Durham University

Constrained Bayesian Optimization

Author:
Adam Howes

Supervisor:
Dr. Georgios Karagiannis

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Mathematics

April 26, 2018

Declaration of Authorship

I, Adam Howes, declare that this project titled, "Constrained Bayesian Optimization" is a result of my own work except where it forms an assessment based on group project work. In the case of a group project, the work has been prepared in collaboration with other members of the group. Material from the work of others not involved in the project has been acknowledged and quotations and paraphrases suitably indicated. Code for all figures and experiments in this project is available at `https://github.com/athowes/cbo`.

Signed:

Date:

Abstract

This project introduces the reader to Bayesian optimization, a sequential algorithm for the global optimization of expensive-to-evaluate, black-box functions. We also present some recent research in extending Bayesian optimization to deal with constrained optimization problems.

"*Is this Bayesian? You know I'm a strict Bayesian, right?*"

@ML_Hipster

Acknowledgements

Thanks to Georgios for his guidance and support throughout.

Contents

Chapter 1

Background

1.1 Problem definition

Optimization, the act of finding the best action in a given situation, is both a ubiquitous problem and one of a great importance. Mathematically, the *global* optimization of a real-valued function $f : \mathcal{X} \to \mathbb{R}$ aims to find a minimizer $\mathbf{x}^\star$ (there may be more than one) in the search space $\mathcal{X}$, such that:

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

Considering $g(\mathbf{x}) = -f(\mathbf{x})$ shows, as would be expected, that maximization is equivalent. The function f to be optimized is referred to as the *objective function*. In contrast to local optimization, global optimization requires that $f(\mathbf{x}^\star) \leq f(\mathbf{x})$ for *all* $\mathbf{x} \in \mathcal{X}$ rather than only in some neighbourhood about $\mathbf{x}^\star$. In this project, we assume that the search space $\mathcal{X}$ is a compact subset of $\mathbb{R}^d$ where $d \in \mathbb{N}$.

This problem, given by equation 1.1, is the subject of a vast array of scientific literature. In practise, the objective function f may additionally have the following challenging properties P1 - P4:

P1 **Non-linear, non-convex**: In general, the objective function cannot be assumed to be linear or, more broadly, convex. The extensive fields of linear programming and convex optimization study these cases respectively. Rockafellar [1] famously describes that in truth the "great watershed in optimization" lies between the difficulty of the convex and non-convex cases.

P2 **Black-box**: A function is called *black-box* if it can only be viewed in terms of its inputs and outputs. If f is black-box then it does not have analytic form

or derivatives, such as the gradient ∇f or Hessian $\mathbf{H}$, which can be reasoned about and, in particular, used in the search for $\mathbf{x}^*$. As Golovin, Solnik, Moitra, *et al.* [2] note in their paper about Google's in-house black-box optimizer:

> "Any sufficiently complex system acts as a black-box when it becomes easier to experiment with than to understand"

To this end, they suggest that as the systems we create, and hope to optimize, increase in complexity, black-box optimization techniques will become increasingly important. For problems with this property, optimization may only proceed by querying f point-wise at inputs $\mathbf{x} \in \mathcal{X}$ in order to learn about its behaviour.

P3 **Expensive to evaluate**: The sampling procedure is computationally, economically or otherwise prohibitively expensive. Most prevalently the objective function is time-consuming to evaluate. For example, when tuning the hyperparameters of a machine-learning model, each function evaluation is potentially very time-intensive as it corresponds to training and evaluating the performance of the primary model [3]. That being said, there are many examples of more general costs. In chemistry, in order to optimize the yield of a reaction, evaluating the objective function involves potentially resource-intensive and laborious experiments. No matter the resource in question, practitioners have finite budgets and, as a result, may only be able to afford relatively few function evaluations. Often it is specified that the number of function evaluations which can be afforded is N- which we will take to be the case in this project. More general formulations may, among other considerations, take into account the variable costs of sampling at different locations [3].

P4 **Noisy**: When f is evaluated at $\mathbf{x}$, the value returned y is contaminated by noise ϵ, assumed to be Gaussian with zero mean and variance σ^2 such that $y = f(\mathbf{x}) + \epsilon$. Problems of this sort are described as *stochastic programming* problems.

This project will begin in Chapters 1, 2 and 3 with an introduction to *Bayesian optimization*, which is a state-of-the-art method intended for optimizing objective functions with the above properties.

Bayesian optimization and related approaches have a rich history. Their popularity among practitioners and researchers is steadily increasing [4].

1.2 Pragmatic aims

Solving equation 1.1, namely finding a minimizer $\mathbf{x}^\star$, with the additional challenges P1-P3 is a difficult undertaking (leaving P4 a moment). If it is assumed that f is Lipschitz-continuous, that is

$$\exists l \geq 0 \quad \text{s.t.} \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \quad |f(\mathbf{x}) - f(\mathbf{x}')| \leq l|\mathbf{x} - \mathbf{x}'|,$$

then it can be guaranteed that a minimizer $\mathbf{x}^\star$ exists [5]. However, finding it in general is not possible using a finite number of function evaluations [6]. Even guaranteeing a solution evaluates to within ϵ of the true maximum $f(\mathbf{x}^\star)$, which we will call *ϵ-optimality*, requires a minimax approach where a grid of samples are made with resolution proportional to ϵ [7]. Sometime called the "curse of dimensionality", the number of points in such a grid scales exponentially with the dimension d, which quickly becomes impractical even in low dimensions due to property P2. The situation is only made worse if evaluating the objective function is also noisy, P4. For this reason, some clarification is required on the aims of optimization in the face of what is considerable adversity.

Lizotte [6] proposes a principle of perseverance, noting that although the problem is difficult that doesn't make it go away. The world is full of difficult optimization problems that we would still like to make progress on. The good news is that overwhelmingly the task is not all-or-nothing. That is, suboptimal solutions may be perfectly viable if the contextual performance difference is small. In some situations, perhaps when sampling has a meaningful financial cost for example, a cheap suboptimal solution may even be preferred. Furthermore, the previous minimax approach pays a high price to insure against worst-case, pathological scenarios which may not be very plausible in reality [8].

These considerations motivate a "practical" [6] or "average-case" [9] approach, where the focus is in finding a "good" solution or converging on a minimizer $\mathbf{x}^\star$ in few evaluations, rather than in making theoretical guarantees about optimality. This is the philosophy that we will adopt during this project.

1.3 Sample designs

How, then, should points be queried to efficiently learn about $\mathbf{x}^\star$ so that the above aims can be met?

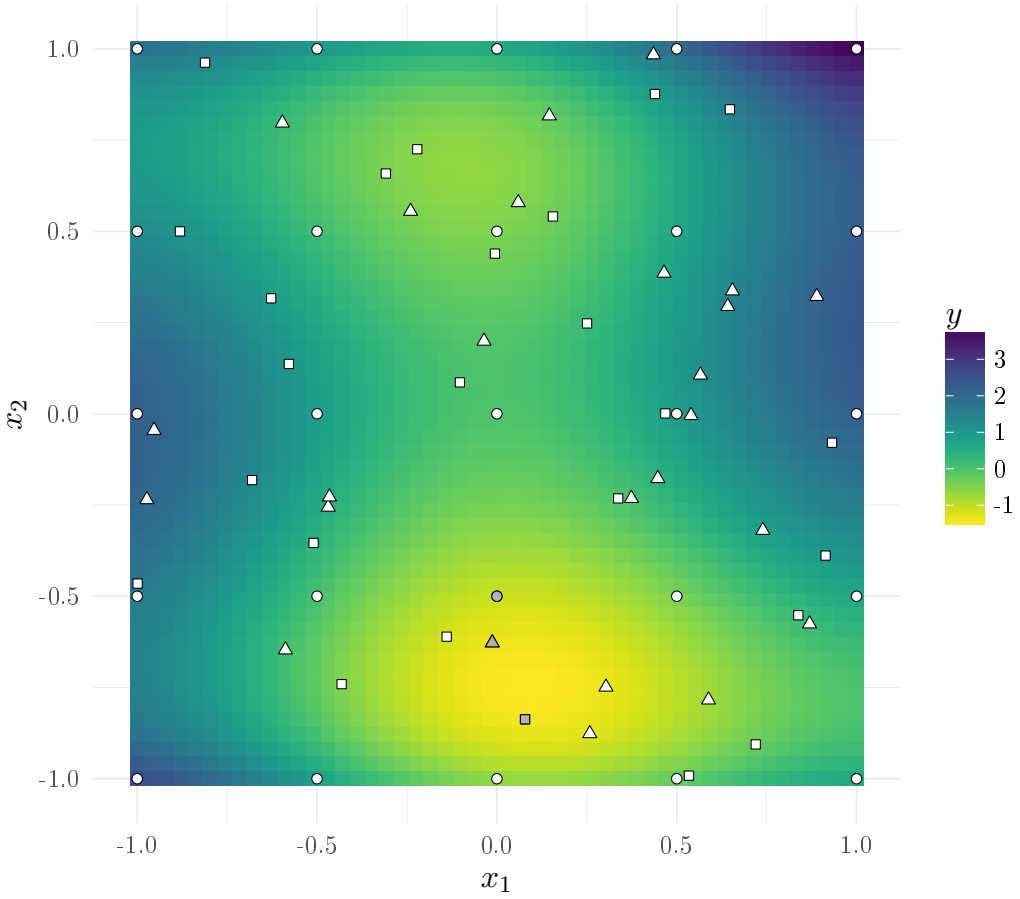
We first present two relatively naive strategies: *grid-search* and *random-search*. These approaches directly select a set of points, called a sample design, to be evaluated. Grid-search is previously mentioned above, where a dense grid was used by Betrò [7] to ensure ϵ-optimality. Of course, the resolution need not be so high. More generally, in a grid-search samples are taken spaced evenly throughout the domain $\mathcal{X}$ at a resolution appropriate to the optimization budget. Although the whole domain is superficially covered, if few function evaluations can be afforded then this coverage is too sparse to reliably locate a minimizer. Meanwhile, random-search simply chooses inputs in the domain $\mathcal{X}$ to evaluate at random.

Perhaps counter-intuitively, Bergstra and Bengio [10] show that in some sense random-search actually performs more efficiently than grid-search. In particular, grid-search performs very poorly when the objective function f has a low effective dimensionality. This is because multiple sample points have the same value when collapsed down onto any dimension. Looking at it another way, if one of the dimensions has little impact then in a grid-search many samples are entirely wasted - without learning *anything* about the other dimensions. That being said, random-search is also severely flawed: complete randomness lends itself to clumps of points and large areas left empty.

One way of overcoming these problems is to use a space-filling, but still random, design such as the *latin-hypercube* [11]. Latin-hypercube designs generalize the latin-square, which is a grid with exactly one sample in each column and each row, to arbitrary dimension. This avoids the problem of collapsing, from which grid-search suffers. Figure 1.1 gives an example of the usage of each of grid-search, random-search and latin-hypercube sampling on a two dimensional objective function.

1.4 Models & trade-offs

Although there are many more complicated and arguably better sample designs than the ones described above, it is intuitively clear that any such strategy is suboptimal: information gained during the search is not used to better inform the next decision. Rather than choose all the points at once it makes sense instead to consider a sequential decision making problem where at each stage a point is carefully chosen to be evaluated next. The same philosophy is true of many situations. It would be unwise to require students starting at Durham to choose their modules for first, second and third year all at once as their preferences will surely change as they learn more.



	Symbol	y_{min}	(x_1, x_2)
Grid-search	Circles	-1.050	$(0.000, -0.500)$
Random-search	Triangles	-1.322	$(0.131, -0.638)$
Latin-hypercube	Squares	-1.382	$(0.078 - 0.837)$

FIGURE 1.1: Density plot of modified two-dimensional camel function $y = (4 - 2.1x_1^2 + 0.3x_1^4)x_1^2 + (x_1 + 0.6)x_2 + (-4 + 4x_2^2)x_2^2$ on the square $x_1 \in [-1, 1], x_2 \in [-1, 1]$. 25 points sampled from each of grid-search, random-search and latin-hypercube sampling; minimum found by each method shown in grey. True global minimum is -1.470 at $(0.094, -0.747)$.

So, how can this information be used? It is important to note that in order to learn about the behaviour of the objective function where it has not been specifically sampled we must make some assumptions about how the data we have may be extrapolated. In particular, we assume that points close in input are to some extent close in output. In other words, that f is somewhat smooth. Making assumptions, like this one, about the nature of the objective function is helpful if those assumptions are right. If they are not too strong or restrictive, then *on-average* they will be right [12]. This is the foundational idea of the model-based approach to optimization, in which statistical inference can be used to aid in the search process.

Assuming that the objective function f is smooth, there are two types of location which should be the focus of the search. Firstly, sampling in uncertain regions, not near to any previous evaluations, will provide helpful information about the global behaviour of the objective function. On the other hand, sampling in promising regions, near to previous low evaluations, will allow solutions to be refined locally. We will term these two contrasting ideas as *global search* and *local search*, respectively. This is slightly non-standard terminology, the reason for which will be explained shortly in section 1.6.

Good strategies then must appropriately balance global search and local search. Too great a focus on global search leads to the optimization never converging on a solution. Conversely, too great a focus on local search tends to locate local rather than global minimizers.

The sample designs given in section 1.3 are all examples of strategies which neglect local search. Once they find a promising location they would be better served to invest more of their attention in that area - so that they may informally "drill-down" [11] to a better solution.

On the other hand, for example techniques which iteratively alter solutions in order to get an idea of the gradient, with the idea being to move in the direction of greatest decrease, have too great a focus on local-search. A ball on the surface of figure 1.1 will roll into the local minima (rather than the global minima) if is placed too close - in the so-called *basin of attraction*. In practise this problem may be remedied by random restarts, i.e. repeating the search a number of times with the ball placed randomly each time. However, this alteration is not viable in view of property P4. The fact that the local minima is a global minima when the objective function is convex is what makes convex optimization significantly more approachable than the general case.

1.5 Bayesian optimization

Bayesian optimization [13] [14] aims at addressing the optimization of objective functions with the above properties P1-P3 and possibly P4. It is a sequential, model-based optimization algorithm which learns from all available observations to make better informed choices about the next point to query. The algorithm is detailed in algorithm 1 and has two key components: the statistical *emulator* and the *acquisition function*.

As f is black-box, there is uncertainty about its true value where it has not been directly evaluated (and when f is noisy then even at evaluated points as well). It is theoretically justified [15] that Bayesian probability theory is a rational way to reason about such uncertain states of knowledge. Prior beliefs about the objective function and data-generation mechanism may be encapsulated with an explicit, probabilistic, statistical model.

This probabilistic model is known as an emulator [11]. It should behave similarly to the objective function f, but be much cheaper to work with. The emulator provides a predictive distribution for f evaluated at new inputs. The mean of this distribution may be used as a *surrogate* for the objective function and the variance allows for uncertainty quantification. When data $\{\mathbf{x}, y\}$ is observed then the model can be sequentially updated and refined, using Bayes' rule.

Typically the emulator used is a *Gaussian process*, which is a convenient and flexible model for this purpose. We will introduce Gaussian processes and their role in Bayesian optimization in more detail in Chapter 2.

To guide the search for a global minimizer, the emulator is leveraged by an acquisition function, sometimes also called an in-fill function. It is a function of the state of the optimization at the current stage n, i.e. all of the data which is available and the Gaussian process emulator to the objective function f, which returns a suggestion of the next point to sample $\mathbf{x}_n$. This function automatically balances the trade-off between global and local search. Many possible acquisition functions have been proposed in the literature. This will be explored in more depth in Chapter 3.

Figure 1.2 gives a toy example of the use of Bayesian optimization for a one-dimensional objective function, illustrating stages $n = 1, 2$ in algorithm 1. In this example, the optimization begins with an already available data-set $\mathcal{D}_0$ which is used to build the inital Gaussian process emulator. It has been suggested that in general a small number of points, depending on the dimension, should be sampled prior to the start of

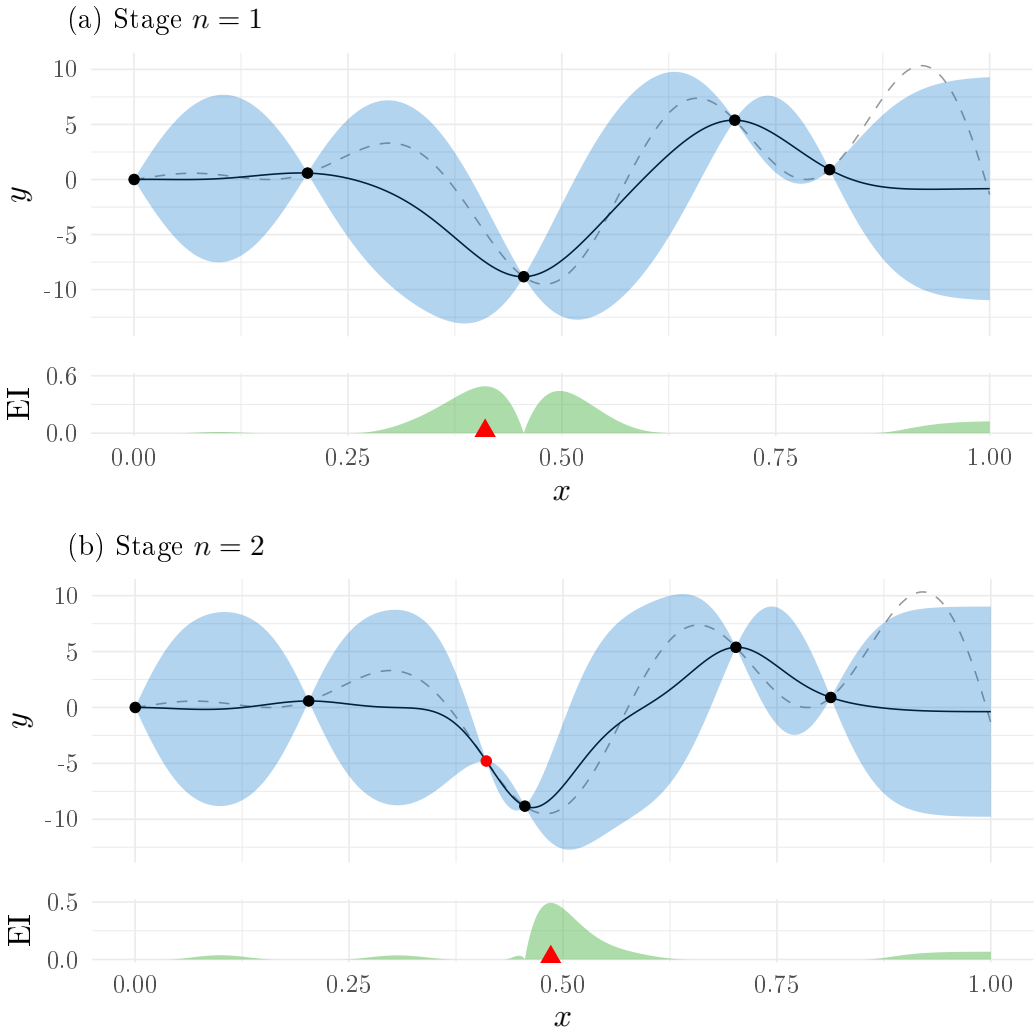


FIGURE 1.2: A two-stage example run of Bayesian optimization. Objective function $f(x) = 10x(\sin(10x) + \cos(20x))$ is shown as the dahsed grey line in both (a) and (b). The mean of the Gaussian process emulator, i.e. the surrogate function, at each stage is shown as the solid black line. Uncertainty about the objective function is shown as the light-blue point-wise 95% confidence region. The initial data-set $\mathcal{D}_0$ is five points and their noise-free evaluations generated by latin hyper-cube sampling. The acquisition function is shown in green, with its maximum at each stage shown by the red arrow. Here we use the one particular acquisition function, expected improvement (EI), which will be properly introduced in Chapter 3. The acquisition function maximum in (a) is shown evaluated as the red circle in (b).

Algorithm 1: Bayesian optimization algorithm

input objective function f, acquisition function α, initial dataset $\mathcal{D}_0$;
for $n = 1, 2, \ldots, N$ **do**
  Fit Gaussian process to dataset at stage n, $\mathcal{D}_{n-1}$;
  Maximize acquisition function α over Gaussian process to propose next point to sample, $\mathbf{x}_n$;
  Query objective function f at $\mathbf{x}_n$, returning $y_n = f(\mathbf{x}_n) + \epsilon$;
  Augment dataset $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}_n, y_n\}$;
end
Fit Gaussian process to complete dataset $\mathcal{D}_N$;
return *recommendation* $\mathbf{x}^-$;

sequential optimization [16]. After the budget has been spent, the algorithm returns a *recommendation* $\mathbf{x}^-$ which it is hoped is a global minimizer. The recommendation is typically the best performing $\mathbf{x}_n$ or the minimizer of the final surrogate function.

One thing that should be clarified is that when the properties stated, particularly P3, are *not* the case then it is more than likely that another algorithm will perform better than Bayesian optimization. It is a topic of recent research extending Bayesian optimization to make use of additional information, such as gradients [6], if it is available.

1.6 Bandits

A similar premise to the problem we have been considering is the so-called *multi-armed bandit* problem [17]. Imagine that a gambler has just one night at a casino and a number of slot-machines (also known as bandits) which can be played. At each stage the gambler must choose an arm to play, with each arm having different odds of paying out. How should the gambler select arms to play in order to maximize their return over the night?

To provide a broader context for Bayesian optimization, we briefly discuss some of the differences between the two problems.

As rewards can be collected at each stage, the multi-armed bandit problem is what is known as an online learning problem [18]. In the Bayesian optimization setting we are typically only concerned with the performance of the final recommendation

of the algorithm $\mathbf{x}^-$, which is contrast is referred to as offline learning. That being said, there is some recent research into online Bayesian optimization, for instance the optimization of financial portfolios [19].

The distinction between on and offline learning is similar to that between summative and formative assessed work. In online learning problems the agent must trade-off between explorative behaviour which sacrifices immediate rewards in the hope of longer-term gains and exploitative, myopic behaviour which greedily aims to do the best it can at the present iteration [20]. For example, in the multi-armed bandit problem choosing to play a new unknown, possibly poor, arm is explorative behaviour, whereas playing the arm with the best odds found so far is exploitative behaviour. Many authors use the terminology exploration and exploitation to refer to the what we have termed called global and local search.

Another difference is that the bandit setting is discrete, whereas we consider the domain $\mathcal{X}$ to be continuous. In addition, it is not assumed that the arms have spatial interpretation, i.e. that the behaviour of a given arm can be inferred from that of adjacent arms, whereas we assume that function evaluations are informative for nearby locations.

Finally, the bandit literature often places a greater emphasis on long-run convergence rates and theory than Bayesian optimization [20] which, as we discuss in section 1.2, often can't afford to make such guarantees and is more pragmatic as a result.

Chapter 2

Gaussian processes

In Chapter 1 we introduced the problem and the Bayesian optimization approach to solving it, using Gaussian processes (GPs) as emulators. In this chapter we will discuss the GP and show how it may be used to model the objective function f.

Intuitively, GPs are a generalization of the multivariate Gaussian distribution to infinite dimension. As such, GPs inherit many helpful properties from the multivariate Gaussian, which we review here.

2.1 Multivariate Gaussian distribution

The random vector $\mathbf{x} = (x_1, ..., x_d)^T \in \mathbb{R}^d$ has multivariate Gaussian distribution with mean vector $\mathbf{m}$ and covariance matrix Σ if its probability density is given by:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{q/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m})\} \tag{2.1}$$

This is denoted by $\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \Sigma)$ [21].

The following (non-trivial) closure properties [22] of the multivariate Gaussian make it particularly convenient for modelling:

G1 Marginal distributions of any subset of $\mathbf{x}$ are multivariate Gaussian. Consider partitioning $\mathbf{x}$ into $[\mathbf{x}_1, \mathbf{x}_2]$ where $\mathbf{x}_1 \in \mathbb{R}^p$ and $\mathbf{x}_2 \in \mathbb{R}^q$ with $p + q = d$, so that

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{m}_1 \\ \mathbf{m}_2\end{bmatrix}, \begin{bmatrix}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22}\end{bmatrix}\right) \tag{2.2}$$

where $\mathbf{m}$ and Σ are partitioned naturally. Then the marginal distribution of $\mathbf{x}_1$ is:

$$p(\mathbf{x}_1) = \mathcal{N}(\mathbf{x}_1|\mathbf{m}_1, \Sigma_{11}) \tag{2.3}$$

G2 Similarly, the conditional distribution of any subset of $\mathbf{x}$ conditioned another subset is multivariate Gaussian. The distribution of $\mathbf{x}_1$ given that $\mathbf{x}_2$ is known is given by:

$$p(\mathbf{x}_1|\mathbf{x}_2) = \mathcal{N}(\mathbf{x}_1|\mathbf{m}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \mathbf{m}_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}) \tag{2.4}$$

2.2 Gaussian process definition

A *Gaussian process* is defined to be a collection of random variables, any finite number of which have a joint Gaussian distribution [23]. The random variables $f(\mathbf{x})$ are indexed by the elements $\mathbf{x}$ of some set $\mathcal{X}$.

As we have seen, the multivariate Gaussian distribution defined by equation 2.1 is specified entirely by its mean vector $\mathbf{m}$ and covariance matrix Σ. Analogously, a GP is specified entirely by its prior mean function $\mu_0 : \mathcal{X} \to \mathbb{R}$ and covariance function $k : \mathcal{X}^2 \to \mathbb{R}$, with:

$$\mu_0(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.5}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(f(\mathbf{x}), f(\mathbf{x}')) \tag{2.6}$$

This is commonly written as:

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu_0(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.7}$$

GPs are an example of the broader category of continuous stochastic processes, which may be intuitively thought of as random functions. For this reason GPs can be used as priors and posteriors over functions. This makes them a natural tool for sequential, Bayesian learning.

As we will be using GPs to model the objective function f with domain $\mathcal{X} \subseteq \mathbb{R}^d$ we also assume that the index set is a compact subset of $\mathbb{R}^d$, although this is not a necessary assumption in the wider context.

2.3 Kernel functions

The covariance function k is also regularly called the *kernel* function. It is a positive-definite measure of the expected similarity of two inputs in the output space.

The structure of a GP is primarily determined by the kernel as it specifies the likely properties of functions drawn from the process [24]. It is frequently assumed that the prior mean function is a constant, both for convenience and because uncertainty about the mean function can be encapsulated by the kernel [25]. In practise, this constant may be inferred from the data [6] as a part of model selection. However for simplicity, in this project we will take it to be zero.

For these reasons, the choice of an appropriate kernel for the problem at hand is crucial. Similarly to the choice of an uninformative prior distribution for Bayesian inference about a parameter for which little is known, for Bayesian optimization it is reasonable to choose a kernel function which is flexible and does not impose strong assumptions. Two of the most widely used kernels, in part because of their suitability to this purpose, are the squared exponential kernel and members of the Matérn family of kernels.

2.3.1 Squared exponential

The *squared exponential* (SE) kernel produces sample functions that are infinitely differentiable, and so can be used to model functions which are very smooth.

$$k_{\mathrm{SE}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(\frac{-|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right) \tag{2.8}$$

Variables which are close in the input space are highly correlated, the correlation decreasing as variables become further apart. The kernel attains a maximum value of σ_f^2 at $(\mathbf{x}, \mathbf{x})$.

The amplitude σ_f^2 and length-scale l are examples of kernel *hyper-parameters*, which in general we will denote by the vector $\boldsymbol{\theta}$. The amplitude controls the overall scale of variation. The length-scale controls how quickly the exponential decays and thus the rate at which input variables become de-correlated.

In general, kernel hyper-parameters substantially alter the behaviour of sample functions and generalisation properties of GP models. For this reason they should be tuned as a part of model selection, which will be detailed in section 2.7.

Plots (a), (c) and (e) in figure 2.1 show the effect of changing the length-scale l on sample functions drawn from a GP with SE kernel: the smaller the value of l the more wiggly the samples. In fact, this may be measured by the mean number of level-zero upcrossings on the unit-interval, which for a GP with SE kernel and mean-zero in one-dimension is $(2\pi l)^{-1}$ [23]. The effect of changing the amplitude σ_f^2 has not been shown in figure 2.1, as it simply corresponds to rescaling the y-axis.

2.3.2 Matérn

For realistic optimization problems the distribution of functions generated by squared exponential kernels may be unrealistically smooth [3] [26]. An alternate suggestion is to use a member of the more general *Matérn* family [27], which allows the smoothness to be controlled.

$$k_{\mathrm{M}}(\mathbf{x}, \mathbf{x}') = \frac{\sigma_f^2}{2^{\nu-1}\Gamma(\varsigma)} \left(\frac{\sqrt{2\varsigma}|\mathbf{x} - \mathbf{x}'|}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}|\mathbf{x} - \mathbf{x}'|}{l} \right) \tag{2.9}$$

Here σ_f^2 and l are amplitude and length-scale as before, $\nu > 0$ is an additional hyper-parameter which controls how rough the sample functions are, K_ν is a modified Bessel function of the second kind and $\Gamma(\cdot)$ is the Gamma function.

Sample functions produced by the Matérn kernel are $\lfloor \nu \rfloor$ times differentiable [28]. In the limit as $\nu \to \infty$ the squared exponential kernel is recovered.

Plots (b), (d) and (f) in figure 2.1 show sample functions drawn from a GP with Matérn kernel, with $\nu = 1/2, 3/2$ and $5/2$. These particular values of ν are chosen as when $\nu = p + 1/2$ for some $p \in \mathbb{N}$ then the Matérn kernel may be simply expressed as the product of an exponential and a polynomial of order p [23]. Although it may be argued that the roughness should be learned from the data [26] it is common to chose ν to be either $3/2$ or $5/2$ outright [11], for which:

$$k_{\mathrm{M3/2}}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{3}|\mathbf{x} - \mathbf{x}'|}{l} \right) \exp\left(-\frac{\sqrt{3}|\mathbf{x} - \mathbf{x}'|}{l} \right) \tag{2.10}$$

$$k_{\mathrm{M5/2}}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{5}|\mathbf{x} - \mathbf{x}'|}{l} + \frac{5|\mathbf{x} - \mathbf{x}'|^2}{3l^2} \right) \exp\left(-\frac{\sqrt{5}|\mathbf{x} - \mathbf{x}'|}{l} \right) \tag{2.11}$$

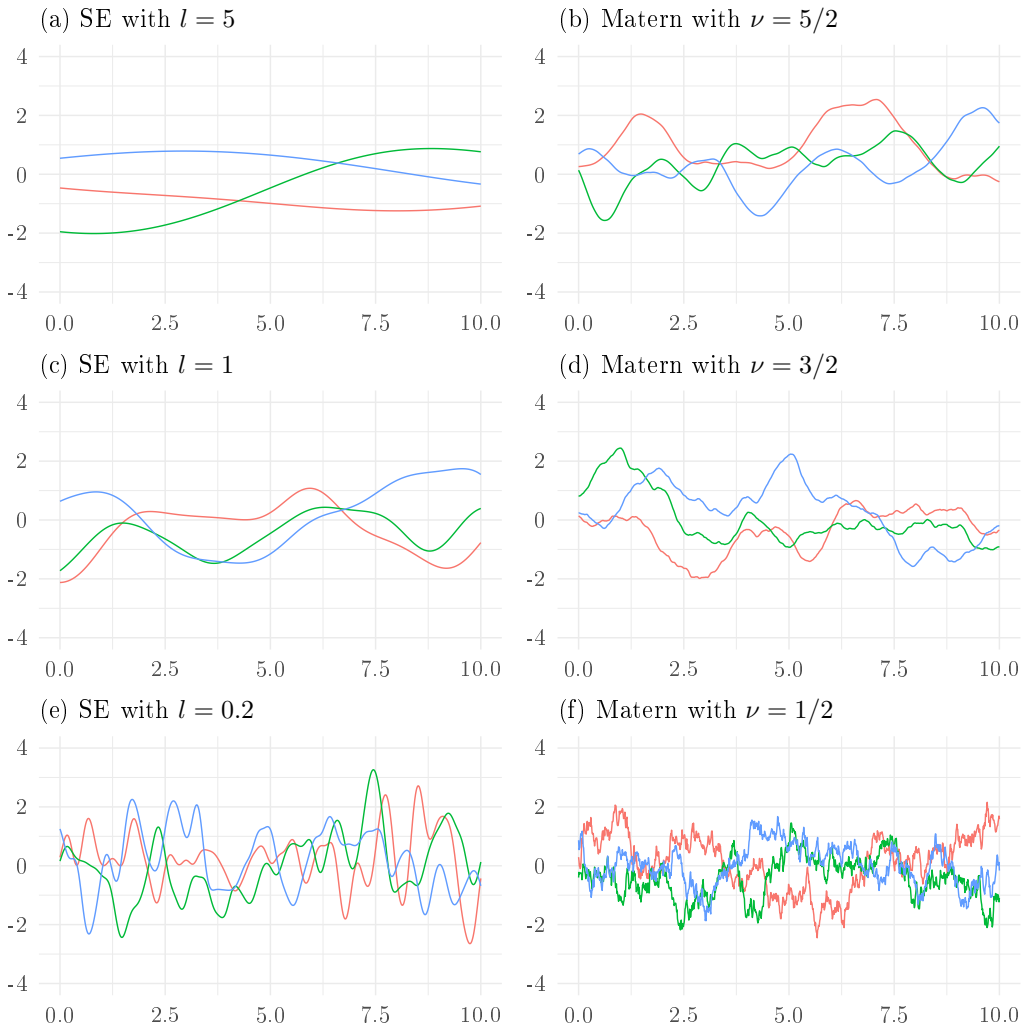


FIGURE 2.1: Six plots, each with three samples from a zero-mean GP. Amplitude is $\sigma_f^2 = 1$ for each plot. In the plots (b), (d) and (f) with Matérn kernel, the length-scale $l = 1$. Variation in the length-scale of the Matérn kernel would have the same effect as in the squared exponential kernel, which is shown in plots (a), (c) and (e).

2.3.3 Automatic relevance determination

Both of the above squared exponential and Matérn kernels are called *stationary* as they are functions of $x - x'$. This means that they are invariant to transformations in the input space and corresponds to the modelling assumption that the objective function varies uniformly across the domain.

Further, since they are also both functions of $|\mathbf{x} - \mathbf{x}'|$ they are called *isotropic* - uniform in all directions. However, this is often not the case and it would be advantageous to allow the kernel to express different rates of variation in different dimensions. More general *anisotropic* kernels may be defined by replacing $|\mathbf{x} - \mathbf{x}'|^2$ by

$$r^2(\mathbf{x}, \mathbf{x}') = (\mathbf{x} - \mathbf{x}')^T D(\mathbf{x} - \mathbf{x}') \tag{2.12}$$

for some positive semi-definite matrix D. Of course if D is a dense (non-sparse) matrix it introduces many hyper-parameters, which may be difficult to deal with. As such, a popular choice for D is a diagonal matrix with entries $D_{j,j} = 1/l_j^2$ such that:

$$r^2(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{d} \frac{(x_j - x'_j)^2}{l_j^2} \tag{2.13}$$

The l_j are simply individual length-scale hyper-parameters for each dimension. Kernels which are functions of equation 2.13 are called *automatic relevance determination* kernels [29] as they allow for the relative importance of each dimension to be learned when the kernel hyper-parameters are tuned. This is helpful in dealing with cases where the objective function has low effective dimensionality, as was briefly mentioned in section 1.3.

2.4 Sampling from a Gaussian process

Consider that we are interested in sampling from a zero-mean GP at a finite collection of points $\mathbf{x}_{1:n} := \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where each $\mathbf{x}$ is in $\mathcal{X}$. Since each $\mathbf{x}$ is itself a vector, any finite vector of indices induces a matrix which we denote by $X := [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T \in \mathcal{X}^n \subseteq \mathbb{R}^{n \times d}$. The vector of random variables $\mathbf{f} := f(X) := [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T$ corresponding to this matrix by definition has a joint multivariate Gaussian distribution.

The parameters of this distribution are determined by applying the mean and kernel functions: $\mathbf{f}$ has mean vector $\mathbf{0}$ and covariance matrix $K := k(X, X)$ with elements $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j), \forall i = 1, \ldots, n, j = 1, \ldots, n$. The requirement that the kernel k is

positive-definite precisely means that any such matrix K, sometimes called the Gram matrix, is a valid covariance matrix. So, the distribution of $\mathbf{f}$ is:

$$\begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \dots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \right) \tag{2.14}$$

Put more succinctly:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, K) \tag{2.15}$$

This distribution is referred to as the marginal likelihood, as it implicitly marginalizes out all possible function values at every $\mathbf{x} \in \mathcal{X}$ which is not in $\mathbf{x}_{1:n}$ [25]. This is justified by property G1 of the multivariate Gaussian. For this reason, although a GP is an infinite object it is only ever necessary to consider finite subsets. Discretizing the domain and using equation 2.15 allows us to visualize samples from a GP, as in figure 2.1.

Although this all does not seem particularly useful as it stands, shortly we will show that this distribution may be used as a prior in order to perform inference directly in function-space. It is for this reason that the notation μ_0 is chosen for the prior mean function. First, however, we will take a brief interlude to discuss Bayesian linear regression and how it relates to GPs.

2.5 Bayesian linear regression

Given a training data-set of observations and a new vector $\mathbf{x}_* \in \mathbb{R}^d$ of input variables, the aim of *regression* is to predict target variable $f(\mathbf{x}_*)$. This is described as a *supervised learning* task as the training data-set, i.e. the values of the target variable at the collection of inputs $\mathbf{x}_{1:n}$, plays the role of teaching us about the behaviour of f [21].

The approach of linear regression is to define a model in terms of a linear combination of fixed basis functions:

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) \tag{2.16}$$

where $\mathbf{w}^T$ is a vector of weights and $\phi(\mathbf{x})$ is a vector of M fixed non-linear basis functions that depend on the input vector $\mathbf{x}$. Many choices of basis functions are possible such as polynomial, Gaussian, sigmoidal or Fourier [30].

Typically the values of $\mathbf{w}$ are estimated by minimizing the sum of squares error from the fitted values to the training data, which corresponds to finding the maximum likelihood estimator. This has analytic solution $\hat{\mathbf{w}}$ given by the normal equations [30]

$$\hat{\mathbf{w}} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{f} \tag{2.17}$$

where $\mathbf{\Phi}$ is the design matrix with elements $\Phi_{n,k} = \phi_k(x_n)$ such that:

$$\mathbf{\Phi} = \begin{pmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_M(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_M(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_n) & \phi_2(\mathbf{x}_n) & \cdots & \phi_M(\mathbf{x}_n) \end{pmatrix} \tag{2.18}$$

A more Bayesian approach to this problem would be to place a prior distribution, say a Gaussian, on the weights $\mathbf{w}$ such that:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{\Sigma}_w) \tag{2.19}$$

where $\mathbf{\Sigma}_w$ is the M by M covariance matrix. Notice that fixing $\mathbf{w}$ fixes $f(\mathbf{x})$, thus the probability distribution over $\mathbf{w}$ induces a probability distribution over functions $f(\mathbf{x})$. This distribution over functions is exactly a GP prior. As $f(\mathbf{x})$ is a linear combination of Gaussian distributions it is itself Gaussian. The mean function μ_0 is zero, as

$$\begin{aligned} \mu_0(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ &= \phi(\mathbf{x})\mathbb{E}[\mathbf{w}] \\ &= 0 \end{aligned} \tag{2.20}$$

and the covariance function is given by

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \mathrm{Cov}(f(\mathbf{x}), f(\mathbf{x}')) \\ &= \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] \\ &= \phi(\mathbf{x})^T\mathbb{E}[\mathbf{w}^T\mathbf{w}]\phi(\mathbf{x}') \\ &= \phi(\mathbf{x})^T\Sigma_w\phi(\mathbf{x}') \end{aligned} \tag{2.21}$$

This GP is called *degenerate* precisely because it can be represented with a finite collection of basis functions as we have shown above. In fact, by Mercer's theorem,

for every valid kernel function there exists a (possibly infinite) expansion in terms of basis functions [23]. For example, taking $l^2 = \sigma_f^2 = 1$, the squared exponential kernel in one-dimension is given by [31]

$$\begin{aligned} k(x, x') = e^{-\frac{(x-x')^2}{2}} &= e^{-\frac{x^2}{2}} e^{-\frac{x'^2}{2}} \sum_{k=0}^{\infty} \frac{(xx')^k}{k!} \\ &= \phi(x)^T \phi(x') \end{aligned} \tag{2.22}$$

where we have the infinite basis expansion

$$\phi(x) = e^{-\frac{(x-x')^2}{2}} \left[1, x, \frac{x^2}{\sqrt{2}}, \frac{x^3}{\sqrt{6}}, ...\right] \tag{2.23}$$

2.6 Gaussian process regression

In this section we will show how regression can be done directly with GPs, *without* having to think about the basis functions.

Suppose that at inputs $\mathbf{x}_{1:n}$ the values of the GP $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T$ are observed. We are interested in the value of the GP $f_* := f(\mathbf{x}_*)$ at a new input $\mathbf{x}_*$. By the GP definition, the joint prior distribution of $\mathbf{f}$ and f_* is

$$\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} K & k(X, \mathbf{x}_*)^T \\ k(X, \mathbf{x}_*) & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right) \tag{2.24}$$

where $k(X, \mathbf{x}_*) = [k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_n, \mathbf{x}_*)]^T$ is the vector of covariance terms between the locations $\mathbf{x}_{1:n}$ and $\mathbf{x}_*$. For brevity this will in future be denoted by $\mathbf{k}_*$. Now we make use of property G2 of the multivariate Gaussian to find that the conditional distribution of f_* given $\mathbf{f}$ is Gaussian

$$f_* | \mathbf{f} \sim \mathcal{N}(\mu_n(\mathbf{x}_*), \sigma_n^2(\mathbf{x}_*)) \tag{2.25}$$

with mean and variance given by

$$\mu_n(\mathbf{x}_*) = \mathbf{k}_*^T K^{-1} \mathbf{f} \tag{2.26}$$

$$\sigma_n^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T K^{-1} \mathbf{k}_* \tag{2.27}$$

This is known as the *predictive distribution* as it gives the mean and variance at any given unobserved test point $\mathbf{x}_*$.

Often, it is not possible to directly observe $\mathbf{f}$ itself. Instead, we may observe a noise contaminated version $\mathbf{y}$, where each $y = f(\mathbf{x}) + \epsilon$ as in property P4. We assume Gaussian additive independent identically distributed noise with zero mean and variance σ^2 such that:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 I) \tag{2.28}$$

Integrating out the unobserved values $\mathbf{f}$ with distribution $p(\mathbf{f})$ from equation 2.15, the marginal likelihood is:

$$p(\mathbf{y}) = \int d\mathbf{f}\, p(\mathbf{y}|\mathbf{f})p(\mathbf{f}) \tag{2.29}$$

$$= \mathcal{N}(\mathbf{y}|\mathbf{0}, K + \sigma^2 I) \tag{2.30}$$

Introducing the additional noise term to 2.24, the joint distribution of the values $\mathbf{y}$ and f_* is:

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} K + \sigma^2 I & \mathbf{k}_*^T \\ \mathbf{k}_* & k(\mathbf{x}_*, \mathbf{x}_*) \end{bmatrix} \right) \tag{2.31}$$

This can be conditioned as before to show that the predictive distribution $p(\mathbf{f}_*|\mathbf{y})$ is Gaussian with mean and variance given by:

$$\mu_n(\mathbf{x}_*) = \mathbf{k}_*^T (K + \sigma^2 I)^{-1} \mathbf{y} \tag{2.32}$$

$$\sigma_n^2(\mathbf{x}_*) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \sigma^2 I)^{-1} \mathbf{k}_* \tag{2.33}$$

Taking $\sigma^2 = 0$ recovers equations 2.26 and 2.27. The predictive distribution actually defines a posterior Gaussian process in its own right [24] with mean function as in equation 2.32 and kernel function

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}^T (K + \sigma^2 I)^{-1} \mathbf{k}' \tag{2.34}$$

where $\mathbf{k} = [k(\mathbf{x}_1, \mathbf{x}), \ldots, k(\mathbf{x}_n, \mathbf{x})]^T$ and $\mathbf{k}'$ is defined similarly.

Observe that rather than depending on a fixed number of parameters, the complexity of the model adapts to the number of training data points [32]. As such, GP regression is an example of a Bayesian *non-parametric* model, where the parameter-space is infinite. This is advantageous as we do not have to specify a parametric structure and have less worries about over-fitting, which in parametric modelling typically must be combated with regularization.

In equation 2.26 notice that that predictive mean is simply a linear combination of the of the observations. For this reason equation 2.26 is known as a linear predictor [23] - in fact it is actually the best linear unbiased predictor [11].

In equation 2.33 the predictive variance is the prior variance minus a (positive) term which only depends on the training data locations. The closer the training data locations $\mathbf{x}_{1:n}$ are to $\mathbf{x}_*$ the greater the values in the vector $\mathbf{k}_*$ are, and the greater the Mahalanobis distance $\mathbf{k}_*^T(K+\sigma^2 I)^{-1}\mathbf{k}_*$. This explains the visual effect, which can be seen in figure 2.1, of the predictive variance decreasing closer to the training data points. That being said, it may be considered disappointing that the predictive variance is completely independent of the actual values observed $\mathbf{y}$: surely if an unexpected value is observed then this should be accounted for by increased uncertainty. Shah, Wilson, and Ghahramani [33] for this and other reasons propose the use of the more general Student-t process as an alternative to the GP.

The dimension of the matrix K is the number of data points observed n. This matrix is inverted in equations 2.26 and 2.27 usually by computing the Cholesky decomposition. This has computational cost big $\mathcal{O}(n^3)$, which is typically the bottleneck of GP regression - limiting it's reach to relatively small n. This has motivated the development of sparse Gaussian processes [34] [35] which approximate the posterior over functions to make inference computationally cheaper.

2.7 Learning the hyper-parameters

In the complete hierarchical Bayesian model selection framework [36] models are specified in three tiers: model parameters, model hyper-parameters and model structures. We will focus our attention on model selection at the second level, the model hyper-parameters.

At the bottom level the model parameters are obtained by the posterior distribution over functions, as our model is non-parametric. The top level corresponds to the choice of kernel function, which is one of the key challenges of GP regression. To alleviate potential problems, the kernel functions chosen are purposefully flexible as we have discussed.

This leaves the model hyper-parameters, which are the hyper-parameters of the kernel function $\boldsymbol{\theta}$ together with the noise-variance σ^2 if it is unknown. The most simplistic approach would be to specify the hyper-parameters a-priori, however ill-fitting choices, illustrated in figure 2.3, can lead to very poor performance. As a result it is

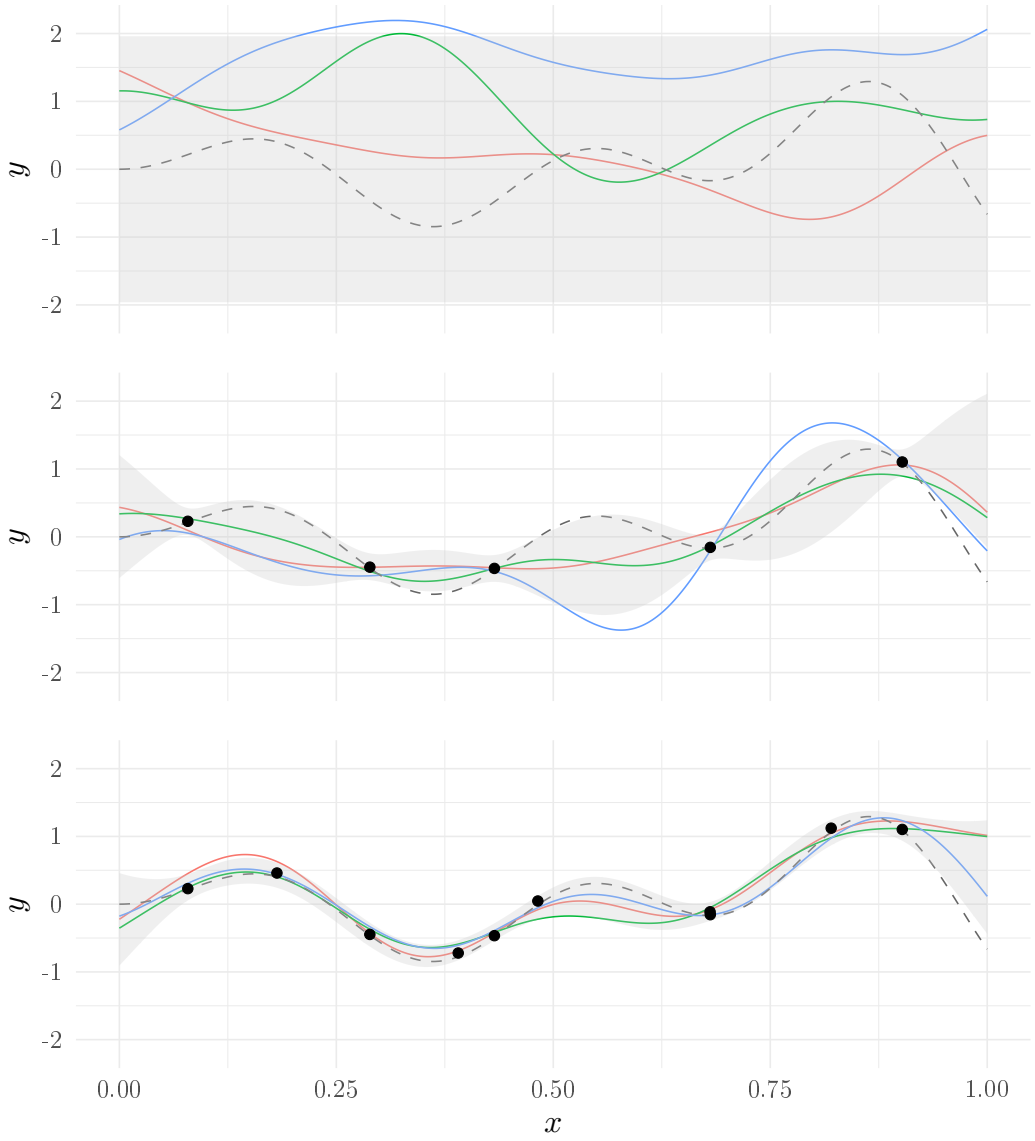


FIGURE 2.2: GP regression example. In each plot there are three samples curves from a GP with with prior mean zero and squared exponential kernel with hyper-parameters $\boldsymbol{\theta} = (\sigma_f, l) = (1, 0.15)$. Data is generated by objective function $f(x) = \sin(5x)\sin(13x)\exp(x - 0.5)$, shown as the grey dotted line, with additive Gaussian noise with variance $\sigma^2 = 0.01$. The shaded region is a 95% confidence interval calculated using the predictive mean 2.32 and variance 2.33 equations.

better to learn the hyper-parameters from the data, which may be done by maximizing the marginal likelihood, which is analytic for GP models and given by equation 2.29. Making the dependence of K on the kernel hyper-parameters $\boldsymbol{\theta}$ explicit, its logarithm is:

$$\log p(\mathbf{y}|X,\boldsymbol{\theta}) = -\frac{n}{2}\log 2\pi \underbrace{-\frac{1}{2}\log|K_{\boldsymbol{\theta}}+\sigma^2 I|}_{\text{controls model capacity}} \underbrace{-\frac{1}{2}\mathbf{y}^T(K_{\boldsymbol{\theta}}+\sigma^2 I)^{-1}\mathbf{y}}_{\text{encourages fit with data}} \tag{2.35}$$

Optimizing this likelihood gives an empirical Bayes estimate for the hyper-parameters. Equation 2.35 balances the trade-off between fit and complexity of the model [25]. The second term penalizes the complexity of the model as the determinant of the covariance matrix grows with more complicated kernels, such as those with smaller length-scales. The final term is a Mahalanobis distance which controls the quality of the fit.

There are many possible ways to optimize equation 2.35. One alternative is to use quasi-Newton methods such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm with random restarts. In this project we will use the L-BFGS-B algorithm [37] which approximates the BFGS algorithm and is adapted to handle simple box-constraints on the hyper-parameters, i.e. bounded above and below. This algorithm is used in many R implementations of GP regression such as the `DiceKriging` [38], `GPfit` [39] and `laGP` [40] packages.

In Bayesian optimization, the hyper-parameters are typically tuned at every iteration of the algorithm - in particular whenever a GP is fit.

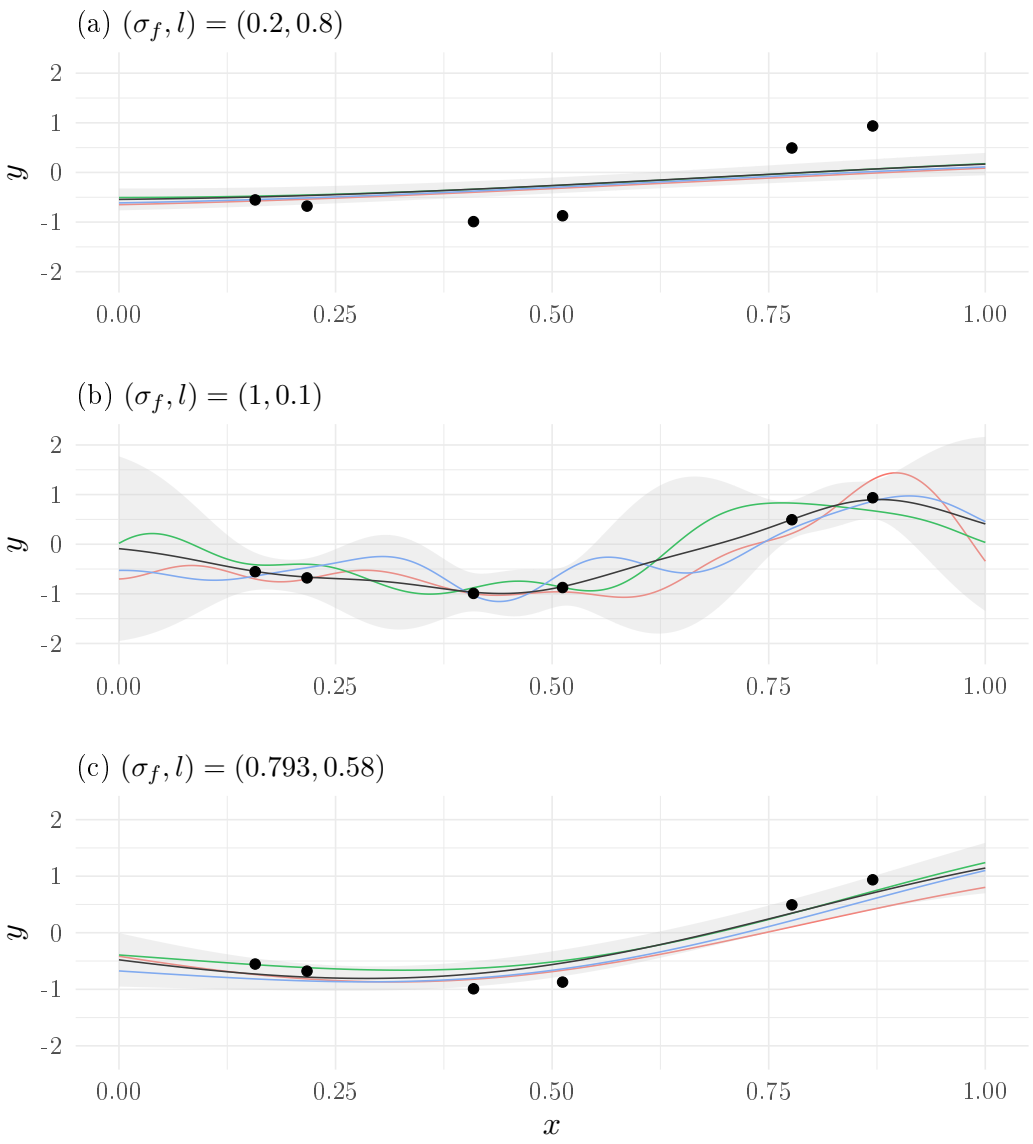


FIGURE 2.3: The impact of different amplitude and length-scale hyper-parameters on squared exponential kernel GP regression models. As in figure 2.2 there are three sample curves with the addition of a black mean function. Figure (a) is an example of under-fitting the data. It is overly simplistic and fails to capture the underlying trend in the data. Figure (b) is an example of over-fitting. It is a more complex model which has mean function that interpolates the data. Finally, figure (c) uses the maximum likelihood estimates of the hyper-parameters and is a balance between figures (a) and (b).

Chapter 3

Acquisition functions

In sequential optimization, the next point in the search space to be evaluated at each stage is determined by an optimization *policy* $\boldsymbol{\pi}$ which is a sequence of decision rules π_n which map from the space of possible training data at stage n to a point $\mathbf{x}_n \in \mathcal{X}$. In Bayesian optimization a GP emulator to the objective function fit on all the data available at stage n is used to help inform this decision.

How should the policy $\boldsymbol{\pi}$ be chosen?

3.1 Optimal policies

A natural place to start is to look for a policy which is in some sense optimal. In order to find such a policy it is necessary to define a metric so that performance can be measured. The loss function regret [41] is a sensible choice [42] for this metric. We define the *instantaneous regret* at stage n as the difference between the (potentially unknown) immediate reward of the optimal action $f(\mathbf{x}^\star)$ and the actual immediate reward received $f(\mathbf{x}_n)$, so that

$$r_n = f(x_n) - f(\mathbf{x}^\star) \tag{3.1}$$

The *cumulative regret* $R_N = \sum_{n=1}^{N} r_n$ is the sum of instantaneous regrets at each stage. *Simple regret* is the instantaneous regret of the final recommendation x^- of the algorithm.

Different varieties of regret are appropriate to different situations. Cumulative regret is a sensible way to measure performance in online problems such as the multi-armed bandit problem, which we introduced in section 1.6. In off-line problems however

there is no need to penalize poor performance during the optimization process as this would discourage exploration. Simple regret instead should be considered.

As the optimization is stochastic, the optimal policy minimizes the *expectation* of the simple regret. Unfortunately, such a policy is the solution to an dynamic programming problem which requires solving many nested maximizations and expectations [43] and is therefore intractable. Although there has been some recent work [44] [43] in this area, the majority of the focus has instead been on more tractable policies defined by so-called acquisition functions.

3.2 Acquisition functions

Acquisition functions provide a heuristic, typically myopic, measure of the utility of prospective query points. At stage n to determine the next point to query $\mathbf{x}_n$, the chosen acquisition function α is maximized over the design space.

$$\mathbf{x}_n = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x}) \tag{3.2}$$

In other words, the optimization policy $\boldsymbol{\pi}$ is, in a way, homogenized - each decision rule π_n is replaced with the consultation of a particular acquisition function. That being said, although it has been omitted in equation 3.2 the acquisition function may depend on the stage n, in some way.

Many possible acquisition function have been suggested in the literature. The chosen acquisition function should effectively balance the trade-off between local and global search. To encourage local search it should be high where the posterior predictive mean is low and to encourage global search it should be high where the posterior predictive marginal variance is high.

A common approach is to look for points which offer improvement over some incumbent value $\nu \in \mathcal{X}$, which represents the best solution found so far[1]. This value ν is typically the minimum function evaluation observed so far

$$\nu = \min\{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{n-1})\} \tag{3.3}$$

[1]Notation has been overloaded: unrelated to the smoothness parameter of the Matérn kernel

when observations are deterministic, or the minimum expected value of the emulator

$$\nu = \arg\min_{\mathbf{x}\in\mathcal{X}} \mathbb{E}[f(\mathbf{x})] \tag{3.4}$$

when the objective function is noisy [6]. We will discuss two improvement-based acquisition functions, probability of improvement and expected improvement, in the following sections 3.2.1 and 3.2.2.

3.2.1 Probability of improvement

Early work by Kushner [13] proposes maximizing the probability that $f(\mathbf{x})$ improves on the incumbent, ν. This may be calculated analytically by noting that at stage n the random variable $f(\mathbf{x})$ has Gaussian distribution with mean and variance given by predictive equations 2.26 and 2.27. The GP emulator has has been conditioned on the data-set $\mathcal{D}_{n-1}$, but for ease-of-exposition we will denote this mean and variance simply by $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ respectively. As such, the *probability of improvement* (PI) acquisition function is

$$\alpha_{\mathrm{PI}}(\mathbf{x}) \equiv \mathbb{P}(f(\mathbf{x}) \leq \nu) = \Phi\left(\frac{\nu - \mu(\mathbf{x})}{\sigma(\mathbf{x})}\right) = \Phi(z(\mathbf{x})) \tag{3.5}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function and $z(\mathbf{x}) = (\nu - \mu(\mathbf{x}))/\sigma(\mathbf{x})$ is the standardization score.

The drawback of this approach is that global search is often neglected because likely improvements, however infinitesimal, are favoured [8]. The search will move on, but only after almost exhaustively fine-tuning the current best solution. To remedy this, Kushner proposes that the minimum improvement accepted is a trade-off parameter $\xi(n)$. His idea being that $\xi(n)$ operates on a cooling schedule, starting high to encourage global search and decreasing towards zero later so that solutions are refined locally. Jones [45] describes that although performance may be impressive, the method is extremely sensitive to the choice of this parameter.

3.2.2 Expected improvement

Expected improvement (EI) [14] is another natural, improvement-based, approach which overcomes some of the drawbacks of PI. As well considering the probability

that $f(\mathbf{x})$ improves on ν, EI also takes into account the degree of that potential improvement. In particular, under the model, the improvement function $\mathrm{I}(\cdot)$ is defined to be the random variable

$$\mathrm{I}(\mathbf{x}) \equiv \max\{0, \nu - f(\mathbf{x})\} \tag{3.6}$$

The likelihood of attaining improvement $I > 0$ is simply Gaussian, with density

$$\frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left\{-\frac{1}{2}\frac{(\nu - I - \mu(\mathbf{x}))^2}{\sigma^2(\mathbf{x})}\right\} \tag{3.7}$$

Taking the expected value of the improvement over this density gives the expected improvement acquisition function

$$\alpha_{\mathrm{EI}}(\mathbf{x}) \equiv \mathbb{E}[\mathrm{I}(\mathbf{x})] = \int_0^\infty I \, \frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left\{-\frac{1}{2}\frac{(\nu - I - \mu(\mathbf{x}))^2}{\sigma^2(\mathbf{x})}\right\} dI \tag{3.8}$$

Using integration by parts [46] this expression has the closed form

$$\alpha_{\mathrm{EI}}(\mathbf{x}) = \sigma(\mathbf{x})[z(\mathbf{x})\Phi(z(\mathbf{x})) + \phi(z(\mathbf{x}))] \tag{3.9}$$

where $\Phi(\cdot)$ is as before and $\phi(\cdot)$ is the standard normal probability density function.

EI has attractive theoretical convergence properties [47] and has been shown to work well in practice [3], for example as a part of the "Efficient Global Optimization" algorithm [48]. Further, it is easy to implement and does not require setting any additional parameters. For these reasons it is often the default choice.

3.2.3 Lower confidence bound

Originally studied in the the multi-armed bandit setting by Lai and Robbins [49], the *lower confidence bound* (LCB) acquisition function [42] is an optimistic acquisition function which assumes a fixed probability best-case scenario occurs, and looks for the point which is best under this scenario. It is defined to be:

$$\alpha_{\mathrm{LCB}}(\mathbf{x}) \equiv \lambda\sigma(\mathbf{x}) - \mu(\mathbf{x}) \tag{3.10}$$

The constant λ is used to to explicitly balance between local search and global search by setting this fixed probability. Srinivas, Krause, Kakade, *et al.* [42] are able to show that with appropriate choice of λ as a function of n the LCB acquisition

function is what is called no-regret, which means that $\lim_{N\to\infty} R_N/N = 0$, with high probability.

3.3 Maximizing the acquisition function

By using an acquisition function, the primary optimization problem 1.1 has been traded for a series of optimization problems 3.2: one for each stage. For this trade to be viable it must be easy to optimize the acquisition function. Thankfully unlike the objective function the acquisition functions all have analytic expressions which can be sampled cheaply.

Various approaches have been proposed and successfully implemented in the literature. Most simplistically, the domain can be discretized and exhaustively grid-searched - which is of course how visualizations like figure 1.2 are produced. In this project we will use the same algorithm as was used to optimize the GP hyperparameters - namely L-BFGS-B with random restarts.

Chapter 4

Constrained optimization

4.1 Introduction

Previously we considered the optimization problem in equation 1.1 where all values $\mathbf{x}$ in the search space $\mathcal{X}$ are valid. In constrained optimization problems this is not the case: there are a set of additional requirements, called *constraints*, that $\mathbf{x}$ must meet for it to be considered feasible. Using a set of K inequality constraints $c_k : \mathcal{X} \to \mathbb{R}$ this may be formulated as the constrained optimization problem:

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \quad \text{s.t.} \quad c_k(\mathbf{x}) \geq 0, \forall k \in \{1, ..., K\} \tag{4.1}$$

The set of points $\mathbf{x}$ which satisfy each of the constraints c_k is called the *feasible region* $\mathcal{R} \subseteq \mathcal{X}$. Points outside the feasible region are called *infeasible*.

Constraints may be of two types, *coupled* or *decoupled*. Coupled constraints may only be evaluated jointly alongside the objective function. That is, when the objective function has been queried the user also learns the value of the constraint function. Decoupled constraints, on the other hand, can be evaluated independently of the objective. This is more of a challenge since if each constraint is decoupled then at every stage there is a choice of $K + 1$ functions (the K constraints plus the the objective function f) which may be queried.

We consider that, in general like the objective function, the constraints are: not necessarily linear or convex P1; black-box P2; and expensive to evaluate individually P3 in the decoupled case. In addition, the constraints may be noisy P4, which will be discussed in section 4.1.1. Without these properties, it may be possible to cheaply

find the feasible region $\mathcal{R}$ and to proceed with unconstrained optimization on this reduced search space.

The addition of constraints considerably widens the scope of problems which may be considered compared with the unconstrained case. Returning to a previous example, when optimizing the hyper-parameters of a machine-learning model there may be settings which cause the model to diverge, so the optimization may need to be constrained. Other examples include optimizing CPU speed of a processor whilst restricting power usage below some threshold or optimizing the yield of a chemical process whilst limiting the amount of unwanted by-products which are produced.

Some authors [50] [51] also present constrained optimization as a proxy method for dealing with trade-offs. In their paper about optimizing systems at Facebook, Letham, Karrer, Ottoni, *et al.* [51] state that:

> " ...there are almost always trade-offs involved in optimizing real systems: improving the quality of images may result in increased data usage; increasing cache sizes may improve the speed of a mobile application, but decrease reliability on some devices; optimizing click-through rates may result in decreases in quality."

In general, these problems may perhaps more generally be described as multi-objective optimization problems in which the output $\mathbf{y}$ has dimension greater than one. Of course there is no strict ordering for points in $\mathbb{R}^2$ and above, and so the aim of multi-objective optimization is to approximate the Pareto set of non-dominated points [52]. For example, a more complete treatment of a chemical yield optimization with constrained unwanted by-products may find the highest yield possible for any given acceptable amount of by-product. The simplification is possible because constrained optimization is a special case of multi-objective optimization, in which the constraint functions are treated as objective functions which are constant over the feasible region and negative infinity elsewhere [20].

4.1.1 Noisy constraints

Problems where evaluations of the constraints are contaminated by noise are particularly challenging. If it is assumed that the noise is Gaussian then no point, even if directly observed many times, can be guaranteed to be feasible. As such, the constrained optimization problem formulated in equation 4.1 is ill-posed: it is impossible to know with certainty that any of the constraints have been met.

It is reasonable instead to require that each constraint c_k is met with high-probability [20]. We define the kth Boolean *probabilistic constraint* $\mathcal{C}_k$ such that

$$\mathcal{C}_k(\mathbf{x}) \quad \iff \quad \mathbb{P}(c_k(\mathbf{x}) \geq 0) \geq 1 - \delta_k \tag{4.2}$$

for some confidence level δ_k. The condition that each of the K probabilistic constraints $\mathcal{C}_k$ are met is denoted $\mathcal{C}$ such that:

$$\mathcal{C}(\mathbf{x}) \quad \iff \quad \forall k, \mathcal{C}_k(\mathbf{x}) \tag{4.3}$$

Now, the noisy constrained optimization problem is:

$$\mathbf{x}^\star = \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}) \quad \text{s.t.} \quad \mathcal{C}(\mathbf{x}) \tag{4.4}$$

Notice that equation 4.1 may be recovered by requiring certainty of satisfying the constraints, that is by setting $\delta_k = 0$ for all k.

4.2 Constrained Bayesian optimization

Bayesian optimization is a promising strategy for constrained optimization problems. For example, the dissertation of Griffiths [53] uses constrained Bayesian optimization to optimize the molecular properties of drug candidates, using constraints to avoid invalid molecular structures.

The plan is to model each of the constraints c_k by GP emulators - which are sequentially updated as data is observed, just as we have done with the objective function. For tractability, the emulators are assumed to be conditionally independent given the input $\mathbf{x}$. Now the only alteration that needs to be made is to in some way incorporate the constraints into the acquisition function. We will discuss two ways of doing this.

4.2.1 Expected improvement with constraints

Expected improvement, introduced in section 3.2.2, may be extended to consider the equation 4.1 where the constraints are coupled. This was first proposed by Schonlau, Welch, and Jones [54] and has been revisited more recently [55] [50] - in particular by Gelbart [20] who extends it to the equation 4.4.

The main idea is to assign zero improvement to all infeasible points and to adjust the incumbent value ν to either be the best point observed to be feasible so far or the minimum of the posterior mean such that $\mathcal{C}(\nu)$ is true.

Proceeding similarly to the unconstrained case, the constrained improvement function is the random variable

$$\mathrm{I}_C(\mathbf{x}) \equiv \max\{0, \nu - f(\mathbf{x})\}\,\Delta(\mathbf{x}) \tag{4.5}$$

where $\Delta(\mathbf{x}) \in \{0, 1\}$ is the constraint indicator function - taking value 1 if each of the constraints are met and 0 otherwise. Taking the expectation as before gives the *constrained expected improvement* (EIC) acquisition function, which we are able to factor due to the assumed conditional independence given $\mathbf{x}$ of the emulators:

$$\begin{aligned}\alpha_{\mathrm{EIC}}(\mathbf{x}) &\equiv \mathbb{E}[\max\{0, \nu - f(\mathbf{x})\}\,\Delta(\mathbf{x}) \,|\, \mathbf{x}] \\ &= \mathbb{E}[\max\{0, \nu - f(\mathbf{x})\} \,|\, \mathbf{x}]\,\mathbb{E}[\Delta(\mathbf{x}) \,|\, \mathbf{x}] \\ &= \alpha_{\mathrm{EI}}(\mathbf{x}) \prod_{k=1}^{K} \mathbb{P}(c_k(\mathbf{x}) \geq 0)\end{aligned} \tag{4.6}$$

Notice that this is easily interpretable as a product of the unconstrained expected improvement function and the probability of satisfying each of the constraints. Applying the closed form from equation 3.9 and noting that $\mathbb{P}(c_k(\mathbf{x}) \geq 0)$ is a Gaussian CDF we have

$$\alpha_{\mathrm{EIC}}(\mathbf{x}) = \sigma(\mathbf{x})[z(\mathbf{x})\Phi(z(\mathbf{x})) + \phi(z(\mathbf{x}))] \prod_{k=1}^{K} \Phi\left(\frac{\mu_k(\mathbf{x})}{\sigma_k(\mathbf{x})}\right) \tag{4.7}$$

where $z(\mathbf{x}) = (\nu - \mu(\mathbf{x}))/\sigma(\mathbf{x})$ as before and the predictive mean and standard deviation of constraint k are given by $\mu_k(\mathbf{x})$ and $\sigma_k(\mathbf{x})$. Due to this analytic form, EIC is easy to implement.

Gelbart [20] points out that this formulation is inconsistent when no feasible point has been observed (that is there does not exist a point $\mathbf{x}$ such that $\mathcal{C}(\mathbf{x})$ holds) and proposes that in this case the algorithm should focus all its attention on locating a such a point:

$$\alpha_{\mathrm{EIC}}(\mathbf{x}) = \begin{cases} \alpha_{\mathrm{EI}}(\mathbf{x}) \prod_{k=1}^{K} \mathbb{P}(c_k(\mathbf{x}) \geq 0), & \text{if } \exists \mathbf{x}\,\text{s.t.}\,\mathcal{C}(\mathbf{x}) \\ \prod_{k=1}^{K} \mathbb{P}(c_k(\mathbf{x}) \geq 0), & \text{otherwise} \end{cases} \tag{4.8}$$

4.2.2 Integrated expected conditional improvement

The EIC acquisition function is structured to avoid sampling infeasible points, but it may be beneficial to sample such points in order to provide information in the long run. This drawback is a symptom of the wider myopia of expected improvement. To remedy this Gramacy and Lee [56] propose using a so-called *integrated expected conditional improvement* (IECI) acquisition function. IECI is also improvement-based like EI, but is designed to have additional look-ahead to help it make better long-term decisions.

In particular, rather than being based on the improvement function, IECI is based on the conditional improvement function $\mathrm{I}(\cdot \mid \cdot)$. It is a measure of the improvement at a reference point $\mathbf{x}'$ after a candidate point $\mathbf{x}$ has been chosen to be sampled, but not yet evaluated:

$$\mathrm{I}(\mathbf{x}' \mid \mathbf{x}) \equiv \max\{0, \nu - f(\mathbf{x}' \mid \mathbf{x})\} \tag{4.9}$$

GP models are not dynamic, and so the distribution of $f(\mathbf{x}' \mid \mathbf{x})$ remains $f(\mathbf{x}')$ prior to the observation at $\mathbf{x}$. However, it is in fact possible to deduce what the predictive variance of the updated model will be once that observation has happened. Recall that the predictive variance equation 2.33 is not dependent on the actual values observed $\mathbf{y}$, only the training data locations $\mathbf{x}_{1:n}$. Following this observation, Gramacy and Lee [56] *define* the distribution of $f(\mathbf{x}' \mid \mathbf{x})$ to be Gaussian with predictive mean $\mu(\mathbf{x}')$ conditioned on the data actually observed and predictive variance $\sigma^2_{\mathbf{x}}(\mathbf{x}')$ conditioned on the data actually observed together with the unobserved $\mathbf{x}$. The closer $\mathbf{x}'$ is to $\mathbf{x}$ the greater the reduction in predictive variance, assuming that the kernel function is concave. Therefore, taking the expectation of equation 4.9 simply amounts to replacing $\sigma^2(\mathbf{x}')$ in equation 3.9 by $\sigma^2_{\mathbf{x}}(\mathbf{x}')$

$$\mathbb{E}[\mathrm{I}(\mathbf{x}' \mid \mathbf{x})] = \sigma_{\mathbf{x}}(\mathbf{x}')[z(\mathbf{x}')\Phi(z(\mathbf{x}')) + \phi(z(\mathbf{x}'))] \tag{4.10}$$

where $z(\mathbf{x}') = (\nu - \mu(\mathbf{x}'))/\sigma_{\mathbf{x}}(\mathbf{x}')$.

Taking a step back for a second, remember that we are looking for an acquisition function which can be maximized as a function of $\mathbf{x}$ only. For this reason we first integrate over the reference point $\mathbf{x}' \in \mathcal{X}$, with arbitrary well-behaved distribution $g(\mathbf{x}')$. Secondly, notice that we prefer candidate points which *minimize* expected conditional improvement, for example by reducing $\sigma_{\mathbf{x}}(\mathbf{x}')$ in equation 4.10. As a result we negate the integral to give a general IECI acquisition function, defined to

be:

$$\alpha_{\text{IECI}}(\mathbf{x}) \equiv -\int_{\mathbf{x}'\in\mathcal{X}} \mathbb{E}[\text{I}(\mathbf{x}' \mid \mathbf{x})]g(\mathbf{x}')\, d\mathbf{x}' \tag{4.11}$$

Rather than think of minimizing the expected conditional improvement, it is perhaps more intuitive to think of maximizing the expected reduction in improvement. Speaking heuristically, the amount of improvement in a system is finite and should decrease with each additional observation, reaching zero when a minimizer $\mathbf{x}^\star$ is found. Choosing candidate points $\mathbf{x}$ which maximize this reduction should lead to an efficient optimization policy.

These two approaches are analogous. To make the connection clear, we use the fact that acquisition functions can scaled additively without changing the maxima - and therefore leading to the same decision making in practise. Since $\mathbb{E}[I(\mathbf{x}')]g(\mathbf{x}')$ is entirely independent of $\mathbf{x}$ we may rescale equation 4.11 to give the equivalent integrated expected reduction in improvement (IERI) acquisition function:

$$\begin{aligned}\alpha_{\text{IERI}}(\mathbf{x}) &\equiv \int_{\mathbf{x}'\in\mathcal{X}} \mathbb{E}[I(\mathbf{x}')]g(\mathbf{x}')\, d\mathbf{x}' - \int_{\mathbf{x}'\in\mathcal{X}} \mathbb{E}[\text{I}(\mathbf{x}' \mid \mathbf{x})]g(\mathbf{x}')\, d\mathbf{x}' \\ &= \int_{\mathbf{x}'\in\mathcal{X}} \big(\mathbb{E}[I(\mathbf{x}')] - \mathbb{E}[\text{I}(\mathbf{x}' \mid \mathbf{x})]\big)\, g(\mathbf{x}')\, d\mathbf{x}' \end{aligned} \tag{4.12}$$

This is all well and good, but we still have the problems of dealing with the distribution $g(\mathbf{x}')$ and incorporating the constraints. However, the IECI acquisition function is built to be able to deal naturally with both these problems at once. This is accomplished by using the density $g(\mathbf{x}')$ to favour reductions in expected improvement in regions likely to be feasible. In particular, taking $g(\mathbf{x}')$ to be the probability of satisfying each of the K black-box constraints c_k gives the constrained IECI acquisition function:

$$\alpha_{\text{IECI}}(\mathbf{x}) \equiv -\int_{\mathbf{x}'\in\mathcal{X}} \mathbb{E}[\text{I}(\mathbf{x}' \mid \mathbf{x})] \prod_{k=1}^{K} \mathbb{P}(c_k(\mathbf{x}') \geq 0)\, d\mathbf{x}' \tag{4.13}$$

This is the form of IECI that we will use in this project. One drawback is that this integral is not tractable in general and is instead approximated [56].

Chapter 5

Experiments

The two-dimensional Branin-Hoo function is often used as a bench-mark for optimization [57] [58]. For our experiments, we use a deterministic version [59] rescaled to the unit square $[0, 1]^2$ with

$$f(\mathbf{x}) = \frac{1}{51.95}\left[(\bar{x}_2 - \frac{5.1}{4\pi^2}\bar{x}_1^2 + \frac{5}{\pi}\bar{x}_1 - 6)^2 + 10(1 - \frac{1}{8\pi}\cos(\bar{x}_1)) - 44.81\right] \tag{5.1}$$

where $\bar{x}_1 = 15x_1 - 5$ and $\bar{x}_2 = 15x_2$. This rescaled version has mean zero and variance one. It is a multi-modal function with global minima -1.047 at three locations: $(0.124, 0.818)$, $(0.543, 0.152)$ and $(0.962, 0.165)$, to three decimal places.

5.1 Unconstrained

To demonstrate the capability of Bayesian optimization we first optimize the unconstrained Branin-Hoo function.

The optimization budget is taken to be $N = 20$ function evaluations. An initial dataset $\mathcal{D}_0$ of five points is generated by latin-hypercube sampling (LHS), leaving fifteen iterations to be selected by the chosen acquisition function.

The implementation of is written in R using the `laGP` [40] package, based on the lecture notes of Gramacy [11]. For simplicity an isotropic squared exponential kernel is used. To ease computation the prior mean function μ_0 of the GP is set to be zero and the amplitude σ_f^2 of the kernel is set to be one. The length-scale l of the kernel is tuned at each iteration of the algorithm to be the maximum likelihood estimator. The acquisition function used is expected improvement with incumbent

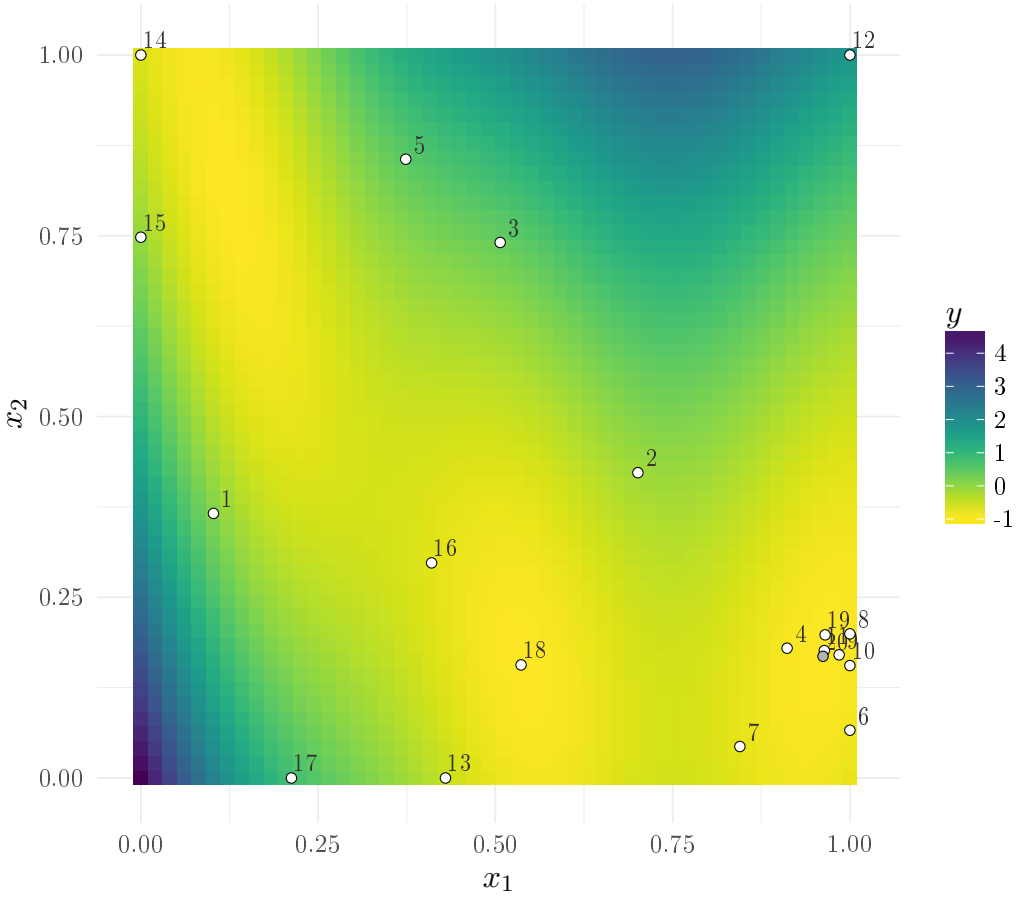


FIGURE 5.1: Density plot of modified Branin-Hoo function, as specified by equation 5.1. Example numbered Bayesian optimization path plotted as white circles. Points one to five are generated by latin-hypercube sampling and points six to twenty are chosen by maximizing EI acquisition function. Minimum sample location, shown as the grey circle, is the final point $(0.965, 0.198)$ with value -1.043 to three decimal places.

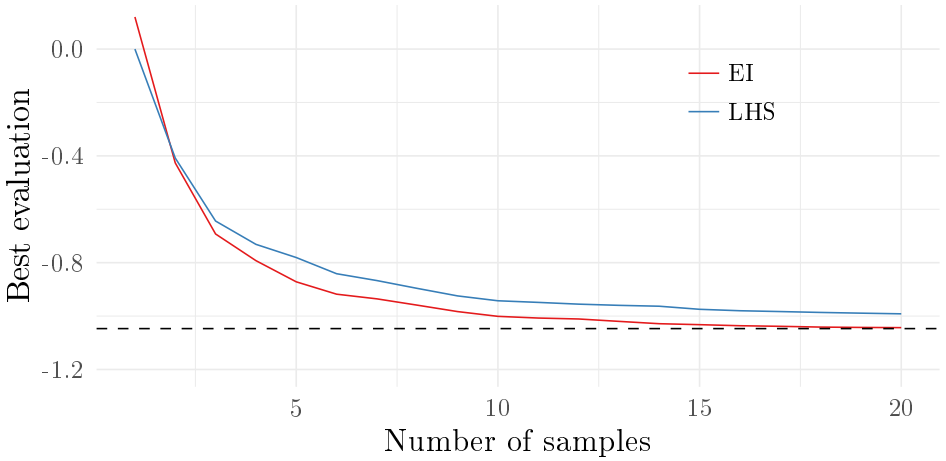


FIGURE 5.2: The progress, averaged over 50 runs, of both EI and LHS. Recall that the first five points chosen by both methods are by LHS, so the variation for these samples is due to randomness. However from that point onwards it is clear that EI convincingly converges on the true global minimum -1.047, shown by the horizontal dashed line, whereas LHS makes comparatively slow progress.

value ν set to be the minimum function value observed so far. One benefit of choosing EI is that it does not require any additional parameters, such as the constant λ for LCB. Both the optimization of the log-likelihood, given by equation 2.35, and the optimization of the acquisition function are performed using five random restarts of `method="L-BFGS-B"` via the default function `optim`.

Figure 5.1 shows the result of one run of the Bayesian optimization procedure, which we henceforth refer to as EI, described above. Of course the results of any single run are not statistically significant because the first five samples locations, together with the starting locations of the log-likelihood and acquisition function optimization procedures, are random.

To average out some of the variation we repeat the EI procedure independently 50 times. To provide a baseline for comparison we also run 50 repeats where all 20 points are chosen by LHS. Figure 5.2 compares the average performance of each procedure. It is clear that the performance of EI in this instance is superior: 29 out of the 50 times EI finds a global minimum to three decimal places compared to only once out of 50 for LHS.

5.2 Constrained

Similarly to the set-up in Gelbart [20] and Griffiths [53] we now consider as before the optimization the Branin-Hoo function, but now with the additional, deterministic, black-box disk constraint $c(\mathbf{x}) \geq 0$, where:

$$c(\mathbf{x}) = \frac{2}{9} - (x_1 - \frac{1}{2})^2 - (x_2 - \frac{1}{2})^2 \tag{5.2}$$

This constraint excludes points outside a circle of radius $\sqrt{2}/3$ centred at $(0.5, 0.5)$. Of the three global minimizers of the unconstrained objective function, two now lie outside the feasible region, leaving only one feasible global minimizer, $(0.543, 0.152)$. It is assumed that the constraint is coupled, so that f and c are evaluated jointly.

We will test the performance of the two constrained optimization acquisition functions introduced in Chapter 4, EIC and constrained IECI, together with LHS. As before, the optimization budget is $N = 20$ and the first five iterations of each method are chosen by LHS.

In both the EIC and IECI implementations the constraint, like the objective function, is modelled by a GP with squared exponential kernel and the incumbent value ν is set to be the minimum function value observed so far. Whereas EIC is entirely tractable, the integral in IECI is approximated by a sum weighted by the probability of satisfying the constraint. For this reason IECI was substantially more computationally demanding.

Figure 5.3 illustrates the region which is now infeasible due to the constraint and shows an example EIC optimization path. Figure 5.4 shows the surrogate functions to the objective and constraint functions, respectively, after the 20 points in the previous figure have been sampled. In this specific instance it is clear that the surrogates are well fitted to both the objective and constraint.

Overall, across 50 repeats, we find that EIC performs slightly better than IECI, with both performing much better than LHS. In particular, the average of the minimum feasible function evaluations found is -1.037 for EIC, -1.032 for IECI and -0.966 for LHS. This is shown by figures 5.5 and 5.6.

One explanation for why EIC may have outperformed IECI in this particular problem is because of the simplicity of the constraint. In particular, there is not much that can be learned from sampling in the infeasible region as the global minimizer is a fair distance from it. Perhaps IECI may prove its value for problems with more

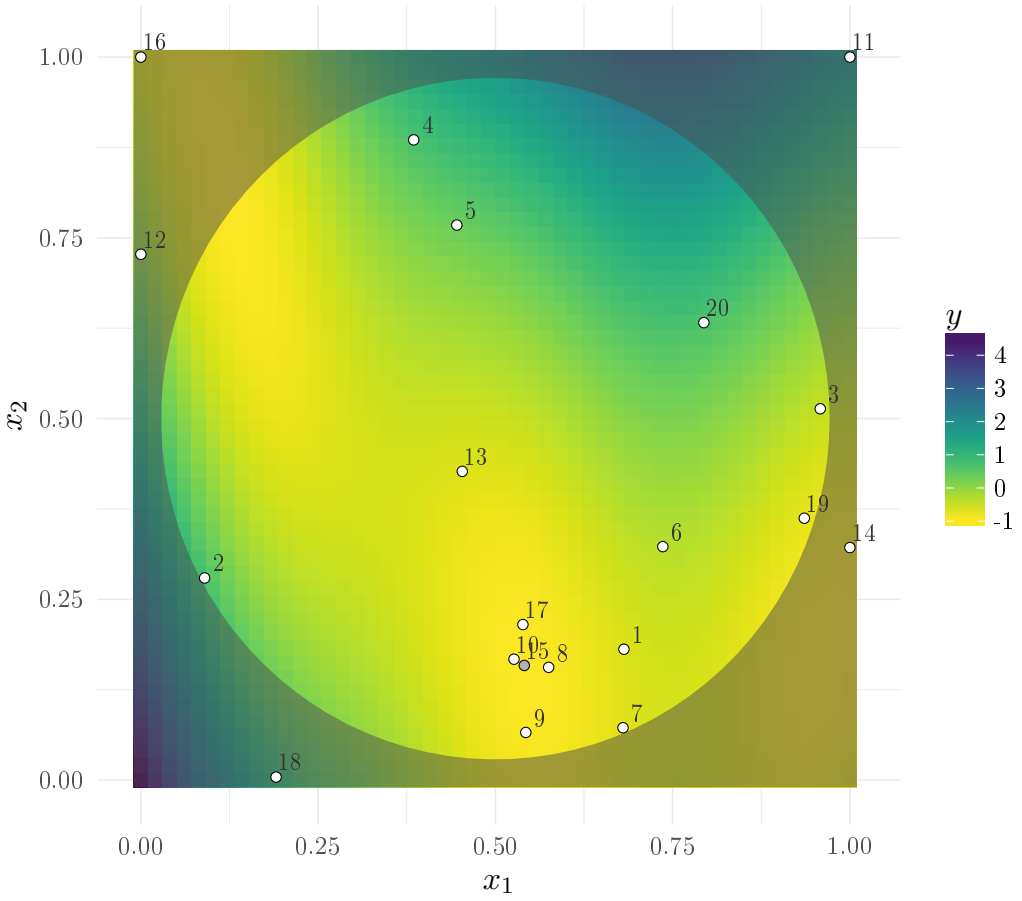


FIGURE 5.3: As in figure 5.1 a density plot of Branin-Hoo function is shown. The region which is infeasible, that is $c(\mathbf{x}) < 0$, is shaded in a darker colour. The white circles show a numbered example EIC optimization path, with the grey circle showing minimum feasible sample location $(0.541, 0.158)$ with value -1.047 to three decimal places. In this instance EIC is able to find the one feasible global minimizer. It is interesting to note samples 11, 12, 14, 16 and 18 are taken close to the edge of the unit square.

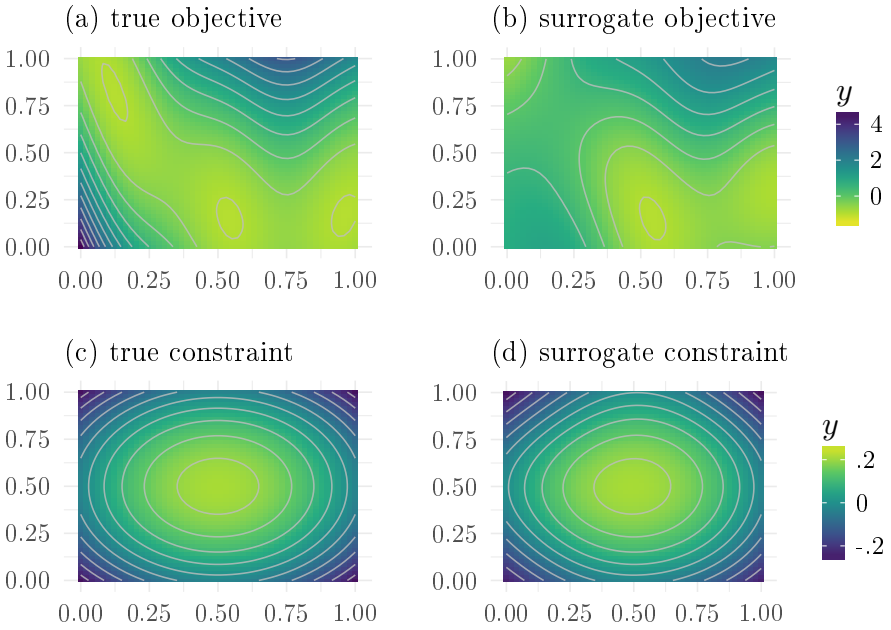


FIGURE 5.4: Plots (a) and (b) show the true objective function and the mean of its GP emulator after the 20 points in figure 5.3 have been sampled. To allow for meaningful comparison, they have been rescaled to a common colour scale. Although the surrogate is overall a good fit, it misses the minimizer in the top left. This corresponds to the fact that there are no samples in that location in figure 5.3. It is also instructive to note that the surrogate function does not fit well in regions where the objective function is high, and that this is not a bad thing. We are only interested in accurately learning the objective function in regions where it may plausibly have a global minimizer. Similarly, plots (c) and (d) show the true constraint function and the mean of its GP emulator after the 20 samples in figure 5.3, on a common colour scale (which is oppositely oriented, because we are looking for positive constraint values). The surrogate is very similar to the true constraint, perhaps due to the constraints simplicity.

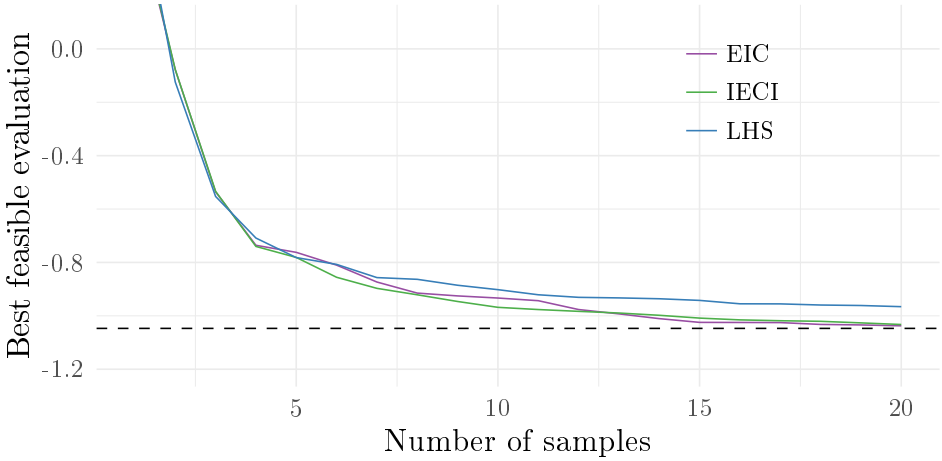


FIGURE 5.5: The progress, averaged over 50 runs, of the EIC, IECI and LHS implementations. As in figure 5.2 the true global minimum -1.047 is shown by the horizontal dashed line.

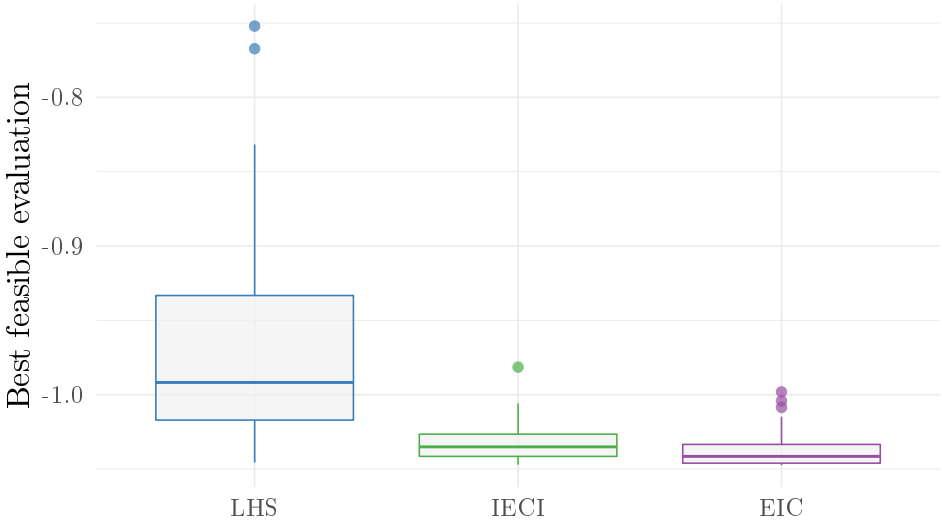


FIGURE 5.6: Boxplot of the best feasible evaluation found after 20 samples for each method, across the 50 runs.

complicated feasible regions. Another explanation is that the IECI implementation is only an approximation: it may be that the true version would perform better. This is plausible because a previous implementation of IECI was run with a faster, less accurate, approximation which averaged only -1.017.

5.3 Discussion

These results show the promise of Bayesian optimization and that the algorithm works essentially as hoped, for both unconstrained and unconstrained optimization problems. Of course from a critical point of view there are some caveats which should be mentioned.

Firstly, in both the constrained and unconstrained cases, the Bayesian optimization implementations are in some sense unfairly helped by the objective function being scaled to have mean zero and variance one, as they have not had to learn these facts from the data.

The Branin-Hoo function is not a particularly difficult optimization problem. As all of the local minimizers are global so there is no threat of placing too great an emphasis on local search. That being said, there is circumstantial evidence of a balanced approach in figures 5.1 and 5.3. Furthermore, it is only two-dimensional which, while being convenient for visualisation, is not particularly demanding. One current challenge is scaling Bayesian optimization to be more effective in higher dimensions, as for the time being it is restricted to roughly 10 dimensions [60].

Furthermore, both the objective and constraint are taken to be deterministic. Had property P4 been the case the results surely would have been less impressive. Problems with high noise-levels are particularly challenging and can lead to odd, pathological, behaviour if not considered more thoroughly [20] [51].

Finally, it almost goes without saying that we have only tested performance on one specific function - more rigorous benchmarks should test on a variety of objective functions.

Chapter 6

Conclusion

In Chapter 1 we began by introducing four properties of the objective function and the challenges they pose to optimization. Via discussion of three sample design strategies we presented some of the prominent considerations, principles and trade-offs in such a problem. This motivated a sequential, model-based approach - known as Bayesian optimization.

Chapter 2 explores Gaussian process emulators to the objective function, which are the most commonly used statistical model for Bayesian optimization. We discussed some of the design challenges, such as the choice of kernel function and handling of hyper-parameters, in the practical implementation of GPs. Chapter 3 explains the role of the acquisition function and gives three examples. Our first experiment, in section 5.1, demonstrated the application of this theory, resulting in a substantial performance improvement on LHS.

We have also considered the extension of Bayesian optimization to deal with black-box, inequality constraints. Chapter 4 introduces two constrained acquisition functions, expected improvement with constraints and integrated expected conditional improvement, which are tested against each other in our second experiment, section 5.2. The results of this experiment seem to suggest that for simple feasible regions EIC may be preferable to IECI - quite how plausible such simplistic feasible regions are in practice is questionable.

The discussion of constrained Bayesian optimization was by no means exhaustive: two suggestions for further study are the work of Gramacy, Gray, Le Digabel, *et al.* [61] on augmented Lagrangian approaches and the work of Hernández-Lobato, Gelbart, Hoffman, *et al.* [62] on so-called predictive entropy search with constraints.

Needless to say, there is still a great deal of work to be done on Bayesian optimization. A look at last years NIPS workshop [63] gives an idea of the direction moving forward: scaling to larger data-sets, moving beyond GPs and making the overall Bayesian optimization toolbox more comprehensive and easy to use. A great deal of this work is directly driven by problems in science and engineering and as such there is a lot of room for collaboration. A number of companies have also been founded aiming to use provide Bayesian optimization as a service [64]. This all goes to say that it is a dynamic and exciting research area to be a part of.

Bibliography

[1] R. T. Rockafellar, "Lagrange multipliers and optimality", *SIAM review*, vol. 35, no. 2, pp. 183–238, 1993.

[2] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. E. Karro, and D. Sculley, Eds., *Google Vizier: A Service for Black-Box Optimization*, 2017. [Online]. Available: `http://www.kdd.org/kdd2017/papers/view/google-vizier-a-service-for-black-box-optimization`.

[3] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms", *ArXiv e-prints*, Jun. 2012. arXiv: `1206.2944 [stat.ML]`.

[4] 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*, vol. 104, no. 1, pp. 148–175, 2016.

[5] R. Horst and P. M. Pardalos, *Handbook of global optimization*. Springer Science & Business Media, 2013, vol. 2.

[6] D. J. Lizotte, "Practical Bayesian optimization", AAINR46365, PhD thesis, Edmonton, Alta., Canada, 2008, ISBN: 978-0-494-46365-9.

[7] B. Betrò, "Bayesian methods in global optimization", *Journal of Global Optimization*, vol. 1, no. 1, pp. 1–14, 1991, ISSN: 1573-2916. DOI: `10.1007/BF00120661`. [Online]. Available: `https://doi.org/10.1007/BF00120661`.

[8] E. Brochu, "Interactive Bayesian optimization: learning parameters for graphics and animation", PhD thesis, PhD thesis, University of British Columbia, 2010.

[9] S. Streltsov and P. Vakili, "A non-myopic utility function for statistical global optimization algorithms", *Journal of Global Optimization*, vol. 14, no. 3, pp. 283–298, 1999.

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization", *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[11] R. B. Gramacy, *STAT 6984: Response surface methods and computer experiments*, Lecture Notes.

[12] N. Lawrence. (2017). What is machine learning?, [Online]. Available: `http://inverseprobability.com/2017/07/17/what-is-machine-learning`.

[13] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise", *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.

[14] J. Mockus, "On Bayesian methods for seeking the extremum", in *Proceedings of the IFIP Technical Conference*, London, UK, UK: Springer-Verlag, 1974, pp. 400–404, ISBN: 3-540-07165-2. [Online]. Available: `http://dl.acm.org/citation.cfm?id=646296.687872`.

[15] R. T. Cox, "Probability, frequency and reasonable expectation", *American Journal of Physics*, vol. 14, no. 1, pp. 1–13, 1946. DOI: `10.1119/1.1990764`. eprint: `https://doi.org/10.1119/1.1990764`. [Online]. Available: `https://doi.org/10.1119/1.1990764`.

[16] B. Bischl, J. Richter, J. Bossek, D. Horn, J. Thomas, and M. Lang, "mlrMBO: A modular framework for model-based optimization of expensive black-box functions", *arXiv preprint arXiv:1703.03373*, 2017.

[17] H. Robbins, "Some aspects of the sequential design of experiments", in *Herbert Robbins Selected Papers*, Springer, 1985, pp. 169–177.

[18] I. O. Ryzhov, W. B. Powell, and P. I. Frazier, "The knowledge gradient algorithm for a general class of online learning problems", *Operations Research*, vol. 60, no. 1, pp. 180–195, 2012. DOI: `10.1287/opre.1110.0999`. [Online]. Available: `https://doi.org/10.1287/opre.1110.0999`.

[19] F. M. Nyikosa, M. A. Osborne, and S. J. Roberts, "Adaptive Bayesian optimisation for online portfolio selection", in *Workshop on Bayesian Optimization at NIPS*, vol. 2015, 2015.

[20] M. A. Gelbart, "Constrained Bayesian optimization and applications", PhD thesis, Harvard University, Graduate School of Arts and Sciences, 2015.

[21] J. Einbeck, *Statistical methods*, Lecture Notes.

[22] W. J. Krzanowski, Ed., *Principles of Multivariate Analysis: A User's Perspective*. New York, NY, USA: Oxford University Press, Inc., 1988, ISBN: 0-198-52211-8.

[23] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005, ISBN: 026218253X.

[24] E. L. Snelson, *Flexible and efficient Gaussian process models for machine learning*. University of London, University College London (United Kingdom), 2008.

[25] D. Duvenaud, "Automatic model construction with Gaussian processes", PhD thesis, University of Cambridge, 2014.

[26] M. L. Stein, *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[27] B. Matérn, *Spatial variation*. Springer Science & Business Media, 2013, vol. 36.

[28] T. J. Santner, B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*. Springer Science & Business Media, 2013.

[29] R. M. Neal, "Bayesian Learning for Neural Networks", 1996.

[30] M. B. Christopher, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[31] K. Vafa, "Training and Inference for Deep Gaussian Processes", 2016.

[32] P. Orbanz and Y. W. Teh, "Bayesian nonparametric models", in *Encyclopedia of Machine Learning*, Springer, 2011, pp. 81–89.

[33] A. Shah, A. Wilson, and Z. Ghahramani, "Student-t processes as alternatives to Gaussian processes", in *Artificial Intelligence and Statistics*, 2014, pp. 877–885.

[34] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression", *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[35] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs", in *Advances in neural information processing systems*, 2006, pp. 1257–1264.

[36] D. J. MacKay, "Bayesian interpolation", *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[37] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization", *SIAM Journal on Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[38] O. Roustant, D. Ginsbourger, and Y. Deville, "DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization", 2012.

[39] B. MacDonald, P. Ranjan, and H. Chipman, "GPfit: an R package for Gaussian process model fitting using a new optimization algorithm", *arXiv preprint arXiv:1305.0759*, 2013.

[40] R. B. Gramacy *et al.*, "laGP: Large-scale spatial modeling via local approximate gaussian processes in R", *Journal of Statistical Software*, vol. 72, no. 1, pp. 1–46, 2016.

[41] G. Loomes and R. Sugden, "Regret theory: An alternative theory of rational choice under uncertainty", *The Economic Journal*, vol. 92, no. 368, pp. 805–824, 1982, ISSN: 00130133, 14680297. [Online]. Available: `http://www.jstor.org/stable/2232669`.

[42] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian Process optimization in the bandit Setting: no regret and experimental design", *ArXiv e-prints*, Dec. 2009. arXiv: `0912.3995 [cs.LG]`.

[43] R. Lam, K. Willcox, and D. H. Wolpert, "Bayesian optimization with a finite budget: An approximate dynamic programming approach", in *Advances in Neural Information Processing Systems*, 2016, pp. 883–891.

[44] J. González, M. Osborne, and N. D. Lawrence, "GLASSES: Relieving the myopia of Bayesian optimisation", *ArXiv e-prints*, Oct. 2015. arXiv: `1510.06299 [stat.ML]`.

[45] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces", *Journal of global optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[46] M. Schonlau, W. J. Welch, and D. Jones, "Global optimization with nonparametric function fitting", *Proceedings of the ASA, section on physical and engineering sciences*, pp. 183–186, 1996.

[47] A. D. Bull, "Convergence rates of efficient global optimization algorithms", *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2879–2904, 2011.

[48] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions", *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[49] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules", *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[50] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints.", in *ICML*, 2014, pp. 937–945.

[51] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Constrained Bayesian optimization with noisy experiments", *arXiv preprint arXiv:1706.07094*, 2017.

[52] C. Caiado and F. Coolen, *Decision theory*, Lecture Notes.

[53] R.-R. Griffiths and J. M. Hernández-Lobato, "Constrained Bayesian Optimization for Automatic Chemical Design", *ArXiv e-prints*, Sep. 2017. arXiv: `1709.05501 [stat.ML]`.

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

[55] J. Snoek, "Bayesian optimization and semiparametric models with applications to assistive technology", PhD thesis, Citeseer, 2013.

[56] R. B. Gramacy and H. K. H. Lee, "Optimization Under Unknown Constraints", *ArXiv e-prints*, Apr. 2010. arXiv: `1004.4027 [stat.ME]`.

[57] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown, "Towards an empirical foundation for assessing Bayesian optimization of hyperparameters", in *NIPS workshop on Bayesian Optimization in Theory and Practice*, vol. 10, 2013.

[58] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization", *Structural and Multidisciplinary Optimization*, vol. 48, no. 3, pp. 607–626, 2013.

[59] *Virtual library of simulation experiments: Test functions and datasets*, `https://www.sfu.ca/~ssurjano/branin.html`, Accessed: 2018-04-01.

[60] Z. Wang, "Practical and theoretical advances in Bayesian optimization", PhD thesis, 2016.

[61] R. B. Gramacy, G. A. Gray, S. Le Digabel, H. K. Lee, P. Ranjan, G. Wells, and S. M. Wild, "Modeling an augmented lagrangian for blackbox constrained optimization", *Technometrics*, vol. 58, no. 1, pp. 1–11, 2016.

[62] J. M. Hernández-Lobato, M. A. Gelbart, M. W. Hoffman, R. P. Adams, and Z. Ghahramani, "Predictive entropy search for Bayesian optimization with unknown constraints", 2015.

[63] *BayesOpt 2017*, `https://bayesopt.github.io/`, Accessed: 2018-04-17.

[64] M. Osborne, "Bayesian optimisation is probabilistic numerics", Probabilistic Numerics Workshop, Alan Turing Institute, 2018.