least one tenth as often as if all points were equally likely to be selected. Since a support set containing a single point which is always selected is not a useful set, we define the rate of production of useful points as $r = \frac{n-1}{t}$ where n is the number of useful points drawn.

We test our method of drawing support points from the GP posterior at each step of the optimization of a selection of objectives which have been optimized using the PES acquisition function. In addition to our two proposals and slice sampling over both EI and LCB we test points drawn uniformly at random to provide a baseline. We show the results on our chosen metrics averaged over each step, and over five repeated runs of optimization, up to 80 steps in Table 4.1. It is worth reiterating that the methods under consideration are generating points from a proposal distribution which we would like to be *similar* to $p(x_*)$. Samples from the GP posterior are then generated on these support points to approximate actual samples from $p(x_*)$, with each new sample requiring only a draw from the posterior multivariate normal distribution over the support points. By contrast methods such as the spectral approximation of Hernández-Lobato et al. [2014] directly generate

samples of $p(x_*)$, but at the cost of considerable additional overhead cost. To obtain samples using the spectral approximation method it is necessary to perform a global optimization of a draw from the approximate model for each sample of x_* required. We would therefore expect this method to have perfect performance in terms of the quality of the points produced when compared to the support point methods, but at the cost of a substantially lower rate of production.

As expected, uniform draws perform poorly, with high KL divergence and almost all points unused. Although this method is technically superior in both execution time and the rate of generation of useful support points, the requirement to take an order of magnitude more support points to obtain a similarly sized useful set would be prohibitively expensive since we must take the Cholesky decomposition of the GP posterior covariance matrix at these points. This additional cost would more than cancel out any gains.

Our methods are comparable to EI and LCB, winning in all but one tests. The more complex MLQM method usually obtains the highest quality support, with the simpler WLH placed second. In both execution time and rate of point production WLH has the advantage over MLQM, again with both methods comfortably ahead of EI and LCB, so this is the method of drawing support points that we have used for optimization unless otherwise noted.

4.4 Results

We compare our method (EnvPES) to expected improvement (EI), Predictive Entropy Search (PES) and FABOLAS. Our own GP code using the Matérn 5/2 kernel is used for PES and EI. The implementation of FABOLAS used is that provided by Klein et al. [2017] in the RoBO package.[1] We have modified this implementation to return the posterior mean minimizer, rather than the best value observed, to provide consistent recommendations, and further, to use the Matérn 5/2 kernel over the environmental variable rather than the parametric form used in the original (since the monotonic assumption does not necessarily hold in our

[1] https://github.com/automl/RoBO/

Objective	Uniform	EI	LCB	WLH	MLQM
	KL-divergence				
Branin 2d	4.53	3.44	3.67	0.425	**0.239**
Hartmann 3d	4.92	2.87	4.82	0.451	**0.262**
Draw 4d	1.75	0.854	1.8	**0.738**	0.85
Hartmann 6d	1.83	0.715	2.13	**0.627**	1.06
	Percent useful points				
Branin 2d	17.3	31.8	24.7	77.1	**91.0**
Hartmann 3d	13.9	36.1	17.1	73.3	**90.5**
Draw 4d	69.0	**89.5**	72.8	68.6	72.7
Hartmann 6d	66.7	**88.2**	65.8	77.9	70.6
	Time				
Branin 2d	*0.000184*	20.8	37.1	**0.489**	1.65
Hartmann 3d	*0.000216*	39.1	104	**1.27**	3.07
Draw 4d	*0.000248*	68.2	147	**2.93**	6.51
Hartmann 6d	*0.000309*	146	230	**8.98**	22.5
	Rate of Useful Point Generation				
Branin 2d	*1.24e+06*	34	12.1	**1.69e+03**	628
Hartmann 3d	*5.37e+05*	12.3	1.73	**745**	513
Draw 4d	*2.2e+06*	15.7	4.44	**255**	178
Hartmann 6d	*2.26e+06*	7.78	3.31	**131**	71.2

Table 4.1: Performance of methods of drawing support. Best performance shown in bold on each row, excluding uniform support which is included only as a reference.

experiments). We show that we are able to match or exceed the performance of existing methods over a selection of objectives. As before the objectives used are fully specified in Appendix A.

For initialization we follow Klein et al. [2017] by evaluating a set of values of the environmental variable for each random draw from x rather than each evaluation being independent. We choose $s = 0.5, 0.75, 0.875$ and use twenty total evaluations in the initialization dataset. It is worth noting that this is a more expensive choice then FABOLAS, which uses $s = 1/64, 1/128, 1/256$, so differences in performance early in the optimization process can be attributed to this substantial difference in initialization cost, rather than the acquisition function used. All times have been measured single threaded using an Intel® Xeon™ E5 processor.

4.4.1 In-model test

To illustrate expected performance we use draws from the Matérn 5/2 kernel as objective functions. The characteristic length parameter is set to 0.3 in each axis and the search domain is $[-1, 1]^2$. We model the environmental variable as an additional dimension of the objective over $[0, 1]$ with characteristic length l_{ev}, and the cost of evaluations as an exponential $\exp(-l_c s)$. We show results for advantageous and adversarial values of l_c and l_{ev} in Figure 4.1. This shows the potential gains of making use of the environmental variable in a suitable scenario, while retaining reasonable performance otherwise.

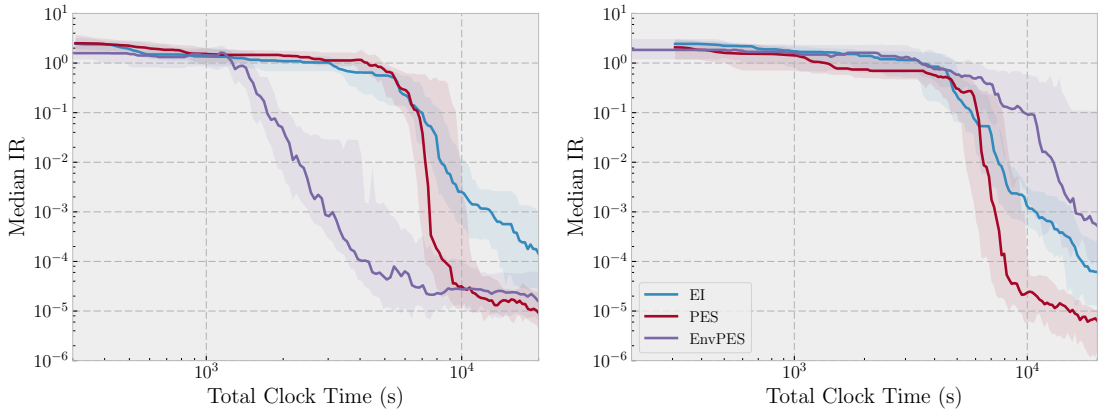

Figure 4.1: Performance of EnvPES (green), PES (red) and EI (blue) on draws from the Matérn kernel. The median (solid line) and interquartile range (shaded) of optimizing 40 draws are shown. Left: The objective and cost function had $l_c = 3$ and $l_{ev} = 1.5$ which allows very good performance as the cost decays quickly while the approximate objective remains very similar to the true objective. Substantial improvement is therefore available using less expensive evaluations. Right: The objective and cost function had $l_c = 1$ and $l_{ev} = 0.4$ which induces poor performance as the cost of evaluation does not reduce much and evaluations at increasing s are only loosely linked to the true objective. Evaluations cannot be made as cheaply compared to the full cost as the previous example and convey little information about the true objective. EnvPES is no longer able to perform as well as other methods since it needs to learn a more complex model.

4.4.2 Off model test

Common Synthetic Functions

We test our method on a selection of common objectives. The results are shown in Figure 4.2. Following the approach of Swersky et al. [2013] we use a linear shift of

the objective for the lower cost evaluations; in this setting the shift is continuous rather than discrete. The cost imposed is of quadratic form rising from two minutes as the cheapest up to thirty minutes for the full cost objective. Performance of EnvPES matches or exceeds the other methods shown on each test. Considering only the cost of evaluation, EnvPES and FABOLAS have similar performance, but, acknowledging the real time to optimize the acquisition function, the lower overheads of EnvPES allow better performance.

Gaussian Process Hyperparameters

Fitting the hyperparameters of a Gaussian process is a common machine learning problem, with evaluation cost that scales as $\mathcal{O}(n^3)$ with the number of datapoints. We show the performance of our method in this task using a selection of datasets using the Matérn 5/2 kernel. We transform the log-likelihood by

$$g(y, n_{\text{sub}}) = \begin{cases} -y\frac{n_{\text{sub}}}{N} & \text{if } y < -1 \\ 1 + \log(-y\frac{n_{\text{sub}}}{N}) & \text{else} \end{cases}, \tag{4.15}$$

which is monotonic, smooth and continuous with respect to the true log-likelihood but reduces the absolute value of large negative likelihoods and normalizes subsets to the value at the full dataset. This reduction in the dynamic range of the objective prevents a reduction in performance due to numerical conditioning problems previously discussed in §3.1.2. EI and PES are only able to evaluate the objective using the full dataset, while variable fidelity allows evaluations using between all and 2% of the full dataset in each case. In each experiment the total computational budget is limited to 24 hours.

Our first test, shown in Figure 4.3, optimizes the input and output scale hyperparameters for regression using the UK electricity dataset. On this two-dimensional problem, evaluation of the log-likelihood typically incurs a cost of around ten minutes when using the full dataset. Using EnvPES we are able to achieve to achieve low values much earlier than methods not making use of the environmental variable, and are faster in real-time convergence than FABOLAS due to reduced overheads.

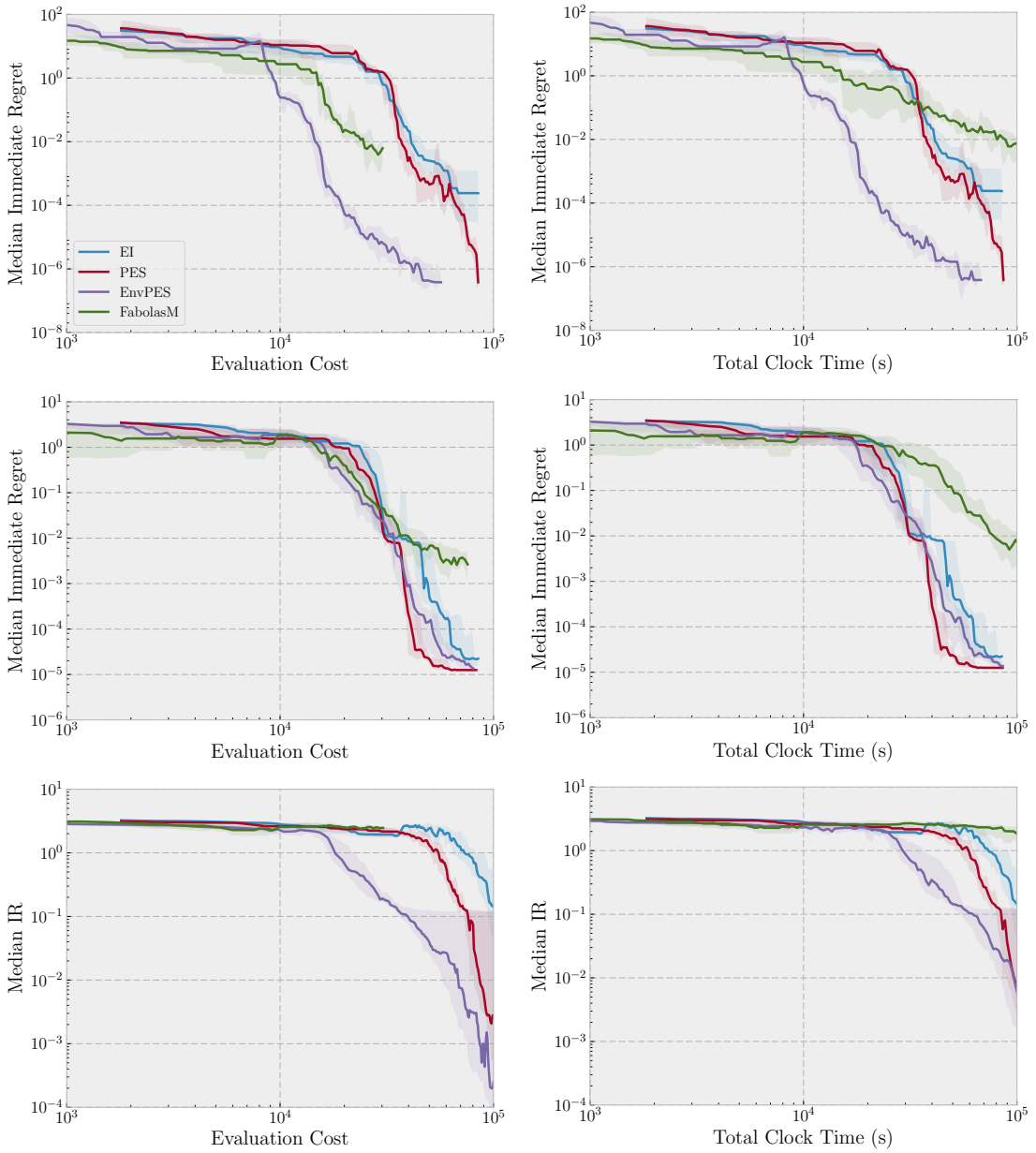

Figure 4.2: Performance of EnvPES, PES , EI and FABOLAS with modifications as noted on the Branin (top), Hartman 3D (middle) and Hartman 6D (bottom) test functions with less expensive evaluations available under a linear shift from the true objective. The performance is shown against evaluation time for the objective (left) and including overhead due to the acquisition function (right). The median and interquartile range (shaded) of 20 runs are shown. Note that while FABOLAS has good performance with evaluation cost the high overheads detract from this substantially when included in the cost.

Using the four dimensional combined cycle power plant dataset we show, in Figures 4.4 and 4.5 respectively, results for both the three dimensional optimization problem over output scale, diagonal noise, and a single isotropic lengthscale; and the six dimensional problem using an independent lengthscale hyperparameter for each axis. These objectives are relatively cheap in comparison to the optimization overhead, typically taking only 30 seconds to evaluate the full-cost objective. All methods quickly obtain very similar values, so we have expressed results in terms of regret with respect to the best single result obtained by any method for greater visual clarity. The improvement gained in evaluation cost alone shown in the leftmost plots has therefore not been translated to a performance improvement with respect to total cost. However, we have not obtained noticeably worse performance either, again showing that use of EnvPES in unfavourable conditions does not lead to significantly reduced performance.

Our final GP hyperparameter test uses the nine dimensional physiochemical properties of protein tertiary structure dataset with an isotropic Matérn 5/2 kernel, giving a three dimensional objective function which typically requires 50 minutes to evaluate using the full dataset. As we can see in Figure 4.6 we are able to achieve substantial improvement in both evaluation and total cost with this more expensive objective. In this objective, which is the most expensive and also has a cubic scaling of cost with the environmental variable, the advantage of the variable fidelity approach is clear, with good values being obtained only just after the first random location has been evaluated by full-cost methods.

Support Vector Machine Hyperparameters

For much larger datasets it is not feasible to use a Gaussian process for inference. Our first non-GP objective optimizes the parameters of a support vector machine used to perform classification of the popular MNIST dataset. We optimize the error penalty and kernel lengthscale hyperparameters for two dimensional objective function. We allow the dataset size to vary from 100 to 10,000, which incurs a cost of around five minutes at the maximum. As shown in Figure 4.7, we are able

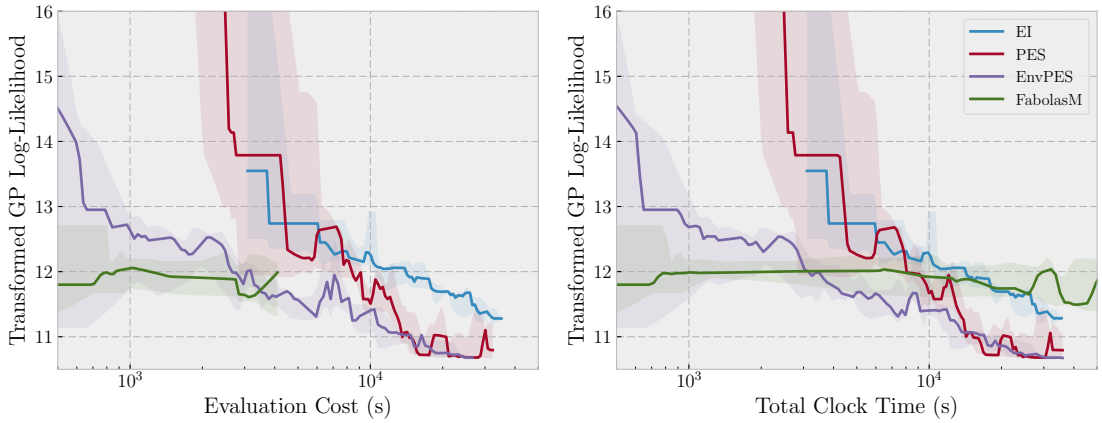

Figure 4.3: Performance of EnvPES, FABOLAS, PES, and EI minimizing the negative log-likelihood of kernel hyperparameters for a Gaussian process on UK power data. The median and interquartile range (shaded) of ten runs are shown. EnvPES and FABOLAS both achieve good performance in terms of evaluations cost, but only EnvPES is able to translate this into superior results when overhead cost is included.

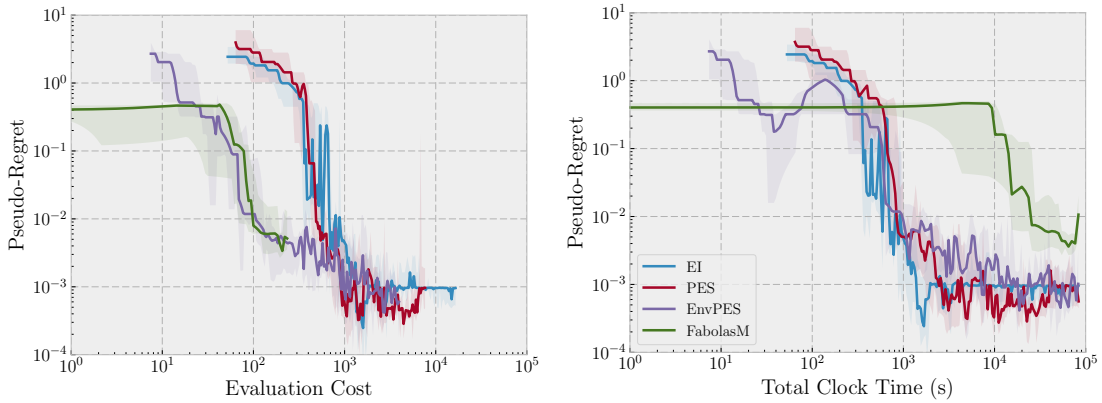

Figure 4.4: Performance of EnvPES, FABOLAS, PES and EI minimizing the negative hyperparameter log-likelihood for the combined cycle power plant dataset with a single lengthscale. The median and interquartile range (shaded) of ten runs are shown. Note that the difference between the maximum evaluation cost and maximum total cost: the majority of time is spent on optimizing the acquisition function. With such a comparatively cheap objective the improvements made in terms of just evaluation cost have been negated by the greater number of steps taken when full cost is considered for both variable fidelity methods. The much earlier start of the plot for FABOLAS is due to the choice of much cheaper initialization evaluations.

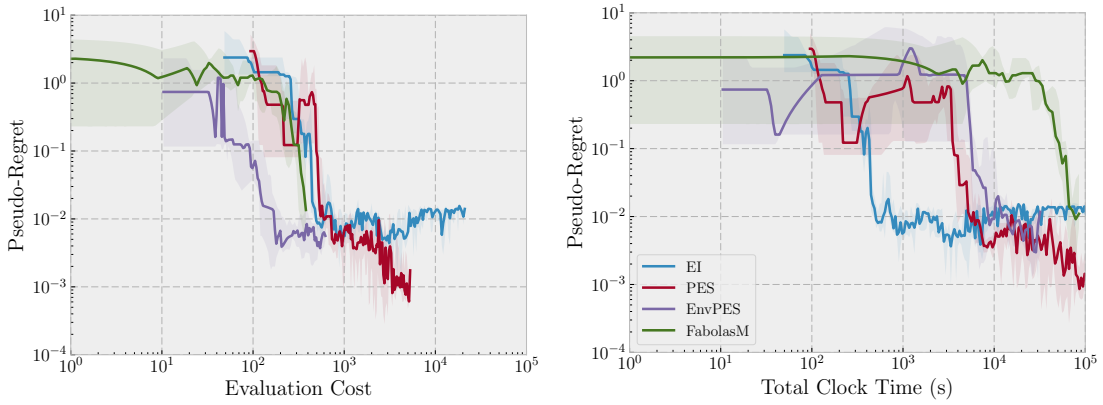

Figure 4.5: Performance of EnvPES, FABOLAS, PES and EI minimizing the negative hyperparameter log-likelihood for the combined cycle power plant dataset with a multiple lengthscales. The median and interquartile range (shaded) of ten runs are shown. Note that the difference between the maximum evaluation cost and maximum total cost: the majority of time is spent on optimizing the acquisition function. With such a comparatively cheap objective the improvements made in terms of just evaluation cost made by EnvPES have been negated by the greater number of steps taken when full cost is considered.

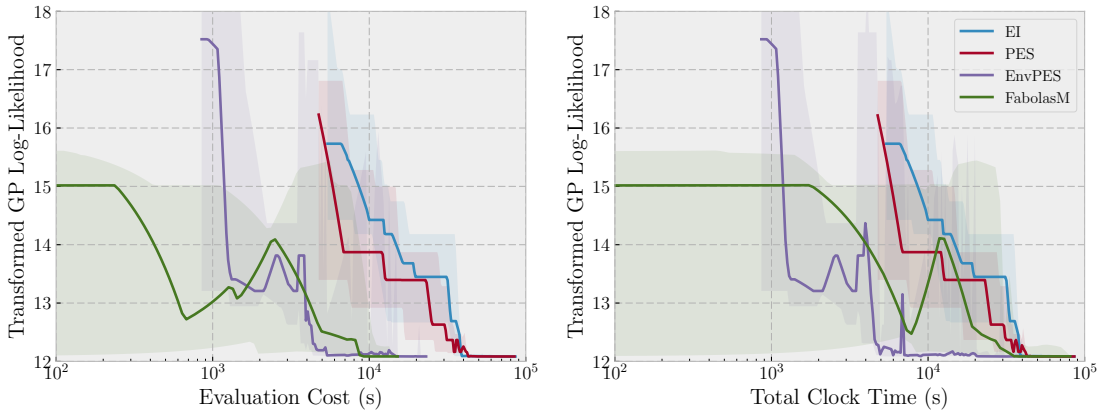

Figure 4.6: Performance of EnvPES, FABOLAS, PES and EI minimizing the negative hyperparameter log-likelihood for the physiochemical properties of protein tertiary structure dataset. The median and interquartile range (shaded) of ten runs are shown. The variable fidelity approach has a clear advantage, with close to optimal performance being obtained in barely more time than required by the first random evaluation at full cost. Both EnvPES and Fabolas show significant improvement in terms of evaluation cost alone. However, the higher overheads of FABOLAS prevent negate most of this improvement when included.

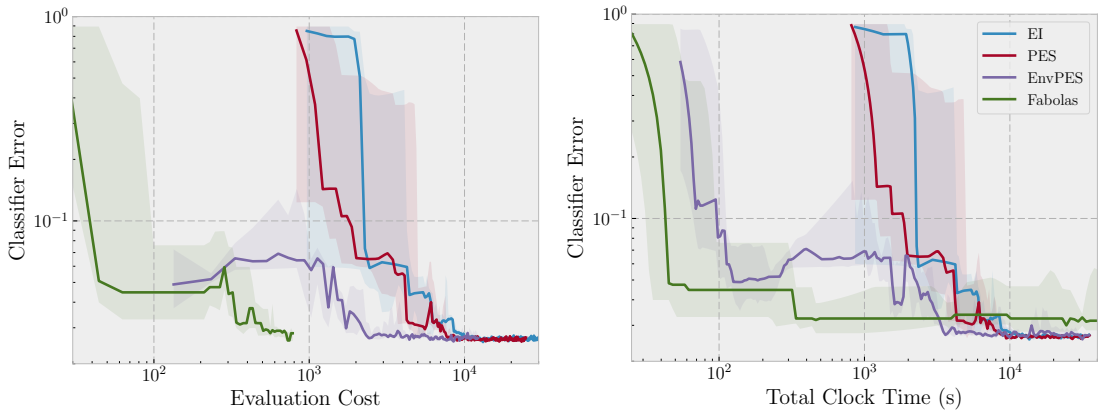

Figure 4.7: Performance of EnvPES, FABOLAS, PES, and EI finding the best hyperparameters for a support vector machine classifying the MNIST dataset. The median and interquartile range (shaded) of seven runs are shown. Here we have used the original form of FABOLAS, since classifier performance should always improve with more data available, and this has resulted in superior performance by FABOLAS in terms of evaluation cost alone. However, the high overhead costs of FABOLAS prevent convergence at the same rate as other methods when measured in terms of total computation time.

to achieve superior performance to the existing methods. Both our method and FABOLAS are able to achieve low values faster than methods not making use of the environmental variable. However, although it achieves the best performance of all methods when considering only evaluation costs, the particularly high overhead required by FABOLAS drastically reduces performance in real time.

Finally, we use support vector regression to perform inference on the 3D road network dataset. We have used half of the full dataset for our true objective, requiring approximately 100 minutes for training, and thus limiting the full-cost methods to only a few steps beyond random initialization within the 24 hour budget. By making intelligent use of reduced cost evaluations EnvPES is able to take many more steps which, as seen in Figure 4.8 results in a small, but still clear, performance improvement in real time.

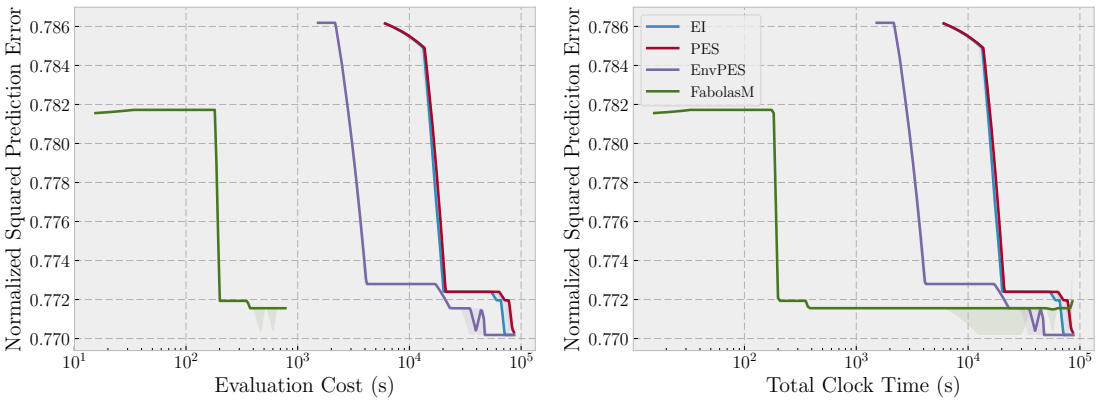

Figure 4.8: Performance of EnvPES, FABOLAS, PES and EI finding the best hyperparameters for support vector regression on the 3D road network dataset. The median and interquartile range (shaded) of ten runs are shown. FABOLAS has superior performance in terms of evaluation cost alone. However, the high overhead costs of FABOLAS prevent convergence at the same rate as other methods when measured in terms of total computation time.

5

Unbiased Variable Fidelity

Contents

In this Chapter we consider the special case of Bayesian optimization in an unbiased variable fidelity setting. We show theoretically for PES, and empirically for both PES and EI, that a practical method for setting the environmental variable does not exist unless both predicted performance and the expected growth of optimization overhead are considered. We then develop simple parametric models for these quantities, and show that using the learned models we can predict the best value for the environmental variable with reasonable accuracy.

5.1 Unbiased Variable Fidelity as a Special Case

In §4.2.1 we defined the variable fidelity problem as using Bayesian optimization to minimize a function $f(x)$ by taking evaluations of $f_v(x, s)$, such that $f_v(x, 0) = f(x)$ and subject to a cost function $c(x, s)$, which ensures that the cost of evaluations is monotonically decreasing with s for fixed x. This can be interpreted as finding the minimum in the $s = 0$ plane of the $D + 1$ dimensional function f_v. In the results presented in §4.4, we used an i.i.d. Gaussian noise model, based on the assumption that the change in the underlying true mean value of f_v is the dominant factor differentiating evaluations of differing cost which share the same x. However, this is not true of all problems. We now consider a special case, in which the environmental variable influences only the evaluation noise. This scenario may occur, for example, if evaluations are obtained by taking the mean of some number of cheap unbiased evaluations. An increased number of components will then provide decreasing variance at increasing costs. Furthermore, by the central limit theorem, the distribution of the mean will tend to a Gaussian distribution for a large number of samples. Our problem is then to minimize $f_v(x, s) = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma(s)^2)$, $\sigma(s)$ is some monotonically increasing function of s. We redefine our evaluation cost function as simply $c(\sigma^2)$, removing the dependence on x and defining the environmental variable as equal to the observation noise variance, $\sigma(s)^2 = s$.

We make the distinction at this point between white noise as additive noise on function evaluations and white noise as a model for a deterministic function. In the former case, repeated evaluation of the objective at the same location is a stochastic process and the covariance matrix of a GP model remains diagonal regardless of the locations of evaluations; in the latter case the function is evaluated using a purely deterministic process and our use of a white noise kernel is a model for some numerical error process. The GP covariance matrix in this case is not diagonal if locations are repeated, as the result obtained will be identical. It is the former case, in which evaluations at the same location are stochastic, and hence can be combined to obtain an equivalent single evaluation with lower variance, that we are concerned with in this chapter.

5.2 Theoretical Performance

To select the most efficient value for the environmental variable, we wish to express our acquisition function in terms of the environmental variable, and divided by the evaluation cost, then optimize over both the input space and the environmental variable, as in Section 4.2.1. For PES, this modified acquisition function is

$$\alpha = \frac{\mathbb{E}[\Delta H](x)}{c(\sigma^2)} = \frac{1}{c(\sigma^2)}\left(0.5\log(v(x)+\sigma^2) - \frac{1}{M}\sum_{i=1}^{M} 0.5\log(v_i(x \mid x_i^*)+\sigma^2)\right), \quad (5.1)$$

where $v(x)$ is the GP posterior variance at x marginalized over the GP hyperparameters and $v_i(x \mid x_i^*)$ is the new posterior variance with the i^{th} sampled global minimum, x_i^*, included in the training data and marginalized over the same hyperparameter samples. Consider the case in which the relation between evaluation cost and σ^2 takes the same form as that for the combination of N independent Normal observations, that is

$$\begin{aligned} \sigma^2 &= \frac{\sigma_0^2}{N} \\ c(\sigma^2) &= aN \end{aligned} \quad (5.2)$$

where a and σ_0^2 are some positive constants representing the cost and variance of a single cheapest possible evaluation. In this case, there is no finite value of the environmental variable which maximizes the acquisition function. Instead, the maximum is the limit of $\sigma^2 \to \infty$. We show that this result is true for a combination of independent, or positively correlated, observations in Appendix B. The greedy choice using this acquisition function and class of cost function will therefore be to always take the cheapest, and highest noise, evaluation available at each step.

Unfortunately, this is not a practical choice. Although very large numbers of cheap evaluations may provide superior performance with respect to only evaluation cost in abstract, we will be unable to exploit this performance due to the rising overhead costs of the acquisition function. Furthermore, we can expect some reduction in performance due to conditioning noise in our covariance matrix, and due to imperfect optimization of acquisition function which is highly mutimodal. While we have only shown that an impractical solution is guaranteed to occur

for PES when dividing the original acquisition function by evaluation cost we should still be concerned with efficiency for other acquisition functions, and other methods of accommodating evaluation costs (e.g. subtracting evaluation cost from the acquisition function value). In general, we can expect that the unmodified acquisition function will not have any significant increases with further reductions in observation noise once a noise level which is sufficiently small compared to the scale of the objective function has been reached. The cost-modified acquisition function must therefore decrease due to rising evaluation costs as noise decreases. We do not know for any scenario other than the PES acquisition function divided by evaluation cost if this trend will be continued for large observation noise levels, in which case optimal theoretical performance is in the limit $\sigma^2 \to \infty$, or if a maximum will be obtained. Even in the case that a maximum does exist, there is no guarantee that it occurs at a location which will lead to a reasonable number of evaluations being taken, as for this we must also consider the total budget available. As an illustration, consider three otherwise identical optimization tasks, in which the first is afforded some budget B, while the second has $100B$ and the third only $0.01B$. Maximizing the acquisition function for optimal performance with respect to evaluation cost will lead to the same noise level in each case. With appropriately chosen parameters in the first scenario these evaluations may cost $\frac{1}{100}^{\text{th}}$ of the available budget and lead to reasonable optimization performance. However, the second scenario takes evaluations costing only $\frac{1}{10000}^{\text{th}}$ of the total budget. In this case optimization overheads will quickly grow to dominate the computational budget, and it seems reasonable to believe that better performance might be obtained with fewer evaluation of higher cost. Finally, the third scenario will use the full budget in a single evaluation, clearly a sub-optimal choice. We therefore turn to empirical observations of performance to obtain a more useful result.

5.3 Empirical Performance

To illustrate the actual range of performance for Bayesian optimization in a variable fidelity context we show in Figure 5.1 the median regret and quartiles of a large

number of optimizations with different noise settings on objectives drawn from the 3D Matérn kernel with lengthscale 0.5. We have shown the same optimizations in terms of iterations, only evaluation cost, and total cost (evaluation and overheads), for both EI and PES. The cost of evaluation is determined as $c(\sigma^2) = \frac{3\times10^{-5}}{\sigma^2}$, which can be interpreted as the combination of independent observations each having variance $\sigma^2 = 1$ and requiring 3×10^{-5}s to evaluate.

Our prediction that using cheaper evaluations will provide superior performance for PES appears to be correct for this example, and EI exhibits the same behaviour. However, as seen on the top row of Figure 5.1, the performance for the same *number* of evaluations is much worse for high variance observations. We can therefore expect to require many more low cost evaluations to achieve the same regret as a smaller number of high cost evaluations (this does not preclude the low cost option being more efficient with respect to cost). At some point the cubic overhead cost at each step must limit our ability to undertake progressively greater numbers of cheaper evaluations. Indeed, when the same results are plotted in terms of total cost we can see that no single choice of evaluation cost yields the best performance. Instead, each curve has a particular region of total cost in which it is best performing. Low evaluation costs are preferable for low total costs, since they make more efficient use of evaluations, while higher evaluation costs are preferable for higher total costs, since the more efficient low evaluation cost choices are handicapped by rising overheads.

We therefore conclude that there is no single optimal fixed value of the environmental variable for Bayesian optimization in this setting. We must instead make a more complex decision, taking into account:

1. the total computation budget available;

2. the predicted overhead costs of our acquisition function;

3. the expected performance of our acquisition function, as a function of observation noise variance, and conditioned on our prior belief for the objective function.

To make this decision we require models for both the rising overhead cost and the expected performance.

5.4 An Algorithm for Unbiased Variable Fidelity

5.4.1 Modelling Overhead Cost

In order to predict the number of iterations that will be taken for a given computation budget and evaluation cost, and hence overall performance, we first require a model for the optimization overhead.

A selection of measured optimization overheads using EI for a three dimensional objective are shown in Figure 5.2. Since the overhead is the sum of times for linear algebra operations with well known scaling characteristics, the choice of a cubic model is simple. However, in addition to per-step noise, there is clearly considerable variation between different optimization runs. We therefore model each individual sequence of optimization overheads as a separate cubic with multiplicative unit mean Gaussian noise

$$p(c \mid n, \Theta) \sim \mathcal{N}(\theta_3 n^3 + \theta_2 n^2 + \theta_1 n + \theta_0, \theta_4)\,, \tag{5.3}$$

where the θ_i are the five components of Θ, which we in turn model as being drawn from a multivariate normal prior, $\Theta \sim \mathcal{N}(\mu, \Sigma)$ with hyperparameters μ and Σ. These hyperparameters will be unique for each dimensionality and acquisition function. To avoid conditioning problems in implementation, we parametrize the cubic mean function in terms of the value and gradient at $n = 0$ and $n = 200$ rather than using the monomial coefficients. This is a simple linear transform of Θ so has no effect on the model.

From a dataset of N optimizations we can infer a set of N maximum likelihood parameters Θ_i using local optimization of the log-likelihood and hence the maximum likelihood values of μ and Σ. We show in Figure 5.2 the learned model predictions, and hence the predictions over the cumulative overhead, using the integral of the cubic model, for an example set of optimization data.

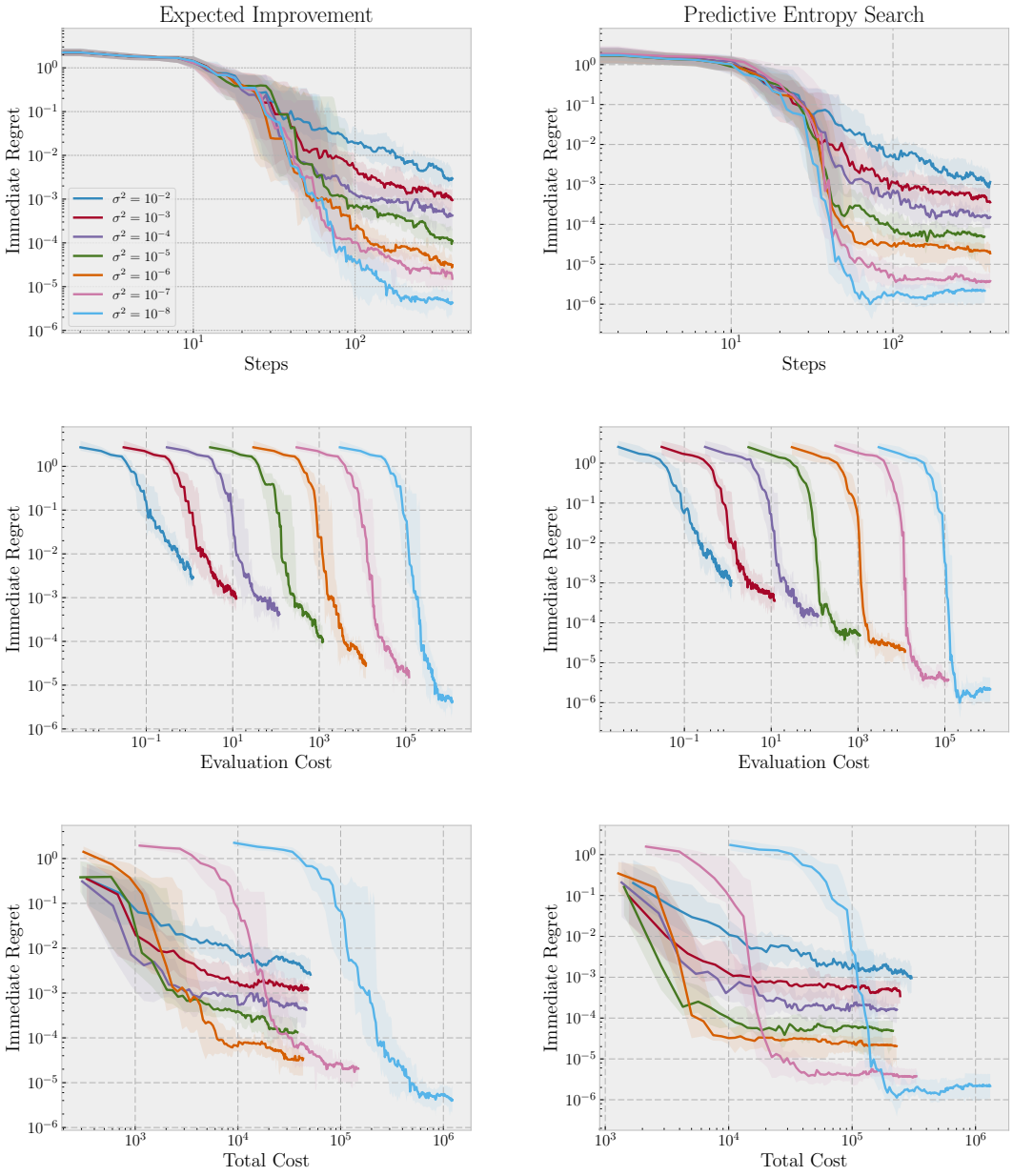

Figure 5.1: The performance of EI and PES on objectives drawn from the Matérn 5/2 kernel with lengthscale 0.5 for a a range of values of observation noise. Noise variance is linked to evaluation cost by $c(\sigma^2) = \frac{3\times10^{-5}}{\sigma^2}$ (an evaluation with variance $\sigma^2 = 10^{-6}$ requires 30s to complete). Results are shown in terms of iterations, evaluation cost and evaluation plus overhead costs. For both acquisition functions high noise evaluations provide better performance in terms of evaluation cost only, but are no longer competitive when the high overhead costs are included.

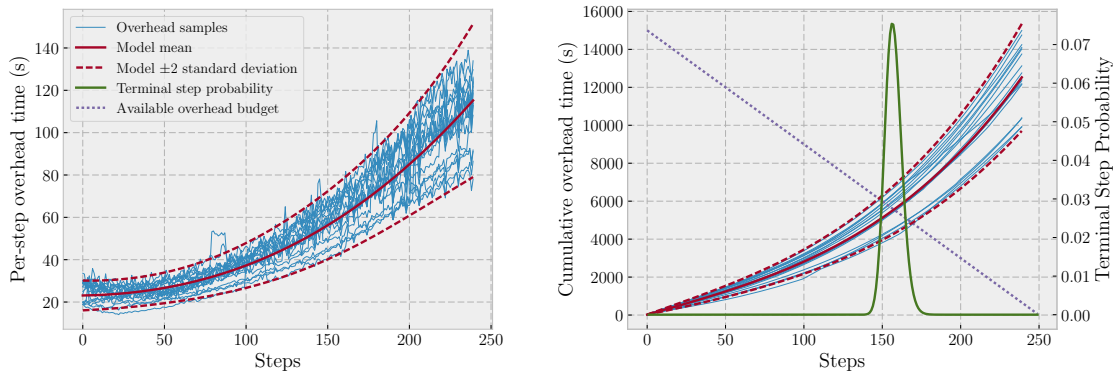

Figure 5.2: Per-step and cumulative overhead costs for EI in three dimensions. The overhead is modelled as a cubic with multiplicative normal noise and parameters drawn from a multivariate normal distribution. The mean and ±2 standard deviation of predictions are shown. With a total computation budget sufficient for 250 evaluations requiring 60 seconds each the number of iterations we are able to be able to take when including the cost of overheads takes a unimodal distribution centred at roughly 160.

5.4.2 Predicting the Number of Iterations

We denote the cumulative overhead cost up to iteration N as

$$C(n) = \sum_{i=1}^{n} c(n) \sim \mathcal{N}(\mu_{\text{cumulative}}(n), \sigma^2_{\text{cumulative}}(n)) \,, \tag{5.4}$$

which has a normal distribution since each individual $c(i)$ is also normally distributed (even if we had chosen some other distribution for $c(i)$ the central limit theorem ensures that $C(n)$ approaches the normal distribution for large n). With a total computation budget B and per-step evaluation cost s the time available for overhead computation in an optimization taking n steps is $B - ns$. Therefore, to take at least n iterations we require $C(i) < B - is$ for all $i \leq n$. However, since $C(i)$ is strictly increasing while $B - is$ is strictly decreasing, this is equivalent to $C(n) < B - ns$. The probability that we will be able to take at least m iterations is therefore:

$$\begin{aligned} p(n \geq m \mid B, s) &= p(C(m) < B - ms) \\ &= \frac{1}{\sigma_{\text{cumulative}}(m)} \Phi\left(\frac{B - ms - \mu_{\text{cumulative}}(m)}{\sigma_{\text{cumulative}}(m)}\right) . \end{aligned} \tag{5.5}$$

Since the number of iterations must be an integer, the probability of taking exactly m iterations is simply:

$$p(n = m \mid B, s) = p(n \geq m \mid B, s) - p(n \geq m + 1 \mid B, s) \,. \tag{5.6}$$

We show this resulting distribution, which we will then use in combination with a model of optimization performance, for an example budget and step cost in Figure 5.2.

5.4.3 Modelling Performance

To predict optimization performance we again select a relatively simple parametric model. The optimization plots shown in Figure 5.1 exhibit a distinct bi-linear trend when shown on a log scale (excluding the first ten points which are given determined by random initialization). We therefore adopt a model using two terms of the form $\frac{a}{x^b}$ (this takes a piecewise linear shape viewed on a log-log scale) with log-normal multiplicative noise. The coefficient of one decay term is specified to be a linear function of the observation noise variance, while the other coefficient and both powers are constant:

$$R = (\phi_0 n^{-\phi_1} + (\phi_2 + \sigma^2 \phi_3) n^{-\phi_4}) \exp(\kappa), \qquad \kappa \sim \mathcal{N}(0, \phi_5)\,. \tag{5.7}$$

These parameters ϕ are then expressed as a linear function of the log lengthscale of the objective function

$$\phi_i = \psi_i + \psi_{6+i} \log(L) \tag{5.8}$$

to give a twelve parameter model for $p(R \mid n, L, \Psi, \sigma^2)$ for optimization regret as a function of step number and objective function lengthscale.

Given a dataset of optimizations with known objective function lengthscale and observation noise we place broad independent normal priors over the components of Ψ and use a local search to obtain the maximum a-priori parameters, Ψ_{MAP}. As shown in Figure 5.3 the model obtained provides reasonable predictive performance, using a Matérn kernel in three dimensions, although it is expected that considerable improvement could be made with more advanced methods. However, this simple model provides sufficiently good performance for our objective here.

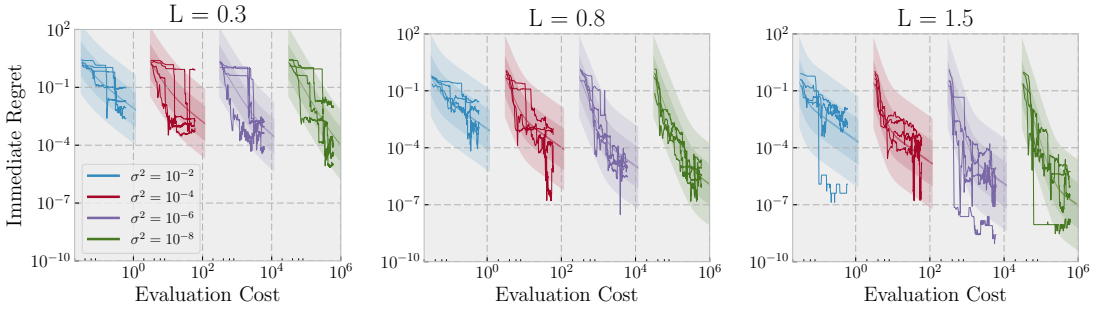

Figure 5.3: Predicted optimization performance and actual observed performance using our log-bi-linear model at a selection of lengthscales and observation noise variances. The model was trained using a dataset of 1848 optimizations on sample objectives with known lengthscale drawn from the Matérn 5/2 kernel.

5.4.4 Regret Prediction

Combining our models for optimization performance and overhead costs we now wish to obtain a prediction for the performance of optimization on a given budget and per-step cost. To achieve this we must marginalize over both our belief over the objective function lengthscale, $p(L)$, and the number of steps that will be taken given the available budget and per-step cost, $p(n \mid B, s)$.

$$\begin{aligned} p(R \mid \Psi_{\text{MAP}}, \sigma^2, B, s)) &= \int_0^\infty \left(\sum_{n=0}^{\frac{B}{s}} p(R \mid n, L, \Psi_{\text{MAP}}, \sigma^2) p(n \mid B, s) \right) p(L) \mathrm{d}L \\ &\approx \sum_{j=0}^{J} \left(\sum_{n=0}^{\frac{B}{s}} p(R \mid n, L_j, \Psi_{\text{MAP}}, \sigma^2) p(n \mid B, s) \right) p(L_j) \\ &= \sum_{j=0}^{J} \sum_{n=0}^{\frac{B}{s}} w_{jn} p(R \mid n, L_j, \Psi_{\text{MAP}}, \sigma^2) \,, \end{aligned} \tag{5.9}$$

where $w_{jn} = p(n \mid B, s)p(L_j)$, and the integral over the lengthscale parameter has been approximated as a summation. Unfortunately as this is a weighted sum of log-normal distributions there is no closed form for the resulting distribution. However, we only require the expected value. Letting the log-mean and log-variance parameters of each $p(R \mid n, L_j, \Psi_{\text{MAP}}, \sigma^2)$ be μ_{jn} σ^2_{jn} this is given by

$$\mathbb{E}\left[R \mid \Psi_{\text{MAP}}, \sigma^2, B, s\right] \approx \frac{\sum_{j=0}^{J} \sum_{n=0}^{\frac{B}{s}} w_{jn} \exp(\mu_{jn} + \frac{\sigma^2_{jn}}{2})}{\sum_{j=0}^{J} \sum_{n=0}^{\frac{B}{s}} w_{jn}} \,. \tag{5.10}$$

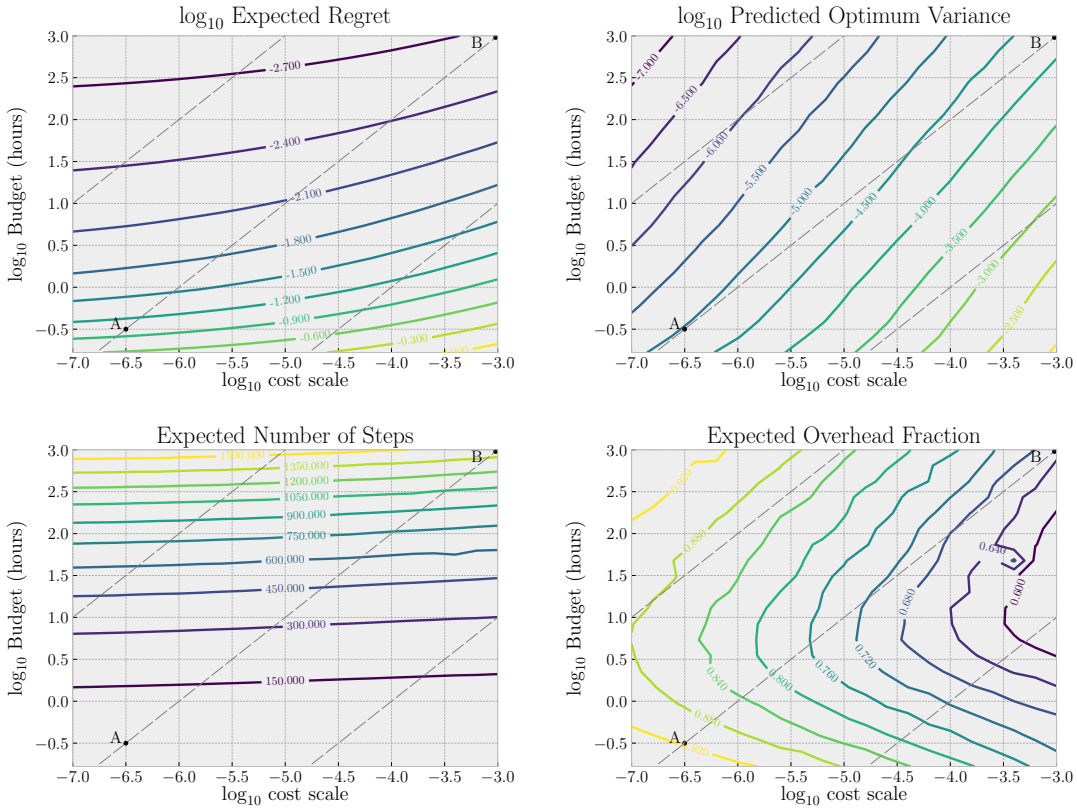

Figure 5.4: Predicted optimum observation noise variance, and conditioned on this choice the associated expected regret and number of steps over a range of total budget and cost function values. Diagonals with unit gradient represent problems which are equivalent in terms of budget and cost function, differences along these paths are the result of overhead costs reducing the budget available to objective evaluations. The cost function coefficient is shown plotted as the $\log_{10}$ variance that would be obtained by an evaluation taking one minute, while budget is plotted as $\log_{10}$ hours.

We can then minimize this expected regret to find the predicted best performing observation variance for a given budget, lengthscale prior distribution and cost-variance relationship

$$\sigma^2_{opt}(\Psi_{\text{MAP}}, B, c(s), p(L)) = \arg\min_{\sigma^2} \mathbb{E}\left[R \mid \Psi_{\text{MAP}}, \sigma^2, B, s\right] . \tag{5.11}$$

5.5 Results

We trained our performance and overhead models using a dataset of 1848 optimizations on $[-1, 1]^3$ up to 200 iterations, with objective lengthscales between

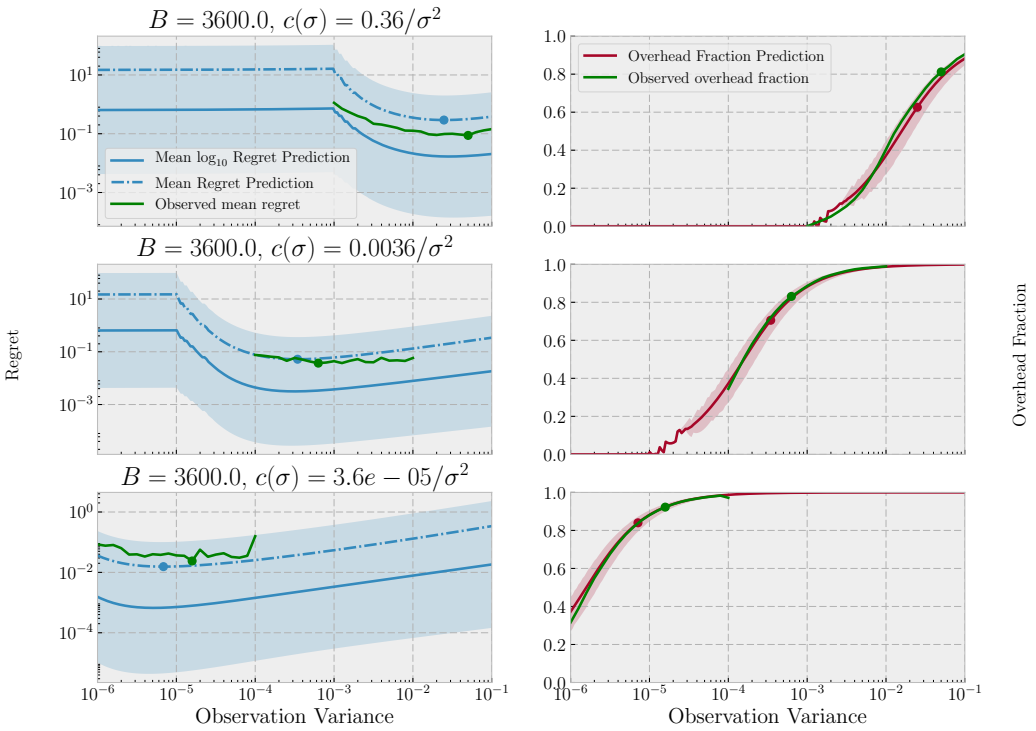

Figure 5.5: Predicted, and observed, regret and overhead cost fraction for three objective cost functions. All three cases have a total budget of one hour, while observations taking one minute to evaluate are subject to noise with variance 10^{-2}, 10^{-4} and 10^{-6} from top to bottom respectively. For regret we show the predicted mean and its minimum, as well as the mean and two standard deviations of the log-distribution. The observed mean regret over 8 optimizations, marginalized over the objective function lengthscale prior, follows a similar shape to the predicted mean. Our prediction of the fraction of the total budget dedicated to overhead costs has a close fit to the observed value, with the optimum performance having over two thirds of the total cost coming from acquisition function overhead.

0.05 and 1.5, and observation noise with variance between 10^{-2} and 10^{-8}. We use a Gamma prior with shape parameter 4 and scale 0.2 for a single lengthscale hyperparameter to make predictions. We have shown in Figure 5.5 predictions of performance and overhead cost in comparison to that of actual optimizations for three pairings of budget and cost function. While the model has not identified the exact optimum observation variance, the overall shape of the performance curve is accurate, and our predicted minimum is reasonably close to the observed location. Our predictions of the fraction of the total computation budget used by overheads

are more accurate. Notably, at the observed overhead fractions yielding optimum performance, the overhead costs account for over two thirds of the total cost, more than our already high predicted optimum values.

The predicted optimum noise variance, and conditioned on this choice the associated expected regret, the number of steps, and the fraction of the total budget given to overhead computation are shown as contour plots covering a range of available budget and evaluation costs in Figure 5.4. Unsurprisingly, increases in budget, or decreases in the cost of observations, lead to selecting more accurate evaluations and improved predicted regret. However, the inclusion of overhead costs in our model means that locations with equivalent budget and cost functions do not have the same predicted performance, with higher budgets leading to better predicted performance using cheaper evaluations, as the increased budget allows for greater overhead costs. Consider the central diagonal in each plot in Figure 5.4. All points on this line represent problems for which 60 objective function evaluations with observation noise variance $\sigma^2 = 10^{-6}$ are available. However, in the bottom left corner (A) this corresponds to evaluations taking $\sim$ 20s out of a 20 minute budget, while at the top right (B) the same evaluations require 16.7 hours from a 41.7 day budget. Under the usual assumption in Bayesian optimization that overheads are negligible we would expect the same performance in both cases, while in fact scenario A has an expected final regret of $\sim 10^{-1}$ while scenario B is predicted a substantially better value of $\sim 10^{-2.6}$. The principle reason for this difference is that the reduction in available budget for evaluations is much more significant for the scenario with a smaller total budget. For scenario A the overhead of less than 150 iterations consumes over 90% of the total budget while for scenario B the overhead of over 1350 iterations consume less than 68%. The combination of a greater number of objective evaluations, and the noise of those evaluations being almost an order of magnitude lower naturally lead to much greater performance in scenario B, a surprising result for two problems which are equivalent up to a constant scaling of cost and budget.

The fraction of the total budget that has been assigned to overhead cost is perhaps surprisingly high: over half for the range of scenarios we have shown. This leads to many more steps being taken than are usually shown in Bayesian optimization literature. In scenario A, where overheads clearly dominate the total computational cost, it is reasonable to question if Bayesian optimization is the best strategy, and we believe investigating quantitative ways of determining the boundary between optimality of Bayesan optimization and other optimization methods would be a valuable avenue for future work. However, the least fraction of overhead cost predicted over a wide range of settings in Figure 5.4 is 60%. Any reduction in overhead costs which does not otherwise change the optimization performance would be equaivalent to an equal scaling up of budget and evaluation cost, and will therefore lead to improved performance. Our result in this chapter, in addition to demonstrating a method for obtaining optimum performance in an unbiased variable fidelity setting, provides a strong motivation to seek reduction in overhead costs.

6 Implementation

In this Chapter we propose two methods for reducing the overhead costs of Bayesian optimization using Gaussian processes without altering the acquisition function in use. We first consider the process of marginalizing the GP hyperparameters, and propose replacing the usual method of slice sampling with adaptive quadrature. We show that the number of expensive $\mathcal{O}(n^3)$ operations then decays during optimization, a substantial cost saving. Finally, we note the potential for cost saving by using batch evaluations of the acquisition function in the inner optimization routine.

Contents

6.1 Problem Motivation

In Bayesian optimization we usually assume that optimization of the acquisition function can be considered to have negligible cost in comparison to the objective. However, the overhead is not truly negligible, taking on the order of seconds

to maximize the EI acquisition function in our implementations (running single threaded on an Intel® Xeon™ E5 processor) even with a small number of observations. This rises to minutes with $\mathcal{O}(100)$ observations due to the cubic scaling of the GP. Bayesian optimization is therefore limited in application to objectives sufficiently expensive to outweigh this overhead. Any substantial reduction in overhead costs, with or without changing the ultimate cubic scaling, is therefore desirable as it allows a greater number of iterations to be taken before the overheads become comparable to the objective. This widens the practical applicability of Bayesian optimization to more complex or cheaper objectives. In this chapter we highlight two opportunities for improved performance when using Gaussian processes independent of the details of the particular Bayesian optimization algorithm in use.

6.2 Hyperparameter Marginalization

6.2.1 Slice Sampling

The use of slice-sampling to generate hyperparameter samples for approximate marginalization is demonstrated by Murray and Adams [2010] and subsequently used by many Bayesian optimization works including Snoek et al. [2012]; Swersky et al. [2013, 2014]; Hernández-Lobato et al. [2014]. This marginalization provides a performance improvement over using only a single maximum likelihood estimate of hyperparameters (Snoek et al. [2012] provide direct comparison for expected improvement on several objectives). A linear combination of the predictions under each hyperparameter sample is then used for inference.

The use of this method for marginalizing over the hyperparameters is attractive, since marginalization is a quadrature problem, and performing quadrature using Monte-Carlo method with m samples provides $\mathcal{O}(m^{-\frac{1}{2}})$ convergence, independent of dimensionality. In a typical Bayesian optimization application the kernel has $D+1$ hyperparameters to marginalize. However, we must draw our m hyperparameter samples from the GP posterior, which we are not able to directly generate samples from since the hyperparameter prior is transformed by the highly non-linear GP log-likelihood (Equation 2.21). Instead we perform slice-sampling using the GP

log-likelihood. This incurs an $\mathcal{O}(n^3)$ cost for each evaluation, and performs a line search *along each axis* between every sample. Generation of the m samples thus incurs an $\mathcal{O}(hmn^3)$ cost, where h is the number of hyperparameters. The multiple evaluations required for each linesearch, and the common practice when using MCMC of discarding an initial burn-in period and subsampling due to correlation between consecutive values, make the constant multiplier of this cost non-trivial.

In Osborne et al. [2009] this high cost at each iteration is avoided by using Bayesian Monte-Carlo [Rasmussen and Ghahramani, 2003] with a fixed hyperparameter grid of samples to perform hyperparameter marginalization (full detail is given in Osborne et al. [2008]). The posterior distribution is then a linear combination of Normal distributions with unequal weightings. Since the same hyperparameter values are reused at each step, the GP log-likelihood can be obtained using only an $\mathcal{O}(n^2)$ update for each sample. However, the sampling pattern is fixed from the first step, at which point it must provide support over a wide prior. By contrast the hyperparameter posterior contracts to a narrow peak as additional information is gained at each step of optimization. This will quickly lead to only a few elements of the sampling pattern having non-trivial quadrature weights.

6.2.2 An Alternative Approach: Adaptive Quadrature

We propose to the use of adaptive quadrature to achieve a balance between the two extremes of cost and effectiveness. The basic procedure for adaptive quadrature in multiple dimensions, as described in Van Dooren and de Ridder [1976], is to divide the space to be integrated into a set of subregions. A two-level quadrature rule is used to estimate the integral, and the error on that estimate, in each region. We then select the region with the greatest error estimate and subdivide. The quadrature rule is then applied to both new regions, hopefully resulting in errors which sum to less than the previous value. This process is repeated until a stopping condition, usually either the total error or the number of iterations, has been satisfied. The effect of this process is to concentrate evaluations in locations where

the function takes on high or rapidly changing values. Adaptive quadrature, while not independent of dimension, is therefore greatly preferable to a fixed scheme.

Between each iteration of Bayesian optimization only a single new point is added to the dataset. The shape of the posterior over hyperparameters should therefore remain similar after each update, with the greatest changes occurring early in the optimization. We therefore expect that from step to step the choice of regions to subdivide in adaptive quadrature will remain mostly the same. Since the locations evaluated within each region follow a fixed pattern determined by our choice of quadrature rule individual hyperparameter values will be reused many times during optimization. If we store the Cholesky decomposition each time we make an evaluation of the GP log-likelihood, we can then use this to make only $\mathcal{O}(n^2)$ updates when we return to that location in future iterations. The full cost, $\mathcal{O}(n^3)$, operation is then only required in subdivisions of hyperparameter space not previously used. We hope that as optimization progresses and our dataset grows, each successive new observation will have a diminishing impact on the shape of the hyperparameter posterior and the number of new evaluations required will approach zero.

As our quadrature scheme we choose to use the Trapezoid rule of second and third order. This places evaluations at the corners of each rectangle for the second order rule, and additionally at the midpoints for the third order rule. Since all second order evaluations are reused by the third order rule, and the midpoint evaluations of parent hyperrectangles become the corners of the child hyperrectangles on division, this choice means the hyperparameters of all log-likelihood evaluations we take will be are used for marginalization. All three of the schemes under comparison are illustrated in Figure 6.1.

6.2.3 Results

We use a fixed sequence of objective evaluations from Bayesian optimization using PES as our dataset. For each iteration of the optimization, n, we then compare the three methods of hyperparameter marginalization considered to approximate the integral in Equation 2.28. We have used the same code (a simple python

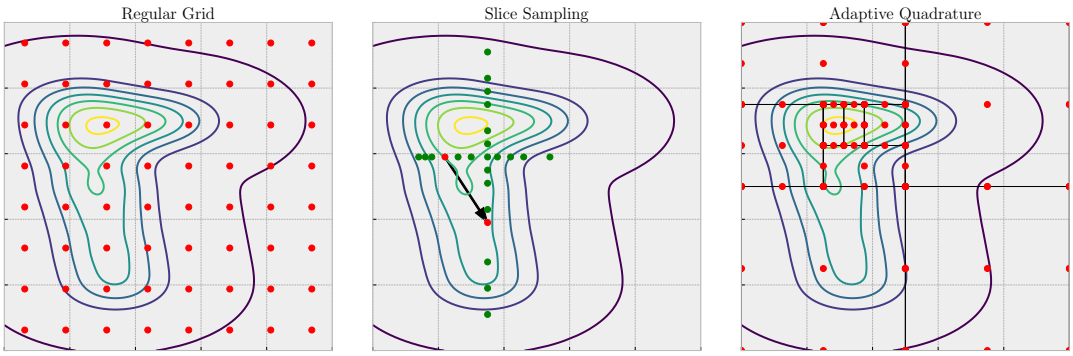

Figure 6.1: Illustration of the quadrature methods under comparison over an example likelihood function. On a regular grid almost all points have very low likelihood so contribute little to marginalization. Slice sampling generates equally weighted samples from the likelihood being integrated over, but a large number of likelihood evaluations are used between each actual sample. Adaptive quadrature divides the space in a deterministic manner to cluster points around the peak likelihood.

implementation) to perform the Cholesky decomposition and updates in each case. A fixed linear spacing of 14 points on each axis is used for the fixed grid, while adaptive quadrature undertakes two hundred divisions. This gives roughly 2500 hyperparameters values in each case, the number for adaptive quadrature is not fixed, as divisions may be able to reuse values already calculated for adjacent hyperrectangles. In our slice sampling test, we only generate a set of only 200 samples, do not subsample, and use a burn-in period of only 20 samples.

The number of log-likelihood evaluations at each iteration taken by each method is shown in Figure 6.2, separated into $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ operations. The grid method obviously performs best under this measure, since the additional memory used to store the Cholesky decompositions across iterations allows every operation to be a fast update. While adaptive quadrature does require some cubic evaluations initially, this number rapidly reduces so that the overwhelming majority of operations are also fast updates. Slice sampling, despite only producing a smaller dataset, requires a large number of log-likelihood evaluations for each point produced, so tests many more hyperparameter values than the other methods, all requiring an expensive cubic calculation. In this measure of performance slice sampling is therefore by far the worst.

We show in Figure 6.3 the average number of $\mathcal{O}(n^2)$ update steps our adaptive method requires to perform log-likelihood evaluations. The plot always takes small values rather than rising linearly, indicating that as expected sequential adaptive quadrature operations take very similar set of subdivisions.

As a measure of the usefulness of the quality of the support points produced, we show in Figure 6.4 the fraction of hyperparameter values with likelihood values greater than 0.01 of the maximum likelihood observed in that iteration. The points produced from slice sampling have been drawn from $p(\theta \mid X, Y)$ so are combined with equal weighting, and almost all samples do fulfil this criterion anyway. However, the values obtained for use in quadrature are weighted according to the $p(\theta \mid X, Y)$. Any points where this value is negligible are assigned similarly negligible weightings, so do not provide a useful contribution to our model. As expected the fraction of points that make non-trivial contribution becomes very low for the fixed grid method since the hyperparameter posterior has contracted due to the observed data. Very few points on the grid are close to the peak hyperparameter likelihood. Use of adaptive quadrature has ensured that about a third of our hyperparameter samples are assigned non-trivial weighting as the integrand contracts.

The execution time for the three methods are shown in Figure 6.5. As expected the fixed grid method has the lowest execution time. Slice sampling is over an order of magnitude slower despite having produced far less samples, due to the large number of evaluations that do not contribute the the marginalization and the cubic cost of each. Our adaptive quadrature method achieves execution time similar to the fixed grid method, while performing a much more effective marginalization. We therefore strongly recommend use of such adaptive schemes for hyperparameter marginalization in any Bayesian optimization implementation requiring low overheads.

6.3 Batched Acquisition Search

Little attention is usually given to the details of optimization of the acquisition function. DIRECT is a particularly common choice (examples include Osborne

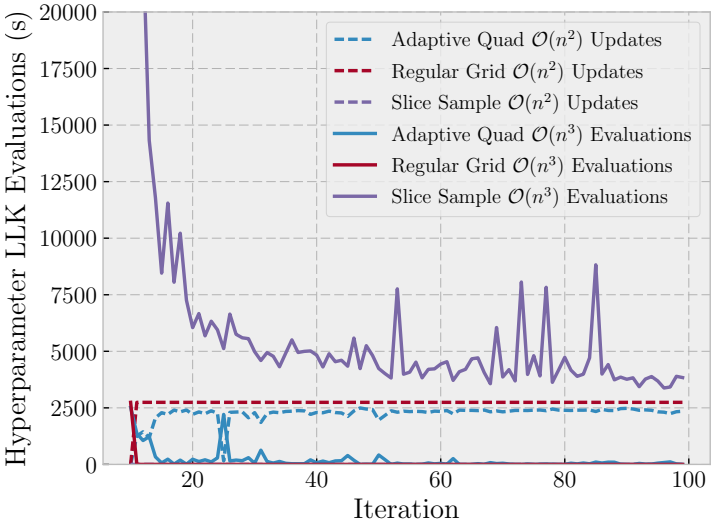

Figure 6.2: Number of full $\mathcal{O}(n^3)$ decompositions and cheap $\mathcal{O}(n^2)$ updates taken by each method over sequential marginalizations. The number of full evaluations required by the adaptive quadrature method quickly falls to a small fraction of the total. Slice sampling requires many more evaluations than other methods and all are cubic.

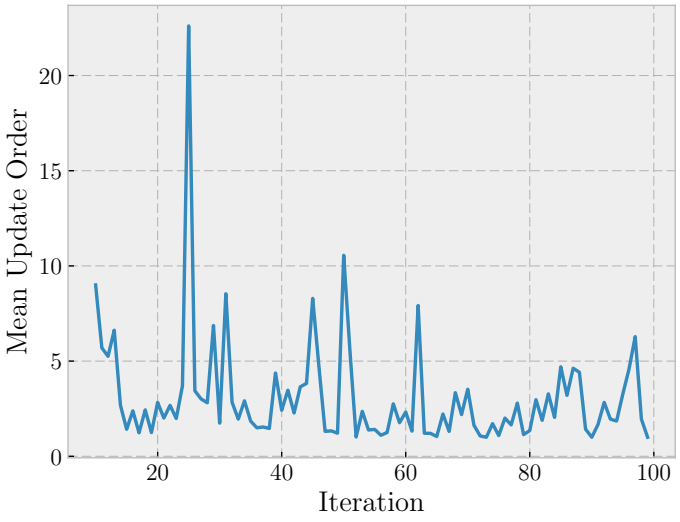

Figure 6.3: Mean number of Cholesky update steps required in evaluations of the log-likelihood by the adaptive quadrature method. For a fixed grid only one update is ever required since all locations are always repeated. While the number of updates required is greater than one on most iterations it is much lower than that of slice sampling, which effectively requires n updates at iteration n since all evaluations values used are unique.

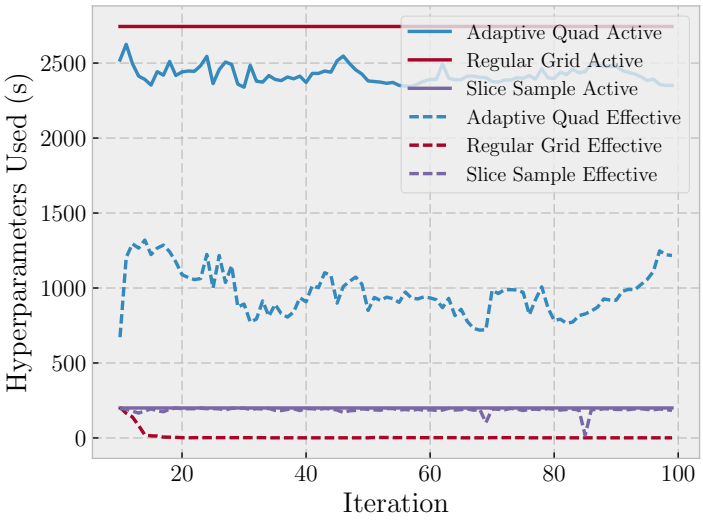

Figure 6.4: Total number, and non-trivial number, of quadrature points obtained by each method. We consider a hyperparameter value to be significant if its likelihood takes a value greater than one percent of the maximum observed. Almost all slice sampling points pass this test while almost all grid points do not. Roughly one third of points used by adaptive quadrature are will be assigned non-trivial weights.

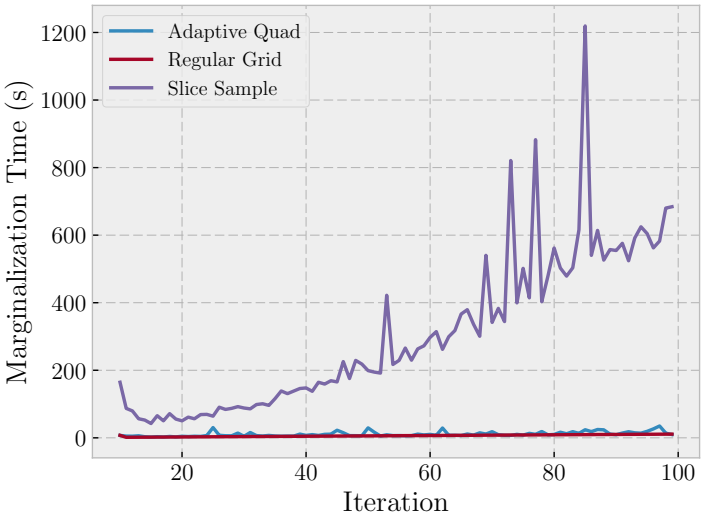

Figure 6.5: CPU time used by each marginalization method. Adaptive Quadrature, while incurring a slightly greater cost than the grid method does appear to have $\mathcal{O}(n^2)$ growth. Slice sampling has much greater cost with faster growth despite producing far less hyperparameter samples. Times are measured single threaded using an Intel® Xeon™ E5 processor.

et al. [2009]; González et al. [2016b]), while Calandra et al. [2016] further refines the solution provided by DIRECT using a local search. The provided implementation of PES [Hernández-Lobato et al., 2014] performs a local search starting from the best location on a regular grid of one thousand points, while CMA-ES is used by Gunter et al. [2014]. Following these examples we have also used DIRECT followed by a local refinement in our implementations. The common feature between these searches is their use of sequential evaluations of the acquisition function in implementation.

The EI, LCB, and PI acquisition functions are all simple functions of the GP posterior mean and variance at a point, while PES requires the mean and covariance matrix between two points (the query point and the sampled minimum), both unaltered and conditioned on global minimum observations. This inference incurs $\mathcal{O}(n^2)$ cost, and naturally inference at m separate locations incurs $\mathcal{O}(mn^2)$ cost. However, these are only orders of scaling for large values of n, m. The timing behaviour for small values is determined not by the mathematical operations to be performed, but by the architecture and operating system of the device performing computation. An introduction to operating systems is provided by Doeppner [2011], however two effects are of particular concern.

Cache hits In order for a CPU to perform operations the data must first be read from memory into a CPU register, usually via a hierarchy of intermediate caches of increasing speed but decreasing size. If a program is designed such that sequential operations are made on data that is stored in adjacent locations then the data required for each operation is far more likely to already be located in the fastest cache when required (a cache hit). This avoids considerable delays in execution that are incurred when the CPU must wait for required data to be fetched from a slower storage location (a cache miss).

Function call overheads Whenever a program makes a function call considerable overhead is incurred in order to save the current state of execution, load the instructions for the new function, and then restore the previous state when

that function returns. In an implementation of Bayesian optimization each function call to evaluate the acquisition function might in turn call a function to infer the posterior mean and variance, which in turn calls a linear algebra subroutine to operate on the raw data. If a program is designed such that functions operate on data in batches whenever possible then much of the overhead that would be incurred by individual function calls can be avoided.

The combination of these effects lead us to expect that inference at m locations in sequence will considerably more expensive than a single operation. Therefore, if the acquisition function is evaluated in batches of locations simultaneously we can expect a substantial decrease in the overhead costs of optimizing the acquisition function.

We show in Figure 6.6 a comparison of the cost of evaluation of the Expected Improvement acquisition function in sequential and batched form. As expected the cost of sequential evaluations is much greater. If we can perform optimization of the acquisition function using an algorithm which performs otherwise identically, but evaluates in batches of $\mathcal{O}(10)$, we can expect nearly an order of magnitude reduction in the overhead cost of this process.

Therefore, we wish to modify our inner search routine to allow batch evaluations. Although we not aware of any existing implementations that take sequential evaluations, the DIRECT and CMA-ES algorithms already have logical divisions into batches of evaluations. In DIRECT, each iteration selects a set of rectangles to divide then splits each along multiple axes, taking two evaluations for each split. CMA-ES evaluates a fixed size population sampled from a multivariate normal distribution at each step. There is also some opportunity for batch evaluations in second-order local optimization algorithms, which commonly alternate between line searches and gradient approximation at a single point. Gradient estimation using function values evaluates multiple locations in a fixed design, and therefore could be performed as a single batch evaluation. Alternatively, if a multi-start local optimization algorithm is being used, then each incidence could be run simultaneously, and the evaluations requested by each collated into a batch at each iteration. Random and grid search implementations can of course also be trivially evaluated as a batch.

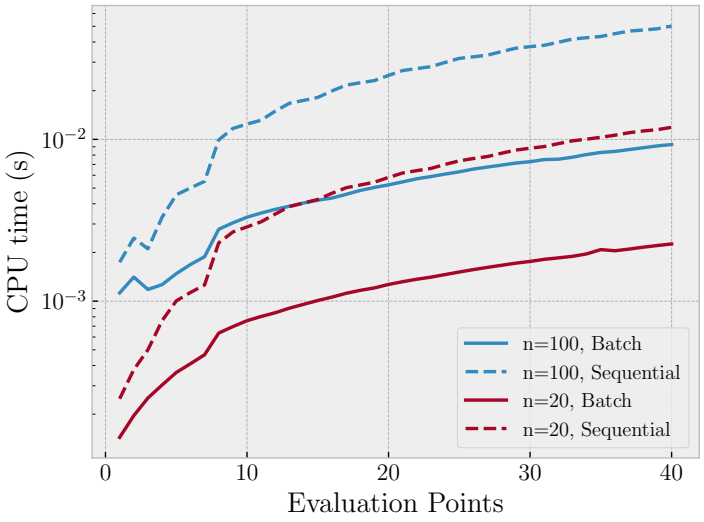

Figure 6.6: Evaluation times for the GP expected improvement averaged over 24 sampled hyperparameters using data sampled at random locations on the Hartmann 3D function. We show the median CPU time taken over 16 repetitions to evaluate EI either as a batch or each point sequentially for datasets of size 20 and 100. Both batch and sequential evaluations have clear linear trends as expected. However the gradient of the later is substantially greater (note the log scale). Times are measured single threaded using an Intel® Xeon™ E5 processor.

7 Conclusion

Contents

7.1 Overview of Contributions

The principal contributions in this thesis have been to the speed and practicality of Bayesian optimization.

The problem of terminating Bayesian optimization once acceptable results have been obtained is approached in Chapter 3. We develop BLOSSOM, a new Bayesian optimization algorithm, which makes an intelligent choice between multiple acquisition functions and pure local search at each step. This allows us to avoid the poor convergence properties of Gaussian process methods. Stopping is handled using an estimate of the expected global regret, a more intuitive quantity than local measures which have previously been proposed. The combination of fast convergence via local search and avoiding wasted effort once a sufficiently good result has been obtained allow us to achieve significantly better optimization performance than existing methods.

In Chapter 4 we first developed a pair of new methods for generating support points to be used in taking draws from the distribution of the minimum of a Gaussian process. In these methods the number of points generated is decoupled from the GP model, allowing large numbers of support points to be generated without significantly increasing costs. Furthermore, the support points produced were shown to have greater similarity to the true minimum distribution than existing methods. We then make use of these methods in adapting the PES acquisition function to the setting of variable fidelity Bayesian optimization. Using this modified algorithm we show superior performance to the state of the art using common optimization test functions and real problems from machine learning.

In Chapter 5 we again considered variable fidelity Bayesian optimization, this time in the special case of unbiased noise. We showed that the overhead cost and a total computation budget must be considered to obtain a practical result, and developed a model of overhead and performance which allowed us to select the optimum fixed per-step cost with reasonable accuracy. The predictions made are of particular note since they allocate the majority of the total computation budget to overhead costs rather than objective function evaluations.

Finally, in Chapter 6, we considered reducing the cost of the acquisition function. Making use of adaptive quadrature, rather than Monte Carlo integration using slice sampling, to marginalize GP hyperparameters allows us to use cheap $\mathcal{O}(n^2)$ Cholesky updates rather than full $\mathcal{O}(n^3)$ decompositions for a majority of log-likelihood evaluations as optimization progresses. This, combined with performing the inner search of the acquisition function using faster batch evaluations, gives the potential for substantial improvements in execution time without changing the underlying Bayesian optimization method used.

7.2 Directions for Future Work

In our experiments using biased variable fidelity we have considered only adding a single observation to the dataset at each iteration. We mentioned in Chapter 6 the existence of works considering Bayesian optimization with simultaneous evaluations

at multiple locations. However, in the biased variable fidelity setting problems exist for which we could evaluate using multiple fidelities at a reduced cost compared to the sum of the individual costs. For example, several of our test objectives maximize GP hyperparameter log-likelihood by using subsets of the full dataset to control the environmental variable. The dominant cost of evaluation is the $\mathcal{O}(n^3)$ Cholesky decomposition of the covariance matrix. However, once this operation has been performed we can reuse this decomposition to evaluate the log-likelihood of any $m < n$ subset sharing the first m elements of the original subset without performing another Cholesky decomposition. We believe that the key obstacle in considering multiple fidelity evaluations will be balancing the performance gains due to additional cheap information with the faster overhead growth incurred by adding multiple observations to the dataset at each iteration.

A further potential extension to the variable fidelity problem would be to accommodate constrained and multi-objective optimization problems. These are common in the engineering design setting, where the existence of an environmental variable in the form of the coarseness of discretization of space and time for ordinary and partial differential equation solvers provides a clear opportunity to apply a variable fidelity approach.

Using BLOSSOM we have achieved good optimization performance for smooth objectives providing exact evaluations by using BFGS as a local optimizer. However, in the common application of hyperparameter tuning for classification the objective function is piecewise constant with discrete steps in multiples of $\frac{1}{n}$, where n is the size of the test dataset. Unless n is sufficiently large that we can still consider the function smooth in practice we cannot use a local optimizer which relies on gradient and Hessian approximation. Exploitation by Gaussian process methods is even more undesirable in this context than the continuous case once uncertainty in the GP model at a local minimum falls below $\frac{1}{n}$. In this setting clearly no final stage of optimization is required. We suggest replacing the estimation of a convex sphere with determining a region surrounding the posterior mean minimum

for which the GP model predicts a probability less than some ϵ of deviating from the relevant $\frac{1}{n}$ window.

Another case not approached by BLOSSOM is that of noisy observations. In this setting no changes are needed to the global stages of the algorithm, since the GP model already accommodates observations noise (although many more noisy observations will be required to reach low values of global regret). However, BFGS is again no longer a suitable choice so must be replaced with an appropriate noisy local optimizer such as stochastic gradient descent, or CMA-ES constrained within the local basin.

There are several directions for progress available in the unbiased variable fidelity setting. Primarily, since we have used a simple parametric model for both overhead and performance, only considering a single lengthscale hyperparameter, considerable improvement can be expected if more complex regression methods are used. Furthermore, our model for overhead growth is specific to the system used to produce training data. An updated model suitable for practical use would consist of a "complexity" model accounting for the growth of the mathematical requirements of the overhead, modified by a simpler "processor" model that could be trained quickly on specific hardware. Further improvements are likely available since our work in this area has only considered making a single fixed choice for the environmental variable, based on our prior for the objective function hyperparameters. Making a selection online at each step using the hyperparameter posterior may yield better performance. However, since such a decision will necessarily require marginalization over the space of all possible future decisions in a similar manner to multi-step lookahead approaches this is a highly non-trivial proposition.

We have demonstrated in Chapter 6 that there are considerable time saving improvements available to both the marginalization of GP hyperparameters and also maximization of the acquisition function. It remains to implement these changes as low-level code specific to Bayesian Optimization, in order to gain real improvements from the reduced mathematical burden. As we have already noted, little attention is given to the method used for the inner search since this cost is traditionally

considered negligible in comparison to the objective. Our results in Chapter 5 show that this may not always be the true, and therefore motivate further investigation into which of the wide range of global optimization algorithms available may be best suited for optimizing the acquisition function.

Appendices

A

Details about Objective Functions

Contents

We have made use of various functions to benchmark our Bayesian Optimization routines. In this section we provide the details on the implementation of these problems.

A.1 Closed Form Objectives

Branin Function

A two-dimensional objective with three distinct global minima. The search domain is $[0, 1]^2$.

$$f(x) = \left(-1.275(\frac{x_0}{\pi})^2 + \frac{5x_0}{\pi} + x_1 - 6\right)^2 + \left(10. - \frac{5}{4\pi}\right)\cos(x_0) + 10 \qquad \text{(A.1)}$$

Camel 3 Hump Function

A two dimensional objective with a single global minimum and other local minima. The search domain is $[-5, 5]^2$.

$$f(x) = 2x_0^2 - 1.05x_0^4 + \frac{x_0^6}{6} + x_0x_1 + x_1^2 \tag{A.2}$$

Camel 6 Hump Function

A two dimensional objective with a single global minimum and other local minima.

The search domain is $[-3, 3] \times [-2, 2]$.

$$f(x) = \left(4. - 2.1x_0^2 + \frac{x_0^4}{3}\right) x_0^2 + x_0x_1 + \left(-4 + 4x_1^2\right) x_1^2 \tag{A.3}$$

Hartmann 3D Function

A three dimensional objective with a single global minimum and other local minima.

The search domain is $[-5, 10]^3$.

$$\begin{aligned} f(x) &= -\sum_{i=0}^{3} \alpha_i \left(-\sum_{j=0}^{2} A_{ij}(x_j - P_{ij})^2 \right) \\ \alpha &= [1.0, 1.2, 3.0, 3.2] \\ A &= \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 35 \\ 0.1 & 10 & 35 \end{bmatrix} \\ P &= \begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.0381 & 0.5743 & 0.8828 \end{bmatrix} \end{aligned} \tag{A.4}$$

Hartmann 6D Function

A six dimensional objective with a single global minimum and other local minima. The search domain is $[0,1]^6$.

$$
\begin{aligned}
f(x) &= -\sum_{i=0}^{3} \alpha_i \left(-\sum_{j=0}^{5} A_{ij}(x_j - P_{ij})^2 \right) \\
\alpha &= [1.0, 1.2, 3.0, 3.2] \\
A &= \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix} \\
P &= \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.1240 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.3810 \end{bmatrix}
\end{aligned}
\tag{A.5}
$$

Hartmann 4D Function

A four dimensional objective with a single global minimum and other local minima. The search domain is $[0,1]^4$.

$$
f(x) = \frac{1}{0.839} \left(1.1 - \sum_{i=0}^{3} \alpha_i \left(-\sum_{j=0}^{3} A_{ij}(x_j - P_{ij})^2 \right) \right) \tag{A.6}
$$

$\alpha, A,$ and P take the same values as above.

A.2 Machine Learning Objectives

Use of a Gaussian Process as a regression model requires selection (or marginalization) of the kernel hyperparameters to obtain good performance. Maximizing the GP log-likelihood is one common method of selection. However, for large datasets the cost of each evaluation is substantial. We have used this optimization of GP hyperparameters as a test objective for our methods using several datasets. Due to the large dynamic range of negative values for log-likelihoods we transform the original log-likelihood by

$$
f'(x) = \begin{cases} f(x) & x > 0 \\ \log(f(x)) & x \leq 0 \end{cases}
$$

to reduce the dynamic range and hence make the objective functions more suitable for modelling by the Matérn 5/2 kernel. We use GP hyperparameter optimization with these modifications for several of the objectives listed.

UK Power Consumption

We use a dataset of half hourly temperature measurements for UK electricity demand[1] over the 2015 calendar year to provide a 1D dataset for fitting. Log-likelihood evaluations using the full 17,520 elements typically requires around ten minutes. We fix the observation noise to 10^{-9} to avoid conditioning problems and optimize over the shape and scale parameters of the Matérn kernel to give a two dimensional objective.

Combined Cycle Power Plant

We use the Combined Cycle Power Plant dataset sourced from [Dheeru and Karra Taniskidou, 2017] of 9,568 of measurements of temperature, ambient pressure, relative humidity and exhaust vacuum to infer net hourly output of a power plant Tüfekci [2014]. The log-likelihood of hyperparameter values using this dataset typically require 30 seconds to evaluate. We use this data to provide two log-likelihood objective functions, first using noise, shape and scale parameters of the isotropic Matérn 5/2 kernel to provide a three dimensional objective, and second using four shape parameters of the anisotropic form of the kernel to provide a six dimensional objective.

Physiochemical Properties of Protein Tertiary Structure

We use the Physiochemical Properties of Protein Tertiary Structure dataset sourced from [Dheeru and Karra Taniskidou, 2017] of 45,730 of measurements of nine physical parameters to infer residue size. The log-likelihood of hyperparameter values using this dataset typically require 50 minutes to evaluate. We use this data

[1] www2.nationalgrid.com/UK/Industry-information/Electricity-transmission-operational-data/Data-explorer

optimize the diagonal noise, shape and scale parameters of the isotropic Matérn 5/2 kernel to provide a three dimensional objective.

MNIST SVM

MNIST [LeCun and Cortes, 2010] is a popular regression dataset in which the task is to classify 10,000 greyscale images of handwritten digits. We use the support vector machine implementation provided by Pedregosa et al. [2011] to perform this task, which typically requires five minutes to complete using the full dataset. The input space for optimization consists of the error penalty and kernel parameters used in training, and the output is the fraction of correct predictions achieved on a held out test set.

Road Network SVR

We use the 3D Road Network (North Jutland, Denmark) dataset sourced from [Dheeru and Karra Taniskidou, 2017], which provides elevation measurements at 434,874 locations [Kaul et al., 2013]. The support vector regression implementation of Pedregosa et al. [2011] is trained to predict elevation using half of the full dataset, a task which requires approximately 100 minutes to perform. Performance is optimized with respect to the error penalty and kernel parameters to achieve the minimum mean squared prediction error.

B

PES Optimal Cost

Contents

In this Appendix we analyse the effect of variation in the environmental variable for optimization using the PES acquisition function in an unbiased variable fidelity setting.

The original PES acquisition function is the difference in entropy at the query point induced by an observation marginalized over the M samples of the true global minimum

$$\alpha = \mathbb{E}[\Delta H](x) = 0.5 \log(v(x) + \sigma^2) - \frac{1}{M} \sum_{i=1}^{M} 0.5 \log(v_i(x \mid x_i^*) + \sigma^2)\,, \qquad \text{(B.1)}$$

where $v(x)$ is the GP posterior variance at x marginalized over the GP hyper-parameters and $v_i(x \mid x_i^*)$ is the new posterior variance with the i^{th} sampled global minimum, x_i^*, included in the training data and marginalized over the same hyperparameter samples. In a variable fidelity context we divide this quantity by the evaluation cost to obtain a per-unit-cost acquisition function. Since in this section we are not concerned with variations in the query location or hyperparameters

we simplify notation by defining

$$\begin{aligned} v &= v(x) \\ v_i^* &= v_i(x \mid x_i^*) \\ \sigma^2 &= u(s) \end{aligned} \tag{B.2}$$

where $u(s)$ is some function determining the observation variance in terms of the environmental variable s. Defining the cost of taking an evaluation as $c(s)$ the modified acquisition function which seeks the greatest reduction in Entropy per unit cost can now be expressed as

$$\alpha = \frac{0.5}{Mc(s)} \sum_{i=1}^{M} \log\left(\frac{v + u(s)}{v_i^* + u(s)}\right) . \tag{B.3}$$

where the $v(x)$ term has now been included within the summation over sampled global minima.

B.1 Independent Observations

We first consider the case where the environmental variable controls the number n of repeated i.i.d. observations of the objective, with each having variance u_0. We then have $u(s) = \frac{u_0}{n}$ and $c(s) = An$ for some positive constant A. Substituting these into our acquisition and rearranging we have

$$\alpha = \frac{0.5}{AM} \sum_{i=1}^{M} \frac{1}{n} \log\left(\frac{nv + u_0}{nv_i^* + u_0}\right) . \tag{B.4}$$

Ideally we would like to find an value for the n which maximizes α. However,

$$\begin{aligned}
\frac{\partial \alpha}{\partial n} &= \frac{0.5}{AM}\sum_{i=1}^{M}\left(-\frac{1}{n^2}\log\left(\frac{nv+u_0}{nv_i^*+u_0}\right)+\frac{1}{n}\frac{(v-v_i^*)u_0}{(nv_i^*+u_0)(nv+u_0)}\right)\\
&= \frac{0.5}{AM}\sum_{i=1}^{M}\frac{1}{n^2}\left(\frac{n(v-v_i^*)u_0}{(nv_i^*+u_0)(nv+u_0)}-\log\left(1+\frac{n(v-v_i^*)}{nv_i^*+u_0}\right)\right)\\
&\leq \frac{0.5}{AM}\sum_{i=1}^{M}\frac{1}{n^2}\left(\frac{n(v-v_i^*)u_0}{(nv_i^*+u_0)(nv+u_0)}-\frac{\frac{n(v-v_i^*)}{nv_i^*+u_0}}{1+\frac{n(v-v_i^*)}{nv_i^*+u_0}}\right)\\
&= \frac{0.5}{AM}\sum_{i=1}^{M}\frac{1}{n^2}\left(\frac{n(v-v_i^*)u_0}{(nv_i^*+u_0)(nv+u_0)}-\frac{n(v-v_i^*)(nv_i^*+u_0)}{(nv_i^*+u_0)(nv+u_0)}\right)\\
&= \frac{0.5}{AM}\sum_{i=1}^{M}\frac{1}{n^2}\left(\frac{-n^2(v-v_i^*)v_i^*}{(nv_i^*+u_0)(nv+u_0)}\right)\\
&< 0\,.
\end{aligned} \tag{B.5}$$

where we have used the property $\log(1+x) \geq \frac{x}{1+x}$ and $v > v_i^*$ (since v_i^* is an update of v given additional information). Our acquisition is therefore monotonically decreasing, and the least expensive evaluation at $n = 1$ is optimal.

B.2 Correlated Observations

We now consider the more complex case where observations may be made with non-zero covariance structure. Consider n identically distributed but dependant zero-mean observations $\mathcal{N}(0, \Sigma)$, where Σ is any valid covariance matrix with some kernel $k(i, j)$ where $k(i, i) = U$. The sum of off-diagonal elements for size n is

$$g(n) = \sum_{i=1}^{n}\sum_{j=i+1}^{n} 2k(i, j)\,. \tag{B.6}$$

and the variance of the mean of the n observations is given by

$$u(n) = \frac{Un + g(n)}{n^2}\,. \tag{B.7}$$

Our per-unit cost acquisition function and its derivative are now

$$
\begin{aligned}
\alpha &= \frac{0.5}{AM}\sum_{i=1}^{M}\frac{1}{n}\log\left(\frac{n^2v+Un+g(n)}{n^2v_i^*+Un+g(n)}\right)\\
\frac{\partial\alpha}{\partial n} &= \frac{0.5}{AM}\sum_{i=1}^{M}\Bigg(-\frac{1}{n^2}\log\left(\frac{n^2v+Un+g(n)}{n^2v_i^*+Un+g(n)}\right)\\
&\qquad\qquad + \frac{1}{n}\frac{(v-v_i^*)(2ng(n)-n^2g'(n)+n^2U)}{(n^2v+Un+g(n))(n^2v_i^*+Un+g(n))}\Bigg)\\
&= \frac{0.5}{AM}\sum_{i=1}^{M}\frac{1}{n^2}\Bigg(\frac{(v-v_i^*)(2g(n)-ng'(n)+nU)}{(n^2v+Un+g(n))(n^2v_i^*+Un+g(n))}\\
&\qquad\qquad - \log\left(1+\frac{n^2(v-v_i^*)}{n^2v_i^*+Un+g(n)}\right)\Bigg)\\
&\leq \frac{0.5}{AM}\sum_{i=1}^{M}\frac{1}{n^2}\left(\frac{(v-v_i^*)(g(n)-ng'(n)+nU)}{(n^2v+Un+g(n))(n^2v_i^*+Un+g(n))} - \frac{\frac{n^2(v-v_i^*)}{n^2v_i^*+Un+g(n)}}{1+\frac{n^2(v-v_i^*)}{n^2v_i^*+Un+g(n)}}\right)\\
&= \frac{0.5}{AM}\sum_{i=1}^{M}\left(\frac{(v-v_i^*)(n^2v_i^*-g(n)+ng'(n))}{(n^2v+Un+g(n))(n^2v_i^*+Un+g(n))}\right).
\end{aligned}
\tag{B.8}
$$

This is less than zero if $n^2v_i^* - g(n) + ng'(n) > 0$. This trivially includes the independent case $g(n) = g'(n) = 0$. If we restrict Σ to be a Toeplitz matrix defined by some series such that the elements on the ith diagonal are $a_i = k(i,0)$, with $a_0 = U$, we can consider $g'(n)$ as a piecewise constant function, defined as the sum of step functions of $a_iH(n-i)$ at each $i \in \mathbb{N}$. We then have

$$
\begin{aligned}
g(n) &= \int_1^n \sum_{i=1}^{n} 2a_iH(n-i)dn\\
&= \sum_{i=1}^{n} 2a_i(n-i)H(n-i)
\end{aligned}
\tag{B.9}
$$

where the constant arising from integration is zero to satisfy $g(1) = 0$. Substituting these terms we have

$$
\begin{aligned}
&n^2v_i^* - g(n) + ng'(n)\\
&= n^2v_i^* - \sum_{i=1}^{n} 2a_i(n-i)H(n-i) + n\sum_{i=1}^{n} 2a_iH(n-i)\\
&= n^2v_i^* + \sum_{i=1}^{n} 2a_iiH(n-i)
\end{aligned}
\tag{B.10}
$$

which is positive for all Σ with strictly positive entries, and all series of a_i satisfying $\sum ia_i > 0 \quad \forall i$.

Correlations between observations that satisfy this condition therefore induce a strictly decreasing acquisition with respect to batch size, and a single observation is the optimum choice, as in the independent case. Note that since we have used an inequality in Equation B.8 this is sufficient condition for a monotonicity with respect to the environmental variable. Scenarios with negative correlations between observations that do not meet this condition may still have a single observation as the optimum choice.

Bibliography

Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Courier Corporation, 1964. URL http://people.math.sfu.ca/~cbm/aands/abramowitz_and_stegun.pdf.

Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002. URL http://www.jmlr.org/papers/volume3/auer02a/auer02a.pdf.

Javad Azimi, Ali Jalali, and Xiaoli Fern. Hybrid Batch Bayesian Optimization. *arXiv:1202.5597 [cs]*, February 2012. URL http://arxiv.org/abs/1202.5597.

Simeon M. Berman and Norio Kono. The maximum of a Gaussian process with nonconstant variance: a sharp bound for the distribution tail. *The Annals of Probability*, pages 632–650, 1989. URL http://www.jstor.org/stable/2244285.

Adam D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct):2879–2904, 2011. URL http://www.jmlr.org/papers/v12/bull11a.html.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, February 2016. URL http://link.springer.com/10.1007/s10472-015-9463-9.

Dua Dheeru and Efi Karra Taniskidou. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2017. URL http://archive.ics.uci.edu/ml.

T.W. Doeppner. *Operating Systems In Depth: Design and Programming*. John Wiley & Sons, 2011. ISBN 978-1-118-13639-3. URL https://books.google.co.uk/books?id=xX5tfrAQQ8cC.

David Duvenaud. Automatic model construction with Gaussian processes. (June), 2014. URL https://www.repository.cam.ac.uk/handle/1810/247281.

Alexander I. J. Forrester, András Sóbester, and A. J. Keane. *Engineering design via surrogate modelling: a practical guide*. J. Wiley, Chichester, West Sussex, England ; Hoboken, NJ, 2008. ISBN 978-0-470-06068-1. URL https://onlinelibrary.wiley.com/doi/book/10.1002/9780470770801.

Alexander I.J. Forrester, András Sóbester, and Andy J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463(2088):3251–3269, December 2007. URL http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.2007.1900.

Roman Garnett, Michael A. Osborne, and Stephen J. Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219. ACM, 2010. URL http://www.robots.ox.ac.uk/~mosb/public/pdf/1242/ipsn673-garnett.pdf.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. Technical report, March 2008. URL https://hal.archives-ouvertes.fr/hal-00260579.

J. González, Z. Dai, P. Hennig, and N. Lawrence. Batch Bayesian Optimization via Local Penalization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS 2016)*, volume 51 of *JMLR Workshop and Conference Proceedings*, pages 648–657, 2016a. URL http://jmlr.org/proceedings/papers/v51/gonzalez16a.pdf.

Javier González, Michael Osborne, and Neil Lawrence. GLASSES: Relieving the myopia of Bayesian optimisation. In *Artificial Intelligence and Statistics*, pages 790–799, 2016b. URL https://bayesopt.github.io/papers/2015/gonzalez.pdf.

Tom Gunter, Michael A. Osborne, Roman Garnett, Philipp Hennig, and Stephen J. Roberts. Sampling for inference in probabilistic models with fast Bayesian quadrature. In *Advances in neural information processing systems*, pages 2789–2797, 2014. URL https://papers.nips.cc/paper/5483-sampling-for-inference-in-probabilistic-models-with-fast-bayesian-quadrature.pdf.

Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, June 2001. ISSN 1063-6560, 1530-9304. doi: 10.1162/106365601750190398. URL http://www.mitpressjournals.org/doi/10.1162/106365601750190398.

Phillipp Hennig and Christian J Schuler. Entropy Search for Information-Efficient Global Optimization. *Machine Learning Research*, 13(1999):1809–1837, 2012. URL http://jmlr.csail.mit.edu/papers/volume13/hennig12a/hennig12a.pdf.

Daniel Hernández-Lobato, José Miguel Hernández-Lobato, Amar Shah, and Ryan P Adams. Predictive Entropy Search for Multi-objective Bayesian Optimization. *arXiv preprint arXiv:1511.05467*, 2015. URL http://proceedings.mlr.press/v48/hernandez-lobatoa16.pdf.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. *Advances in Neural Information Processing Systems 28*, pages 1–9, 2014. URL https://jmhldotorg.files.wordpress.com/2014/10/pes-final.pdf.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *Proceedings of the conference on Learning and Intelligent OptimizatioN (LION 5)*, pages 507–523, January 2011. URL http://ml.informatik.uni-freiburg.de/papers/11-LION5-SMAC.pdf.

Carlos M Jarque and Anil K Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics letters*, 6(3):255–259, 1980.

Edwin T Jaynes, G. Larry Bretthorst, and EBSCO Publishing. *Probability theory: the logic of science*. Cambridge University Press, Cambridge, 2003. ISBN 978-0-511-06589-7. URL http://www5.unitn.it/Biblioteca/it/Web/LibriElettroniciDettaglio/50847.

Donald Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001. ISSN 0925-5001. doi: 10.1023/a:1012771025575. URL http://dx.doi.org/10.1023/a:1012771025575.

Donald R Jones, Matthias Schonlau, and J William. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13:455–492, 1998. URL http://www.ressources-actuarielles.net/EXT/ISFA/1226.nsf/0/f84f7ac703bf5862c12576d8002f5259/$FILE/Jones98.pdf.

M. Kaul, B. Yang, and C. S. Jensen. Building Accurate 3d Spatial Networks to Enable Next Generation Intelligent Transportation Systems. In *2013 IEEE 14th International Conference on Mobile Data Management*, volume 1, pages 137–146, June 2013. URL https://www.iith.ac.in/~mkaul/papers/mdm.pdf.

A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the AISTATS conference*, 2017. URL http://proceedings.mlr.press/v54/klein17a/klein17a.pdf.

Harold J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964. URL http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1431594.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Daniel J. Lizotte, Tao Wang, Michael H. Bowling, and Dale Schuurmans. Automatic Gait Optimization with Gaussian Process Regression. In *IJCAI*, volume 7, pages 944–949, 2007. URL http://papersdb.cs.ualberta.ca/~papersdb/uploaded_files/352/additional_IJCAI07-152.pdf.

Romy Lorenz, Ricardo P. Monti, Ines R. Violante, Aldo A. Faisal, Christoforos Anagnostopoulos, Robert Leech, and Giovanni Montana. Stopping criteria for boosting automatic experimental design using real-time fMRI with Bayesian optimization. *arXiv preprint arXiv:1511.07827*, 2015. URL https://arxiv.org/abs/1511.07827.

Tom Minka. Expecatation Propagation for Approximate Bayesian Inference. *Conference on Uncertainty in Artificial Intelligence*, pages 362–369, 2001. URL http://arxiv.org/pdf/1301.2294.

Hossein Mohammadi, Rodolphe Le Riche, and Eric Touboul. Making EGO and CMA-ES Complementary for Global Optimization. In Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eléonore Marmion, editors, *Learning and Intelligent Optimization*, pages 287–292, Cham, 2015. Springer International Publishing. URL https://link.springer.com/chapter/10.1007/978-3-319-19084-6_29.

Kevin P. Murphy. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012. ISBN 978-0-262-01802-9. URL https://www.cs.ubc.ca/~murphyk/MLbook/.

Iain Murray and Ryan P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *Advances in Neural Information Processing Systems*, pages 1732–1740, 2010. URL http://homepages.inf.ed.ac.uk/imurray2/pub/10hypers/hypers.pdf.

Radford M. Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003. URL https://projecteuclid.org/download/pdf_1/euclid.aos/1056562461.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006. URL http://www.springer.com/gb/book/9780387303031.

Michael A. Osborne, Stephen J. Roberts, Alex Rogers, Sarvapali D. Ramchurn, and Nicholas R. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 109–120. IEEE Computer Society, 2008. URL http://www.robots.ox.ac.uk/~parg/pubs/osborne-GaussianProcesses.pdf.

Michael A Osborne, Roman Garnett, and Stephen J Roberts. Gaussian Processes for Global Optimization. *3rd International Conference on Learning and Intelligent Optimization LION3*, (x):1–15, 2009. URL http://www.robots.ox.ac.uk/~mosb/OsborneGarnettRobertsGPGO.pdf.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007. URL https://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

Carl Edward Rasmussen and Zoubin Ghahramani. Bayesian Monte Carlo. *Advances in Neural Information Processing Systems 15*, (1):489–496, 2003. URL http://mlg.eng.cam.ac.uk/zoubin/papers/RasGha03.pdf.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9. URL http://www.gaussianprocess.org/gpml/chapters/RW.pdf. OCLC: ocm61285753.

Luis Miguel Rios. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56 (3):1247–1293, July 2013. ISSN 1573-2916. doi: 10.1007/s10898-012-9951-y. URL https://doi.org/10.1007/s10898-012-9951-y.

Rudolf Schürer. A comparison between (quasi-)Monte Carlo and cubature rule based methods for solving high-dimensional integration problems. *Mathematics and Computers in Simulation*, page 9, 2003.

Warren Scott, Peter Frazier, and Warren Powell. The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression. *SIAM Journal on Optimization*, 21(3):996–1026, July 2011. ISSN 1052-6234, 1095-7189. doi: 10.1137/100801275. URL http://epubs.siam.org/doi/10.1137/100801275.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, January 2016. URL https://www.cs.ox.ac.uk/people/nando.defreitas/publications/BayesOptLoop.pdf.

Claude E Shannon, Warren Weaver, and Arthur W Burks. The mathematical theory of communication. 1951.

Steven H. Simon, Aris L. Moustakas, and Luca Marinelli. Capacity and Character Expansions: Moment-Generating Function and Other Exact Results for MIMO Correlated Channels. *IEEE Transactions on Information Theory*, 52(12):5336–5351, December 2006. ISSN 0018-9448, 1557-9654. doi: 10.1109/TIT.2006.885519. URL https://ieeexplore.ieee.org/document/4016317/.

Edward Snelson, Zoubin Ghahramani, and Carl E. Rasmussen. Warped gaussian processes. In *Advances in neural information processing systems*, pages 337–344, 2004. URL http://www.gatsby.ucl.ac.uk/~snelson/gpwarp.pdf.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012. URL https://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms.pdf.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian Optimization Using Deep Neural Networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2171–2180, Lille, France, July 2015. PMLR. URL http://proceedings.mlr.press/v37/snoek15.html.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. URL http://www-stat.wharton.upenn.edu/~skakade/papers/ml/bandit_GP_icml.pdf.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. *Multi-Task Bayesian Optimization*. 2013. URL http://papers.nips.cc/paper/5086-multi-task-bayesian-optimization.pdf.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014. URL http://arxiv.org/abs/1406.3896.

Matthew Tesch, Jeff Schneider, and Howie Choset. Using response surfaces and expected improvement to optimize snake robot gait parameters. *IEEE International Conference on Intelligent Robots and Systems*, pages 1069–1074, 2011a. URL https://www.cs.cmu.edu/~schneide/IROS11snake.pdf.

Matthew Tesch, Jeff Schneider, and Howie Choset. Using response surfaces and expected improvement to optimize snake robot gait parameters. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1069–1074. IEEE, 2011b. URL https://www.cs.cmu.edu/~schneide/IROS11snake.pdf.

Pınar Tüfekci. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126 – 140, 2014. URL http://www.sciencedirect.com/science/article/pii/S0142061514000908.

Paul Van Dooren and Luc de Ridder. An adaptive algorithm for numerical integration over an N-dimensional cube. *Journal of Computational and Applied Mathematics*, 2(3):207–217, 1976. URL https://www.sciencedirect.com/science/article/pii/0771050X7690005X.

Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509–534, August 2009. ISSN 0925-5001, 1573-2916. doi: 10.1007/s10898-008-9354-2. URL http://link.springer.com/10.1007/s10898-008-9354-2.

Kim Peter Wabersich and Marc Toussaint. Advancing Bayesian Optimization: The Mixed-Global-Local (MGL) Kernel and Length-Scale Cool Down. *arXiv preprint arXiv:1612.03117*, 2016. URL https://arxiv.org/abs/1612.03117.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016. URL http://www.jair.org/papers/paper4806.html.

Brian J. Williams, Thomas J. Santner, and William I. Notz. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, pages 1133–1152, 2000. URL http://www.jstor.org/stable/24306770.