Interactive Bayesian Optimization

Learning User Preferences for Graphics and Animation

by

Eric Brochu

BA, University of Regina, 1997

BSc, University of Regina, 1998

MSc, The University of British Columbia, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

The University of British Columbia

(Vancouver)

December 2010

Abstract

Bayesian optimization with Gaussian processes has become increasingly popular in the machine learning community. It is efficient and can be used when very little is known about the objective function, making it useful for optimizing expensive black box functions. We examine the case of using Bayesian optimization when the objective function requires feedback from a human. We call this class of problems *interactive* Bayesian optimization. Here, we assume a parameterized model, and a user whose task is to find an acceptable set of parameters according to some perceptual value function that cannot easily be articulated. This requires special attention to the qualities that make this a unique problem, and so, we introduce three novel extensions: the application of Bayesian optimization to "preference galleries", where human feedback is in the form of preferences over a set of instances; a particle-filter method for learning the distribution of model hyperparameters over heterogeneous users and tasks; and a bandit-based method of using a portfolio of utility functions to select sample points. Using a variety of test functions, we validate our extensions empirically on both low- and high-dimensional objective functions. We also present graphics and animation applications that use interactive Bayesian optimization techniques to help artists find parameters on difficult problems. We show that even with minimal domain knowledge, an interface using interactive Bayesian optimization is much more efficient and effective than traditional "parameter twiddling" techniques on the same problem.

Preface

All the work presented here has been performed under the supervision of Nando de Freitas. In addition, I have been fortunate enough to work with outstanding co-authors on several publications which have become parts of this thesis.

- The initial work in using Bayesian optimization for preference galleries (Chapter 3) was presented at NIPS and SIGGRAPH [Brochu *et al.*, 2007a; 2007b]. On these publications, we worked with Abhijeet Ghosh, who provided the parameterized Bidirectional Reflectance Distribution Function modelling and rendering code we used as the test problem (§6.3). Abhijeet also helped tailor the SIGGRAPH poster presentation to a graphics audience, which helped it to win First Prize at the SIGGRAPH Student Research Competition.

- Chapter 4 describes a novel method of using particle filters to track hyperparameter evolution over multiple users. This work was presented at the SIGGRAPH/Eurographics Symposium on Computer Animation [Brochu *et al.*, 2010a]. Here we worked with co-author Tyson Brochu, who created the animation application we used for testing. The animation method presented in §6.4.1 is a result of his research, and that section was largely written by Tyson. The implementation of the parameterized interface (but not the machine learning aspects) was designed in consultation with Tyson.

- The idea of using hedging methods to manage a portfolio of acquisition functions (Chapter 5) was greatly aided by discussions with Matt

Hoffman. Matt was instrumental in helping to test the idea and provided valuable input in the writing of Chapter 5. He also wrote the description of the control test problem in §5.3. The control experiment framework we used was coded by Matt with Hendrik Kück for their earlier publication [Hoffman *et al.*, 2009]. A version of the work has been made available as a technical report on the arXiv e-print archive [Brochu *et al.*, 2010b]. An expanded version is under preparation for future submission.

- Finally, much of Chapters 1, 2, 3 and 7 will appear as a chapter [Brochu *et al.*, 2011] in an upcoming book, and some parts of Chapter 2 were previously presented in an article for *Autonomous Robots* [Martinez–Cantin *et al.*, 2009]. The work of my co-authors in those publications is not part of this thesis.

The human subject experiments were approved by the University of British Columbia Behavioural Research Ethics Board under certificate H10-01682.

Table of Contents

List of Tables

List of Figures

List of Symbols

The symbols used in this thesis follow a few conventions for clarity.

- Lower-case, unbolded Latin and Greek letters (x, α, etc.) represent scalar quantities and functions.
- Bold, lower-case Latin and Greek letters ($\mathbf{x}, \boldsymbol{\alpha}$, etc.) represent vectors.
- Bold, upper-case Latin letters ($\mathbf{X}$, etc.) represent matrices.
- Calligraphic upper-case Latin letters ($\mathcal{X}$, etc.) represent sets and distributions.
- The subscript notation $Z_{a:b}$ indicates the sequence $Z_a, Z_{a+1}, \ldots, Z_b$.

We additionally use a number of symbols not listed here, where the symbols are needed for discussion, but not relevant to the rest of the thesis.

symbol	section	definition
$f(\cdot)$	1	objective function
$\mathcal{A}$	1	compact set being optimized
d	1	dimensionality of the compact set being optimized
$\mathbf{x}$	1	an instance in $\mathcal{A}$
$\mathbf{x}_n$	1.2	the n^{th} sequential sample drawn during an optimization
$\mathcal{D}$	1.2	a sample and observation pair, e.g. $\{\mathbf{x}, y\}$
$\mathbf{x}^\star$	2.1	the global argmax
$f(\mathbf{x}^\star)$	2.1	the global maximum
$u(\cdot)$	2.1	an acquisition function

symbol	section	definition
$y, y(\cdot)$	2.1	a (possibly noisy) sample from the objective function
$y_n, y(\mathbf{x}_n)$	2.1	a (possibly noisy) sample from the objective function at the instance $\mathbf{x}_n$
ε_n	2.1	additive Gaussian noise at the instance $\mathbf{x}_n$
$\mathcal{D}_{1:t}$	2.1	a set of t observations, $\{\mathbf{x}_{1:t}, y_{1:t}\}$
$m(\cdot)$	2.1	the mean function of a Gaussian process
$\mathbf{x}^+$	2.1	incumbent of a set of points: $\text{argmax}_{\mathbf{x}_i} f(\mathbf{x}_i)$
$\mathbf{x}^{(\star)}$	2.1	the argmax of a local maximum
σ^2_{noise}	2.1	variance of Gaussian noise
$\mathcal{GP}(m(\cdot), k(\cdot,\cdot))$	2.1.1	Gaussian Process, defined by mean function m and covariance function k
$k(\cdot,\cdot)$	2.1.1	the kernel (covariance) function of a Gaussian process
f_t	2.1.1	noise-free observation of $\mathbf{x}_t$
$\mathbf{f}_{1:t}$	2.1.1	noise-free observations of $\mathbf{x}_{1:t}$
$\mathbf{K}$	2.1.1	kernel matrix
$\mathbf{k}$	2.1.1	vector of kernel function values for $\mathbf{x}_{t+1}$
$\mu(\cdot)$	2.1.1	posterior mean function
$\sigma^2(\cdot)$	2.1.1	posterior variance function
θ	2.1.2	length-scale hyperparameter
$\boldsymbol{\theta}$	2.1.2	vector of length-scale hyperparameters
$\text{PI}(\cdot)$	2.2.1	the "probability of improvement" acquisition function
$\text{I}(\cdot)$	2.2.1	the improvement function
$\text{EI}(\cdot)$	2.2.1	the "expected improvement" acquisition function
$\Phi(\cdot)$	2.2.1	cumulative distribution function of the standard Normal distribution
$\phi(\cdot)$	2.2.1	probability density function of the standard Normal distribution
ξ	2.2.1	exploration/exploitation trade-off hyperparameter for improvement-based acquisition functions
$\text{LCB}(\cdot)$	2.2.2	lower confidence bound acquisition function
$\text{UCB}(\cdot)$	2.2.2	upper confidence bound acquisition function
$\text{GP-UCB}(\cdot)$	2.2.2	Srinivas *et al.* variant of the confidence bound acquisition function
$r(\cdot)$	2.2.2	instantaneous regret function

symbol	section	definition
ν	2.2.2	exploration/exploitation trade-off hyperparameter for GP-UCB acquisition function
μ^+	2.3	the previously-observed sample with the highest expected value (the incumbent where noise is present)
$\mathbf{X}$	2.4.2	design matrix, where rows are inputs $\mathbf{x}_{1:t}$
$\mathbf{x}^{\text{first}}$	2.5	the value of the first function sample, or the argmax of the set of initial samples
G	2.5	the "gap" measure of optimization performance
$\mathbf{r}_i, \mathbf{c}_i$	3.2	i^{th} preference pair in a probit model. $\mathbf{r}_i$ and $\mathbf{c}_i$ are samples from $\mathcal{A}$, where $\mathbf{r}_i$ is preferred to $\mathbf{c}_i$.
$v(\cdot)$	3.2	valuation function
$\mathbf{f}_{\text{MAP}}$	3.2	maximum a posteriori estimates of $\mathbf{f}$ derived from a set of preference relationships
$\boldsymbol{\gamma}$	4.1	location parameter of mean hyperprior
$\boldsymbol{\Gamma}$	4.1	matrix of all location parameters γ
β	4.1	weight parameter of mean hyperprior
$\boldsymbol{\beta}$	4.1	vector of weight parameters of mean hyperprior
α	4.1	covariance hyperprior
ρ	4.2	generic hyperparameter
$\boldsymbol{\rho}$	4.2	vector of generic hyperparameters
τ	4.2	counter of runs of the optimization system on different objectives

List of Acronyms

Acronyms used in this thesis and the sections in which introduced.

symbol	section	definition
GP	2.1.1	Gaussian process
PI	2.2.1	probability of improvement
EI	2.2.1	expected improvement
GP-UCB	2.2.2	Gaussian process upper confidence bound
RBF	4.1	radial basis function
BRDF	6.3	bidirectional reflectance distribution functions

Acknowledgements

A martial arts student went to his teacher and said earnestly, "I am devoted to studying with you. How long will it take to master your art?"

The teacher's reply was casual: "Ten years."

Impatiently, the student answered, "But I want to master it faster than that! I will work very hard. I will practice every day, ten or more hours a day if I have to. How long will it take then?"

The teacher thought for a moment. "Twenty years."

— traditional Zen story

Since I started grad school all those years ago, Nando de Freitas has been not only a supervisor, but a friend, a mentor and a source of inspiration. He's had the patience to let me follow my own (sometimes misguided) research muse and the strength to see this thesis through to the end. It is with pride that I will always remain one of Nando's students. This thesis would not have been possible without him.

Thanks go to my fellow students, who have made these years so enjoyable and so rewarding, whether we were blasting techno through computer speakers in the wee hours before a paper deadline, discussing graphics and machine learning over sushi, or filling the sofas of "die Kommune" on West 23rd Avenue for marathon *Buffy the Vampire Slayer* viewings. Your friendship has kept me sane and your criticism has kept me honest. I won't try

to name you all for fear of leaving someone out, but you know who you are.

I've been lucky enough to be funded throughout most of my PhD: my Koerner and UBC Graduate fellowships and NSERC doctoral scholarship have made this research possible, and I am truly grateful. I've also been very fortunate to work with the brilliant and creative folks at Worio over the course of my degree, which has influenced my research in ways both large and small.

Last, but very far from least, I thank Janelle. Even when you were on the other side of the planet, you were standing beside me.

Chapter 1

Introduction

An enormous body of scientific literature has been devoted to the problem of optimizing a nonlinear function $f(\mathbf{x})$ over a compact set $\mathcal{A}$. In the realm of optimization, this problem is formulated concisely as follows:

$$\max_{\mathbf{x} \in \mathcal{A} \subset \mathbb{R}^d} f(\mathbf{x})$$

One typically assumes that the *objective* function $f(\mathbf{x})$ has a known mathematical representation, is convex, or is at least cheap to evaluate. Despite the influence of classical optimization on machine learning, many learning problems do not conform to these strong assumptions. Often, evaluating the objective function is expensive or even impossible, and the derivatives and convexity properties are unknown.

In many realistic sequential decision making problems, for example, one can only hope to obtain an estimate of the objective function by simulating future scenarios. Whether one adopts simple Monte Carlo simulation or adaptive schemes, as proposed in the fields of planning and reinforcement learning, the process of simulation is invariably expensive. Moreover, in some applications, drawing samples $f(\mathbf{x})$ from the function corresponds to expensive processes: drug trials, destructive tests or financial investments.

In this thesis, our motivating examples are based on active user modelling for learning parameters for procedural graphics and animation. In active

user modelling, $\mathbf{x}$ represents attributes of a user query—for example, a set of parameters for a procedural animation—and $f(\mathbf{x})$ requires a response from the human in the form of a rating or ranking. Computers must ask the right questions and the number of questions must be kept to a minimum so as to avoid annoying the user.

1.1 Motivation

> *A computer graphics artist sits down to use a simple renderer to find appropriate surfaces for a typical reflectance model. It has a series of parameters that must be set to control the simulation: "specularity", "Fresnel reflectance coefficient", and other, less-comprehensible ones. The parameters interact in ways difficult to discern. The artist knows in his mind's eye what he wants, but he's not a mathematician or a physicist—no course he took during his MFA covered Fresnel reflectance models. Even if it had, would it help? He moves the specularity slider and waits for the image to be generated. The surface is too shiny. He moves the slider back a bit and runs the simulation again. Better. The surface is now appropriately dull, but too dark. He moves a slider down. Now it's the right colour, but the specularity doesn't look quite right any more. He repeatedly bumps the specularity back up, rerunning the renderer at each attempt until it looks right. Good. Now, how to make it look metallic...?*

Problems in simulation, animation, rendering and other areas often take such a form, where the desired end result is identifiable by the user, but parameters must be tuned in a tedious trial-and-error process. This is particularly apparent in psychoperceptual models, where continual tuning is required to make something "look right". Unfortunately, it is not at all easy to find a mapping from parameterized animation to psychoperceptual plausibility. *The perceptual objective function is simply unknown.* However, it is fairly easy for humans to judge the quality of an instance in many graphics and animation problems—in fact, it is trivial and almost instantaneous. The

application of this principle to animation and other psychoperceptual tools is motivated by the observation that humans often seem to be forming a mental model of the objective function. This model enables them to *exploit* feasible regions of the parameter space where the value is predicted to be high and to *explore* regions of high uncertainty. It is our thesis that the process of tweaking parameters to find a result that looks "right" is akin to sampling a perceptual objective function, and that twiddling the parameters to find the best result is, in essence, optimization. Our objective function is the psychoperceptual process underlying judgement—how well a realization fits what the user has in mind.

Conventional parameter-tweaking techniques are often inefficient. For example, Hutter *et al.* [2007] were able to significantly speed up algorithms by automatically tuning the parameters—even improving on the best performance achieved by the algorithm designers themselves. They note that "[t]he fact that these empirical results contradicted the algorithm designer's intuition illustrates clearly the limitations of even an expert's ability to comprehend the complex interplay between the many parameters of a sophisticated heuristic algorithm." By using automatic techniques to help select sample points, we can allow the artist to focus on the task of identification, and possibly fine-tuning: the methods presented in this thesis are intended to work alongside familiar tools, not to replace them. In all this, the goal is to separate high-level cognitive tasks performed by human users from tedious low-level tasks that can be left to machines. At the same time, we need to help the user perform tasks efficiently—cognitive effort (attention) is the limiting resource in effective decision-making [Simon, 1978], and reducing the cognitive effort required for a problem often leads to improved accuracy [Russo and Leclerc, 1991].

This motivates a key insight: *it is not necessary to accurately model the entire objective function.* The problem is actually one of optimization, not regression. We can't directly maximize the user's psychoperceptual model, so we use an *acquisition function* (§2.2) as a principled way of trading off exploration (showing the user examples unlike any they have seen) and exploitation (trying to show the user improvements on examples they have

indicated preference for). Of course, regression-based learning can produce an accurate model of the user's valuation over the entire space of parameters, which would also allow us to find the best value. However, this comes at the cost of asking the user to evaluate many, many examples that have no practical relation what she is looking for. Our method tries instead to make the most efficient possible use of the user's time and cognitive effort: a problem well-suited to Bayesian optimization.

Furthermore, the value function can be *any* psychoperceptual process that lends itself to sliders and preferences: the model can support an animator looking for a particular "cartoon physics" effect, an artist trying to capture a particular mood in the lighting of a scene, or an electronic musician looking for a specific sound or rhythm. Though we use animation and rendering as motivating domains, our work has a broad scope of application in music and other arts, as well as psychology, marketing and econometrics, and human-computer interfaces.

To avoid confusion with other utility models in this thesis, we borrow from the econometrics literature and refer to this objective as the *value model* (or simply the *value function*). In assuming that the user's interests can be modelled this way, we are making an assumption that the value function is smooth, or can be reasonably modelled with a smooth function. In doing so, we follow the extensive work in the psychology and economics of decision-making (e.g., [Kahneman and Tversky, 1979; Payne *et al.*, 1993; McFadden, 2001; Train, 2003]), and direct the interested reader to these resources.

1.2 An introduction to Bayesian optimization

Bayesian optimization is a powerful strategy for finding the extrema of objective functions that are expensive to evaluate. It is applicable in situations where one does not have a closed-form expression for the objective function, but where one can obtain observations (possibly noisy) of this function at sampled values. It is particularly useful when these evaluations are costly, when one does not have access to derivatives, or when the problem at hand

is non-convex. We will provide an informal description here, to give the reader an intuition, and provide a more detailed discussion starting in §2.1.

Bayesian optimization techniques are some of the most efficient approaches in terms of the number of function evaluations required (see, e.g., [Močkus, 1994; Jones *et al.*, 1998; Streltsov and Vakili, 1999; Jones, 2001; Sasena, 2002]). Much of the efficiency stems from the ability of Bayesian optimization to incorporate prior belief about the problem to help direct the sampling, and to trade off exploration and exploitation of the search space. It is called *Bayesian* because it uses the famous "Bayes' theorem", which states (simplifying somewhat) that the *posterior* probability of a model (or theory, or hypothesis) M given evidence (or data, or observations) E is proportional to the *likelihood* of E given M multiplied by the *prior* probability of M:

$$P(M|E) \propto P(E|M)P(M).$$

Inside this simple equation is the key to optimizing the objective function. In Bayesian optimization, the *prior* represents our belief about the space of possible objective functions. Although the cost function is unknown, it is reasonable to assume that there exists prior knowledge about some of its properties, such as smoothness, and this makes some possible objective functions more plausible than others.

Let's define $\mathbf{x}_i$ as the ith sample, and $f(\mathbf{x}_i)$ as the observation of the objective function at $\mathbf{x}_i$. As we accumulate observations[1] $\mathcal{D}_{1:t} = \{\mathbf{x}_{1:t}, f(\mathbf{x}_{1:t})\}$, the prior distribution is combined with the likelihood function $P(\mathcal{D}_{1:t}|f)$. Essentially, given what we think we know about the prior, how likely is the data we have seen? If our prior belief is that the objective function is very smooth and noise-free, data with high variance or oscillations should be considered less likely than data that barely deviate from the mean (see, for example, Figure 4.2). Now, we can combine these to obtain our posterior distribution:

$$P(f|\mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t}|f)P(f).$$

The posterior captures our updated beliefs about the unknown objective

[1]Here we use subscripts to denote sequences of data, i.e., $y_{1:t} = \{y_1, \ldots, y_t\}$.

function. One may also interpret this step of Bayesian optimization as estimating the objective function with a *surrogate function* (also called a *response surface*), described formally in §2.1.1 with the mean function of a Gaussian process.

To sample efficiently, Bayesian optimization uses an acquisition function to determine the next location $\mathbf{x}_{t+1} \in \mathcal{A}$ to sample. The decision represents an automatic trade-off between exploration (where the objective function is very uncertain) and exploitation (trying values of $\mathbf{x}$ where the objective function is expected to be high). This optimization technique has the nice property that it aims to minimize the number of objective function evaluations. Moreover, it is likely to do well even in settings where the objective function has multiple local maxima.

Figure 1.1 shows a typical run of Bayesian optimization on a 1D problem. The optimization starts with two points. At each iteration, the acquisition function is maximized to determine where next to sample from the objective function—the acquisition function takes into account the mean and variance of the predictions over the space to model the utility of sampling. The objective is then sampled at the argmax of the acquisition function, the Gaussian process is updated and the process is repeated.

1.3 Contributions and overview

In this thesis, we present a novel, optimization-based approach to setting parameters interactively. In our approach, we implicitly model the user's *value* function from feedback on instances of images or animations generated from a space of valid parameter settings for procedural graphics and animation tasks. In effect, we are automating the "slider twiddling" process typically used by artists to find desired settings. The method is based on Bayesian optimization, which allows prior knowledge about the problem—whether from experts or simply regular users of the system—to be incorporated. We refer to this as *interactive* Bayesian optimization. Our goal is to offload the cognitive burden of estimating and exploring different sets of parameters from the user to the computer, though we can incorporate conventional

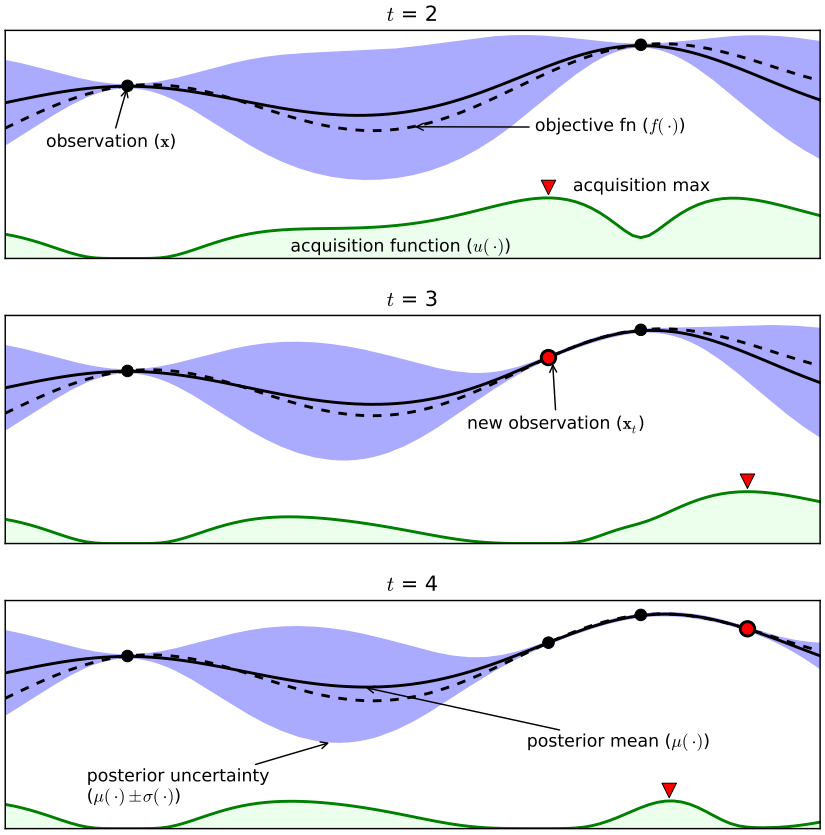


Figure 1.1: *An example of using Bayesian optimization on a toy 1D design problem. The figures show a Gaussian process (GP) approximation of the objective function over four iterations of sampled values of the objective function. The figure also shows the acquisition function in the lower shaded plots. The acquisition is high where the GP predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration)—areas with both attributes are sampled first. Note that the area on the far left remains unsampled, as while it has high uncertainty, it is (correctly) predicted to offer little improvement over the highest observation.*

slider twiddling into the framework easily.

In Chapter 2, we provide the necessary background for the problem. We formally present Bayesian optimization with Gaussian processes (§2.1) and describe various acquisition functions (§2.2) and the role of Gaussian

noise (§2.3). In §2.4, we cover the history of Bayesian optimization, and the related fields of kriging, GP experimental design and GP active learning. In §2.5, we present background information on the experimental methods we use in Chapters 3–5.

To deal with the unique problems involved in having humans in the loop, we introduce several novel extensions to Bayesian optimization, which form our main contributions to the field:

- In Chapter 3, we show how our optimization methods can be applied to "preference galleries". In a preference gallery interface, users need only indicate which of a set of generated instances look the most like the target. This is a much easier task for humans than reliably providing absolute magnitudes of single instances.

- Chapter 4 introduces a novel way of adaptively learning the model hyperparameters using particle filters. Because of the small number of data we collect during each optimization, we cannot learn hyperparameters through conventional means. However, we know that there is regularity among users and sessions, even when users have different goals. Our insight is that every time the application is used, a model is trained. While different runs might involve users with different simulation goals, we can track the distribution over hyperparameter settings across user sessions. This allows the system to exploit regularity between users and sessions and learn from all the users of the system.

- Effective Bayesian optimization requires the optimization of an acquisition function to select sample points. However, there are several different acquisition functions in the literature, none of which is guaranteed to outperform the others on an arbitrary unknown function. In Chapter 5, we present a method of adaptively updating a portfolio of acquisition functions. At each time step, one is selected, and the entire portfolio can be updated based on changes in the posterior. This method is shown to perform better than the best individual acquisition function in the portfolio.

In Chapter 6, we present practical applications of our model in design galleries that allow artists to find parameters efficiently for graphics and animation tasks. Our most-sophisticated interface also allows expert users to continue to use familiar parameter-finding techniques, and to restrict areas of the optimization. User expertise is a valuable resource, and it is important that our system augments, rather than replaces, existing techniques. We show that users are able to find instances efficiently using the tool, and adaptive prior updates make the system significantly and progressively better at modelling the user.

Finally, Chapter 7 is a brief discussion of interactive Bayesian optimization, future work, and advice to future practitioners.

Chapter 2

Background

Numerous researchers before us have faced the task of optimizing expensive, black-box functions. In this chapter[1], we will look at the existing work in this area. In §2.1–§2.3, we will detail the Gaussian process-based Bayesian optimization framework we will use as the basis of the work in the remainder of this thesis. In §2.4, we will look at the history and recent successes of Bayesian optimization and some of the related fields. We will also describe the experimental setting we use to evaluate our techniques in §2.5.

The basic concepts of optimization and Gaussian processes are well-covered in the open literature, so we do not spend a great deal of time re-presenting this material. However, as the devil is often in the details, the reader unfamiliar with the computational details of Gaussian process inference and optimization may wish to have a more thorough text at hand. In working with Gaussian processes, we extensively consulted Rasmussen and William's book on the topic [2006]. There is not yet an equivalent canonical text on Bayesian optimization, but we found the PhD theses of Schonlau [1997] and Lizotte [2008], and the book *The Design and Analysis of Computer Experiments* [Santner *et al.*, 2003] to be excellent resources while working on this thesis. In addition, the PhD thesis of Osborne [2010] does

[1] An earlier version of this chapter was the basis of an online tutorial [Brochu *et al.*, 2009], and a modified version will appear as part of an upcoming collection [Brochu *et al.*, 2011]. Small parts of this chapter have also appeared in our previously-published work in the field [Brochu *et al.*, 2007a; 2007b; 2010a; 2010b; Martinez–Cantin *et al.*, 2009].

an admirable job of covering Gaussian processes for prediction, optimization and quadrature, though as it was published after we had already written most of this chapter, it did not have a large impact on our work.

2.1 The Bayesian optimization approach

Optimization is a broad and fundamental field of mathematics. In order to harness it to our ends, we need to narrow it down by defining the conditions we are concerned with.

Our first restriction is to simply specify that the form of the problem we are concerned with is *maximization*, rather than the more common form of minimization. We adopt this form for convenience, because the main problems we are concerned with are optimizing the value and acquisition functions. The maximization of a real-valued function $\mathbf{x}^\star = \text{argmax}_\mathbf{x} f(\mathbf{x})$ can be regarded as the minimization of the transformed function

$$g(\cdot) = -f(\cdot).$$

We also assume that the objective is *Lipschitz-continuous*. That is, there exists some constant C, such that for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{A}$:

$$\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\| \leq C\|\mathbf{x}_1 - \mathbf{x}_2\|,$$

though C may be (and typically is) unknown.

We can narrow the problem down further by defining it as one of *global*, rather than *local* optimization. In local maximization problems, we need only find a point $\mathbf{x}^{(\star)}$ such that

$$f(\mathbf{x}^{(\star)}) \geq f(\mathbf{x}), \forall \mathbf{x} \text{ s.t. } \|\mathbf{x}^{(\star)} - \mathbf{x}\| < \epsilon.$$

If $-f(\cdot)$ is convex, then any local maximum is also a global maximum. However, in our optimization problems, we cannot assume that the negative objective function is convex. It *might* be the case, but we have no way of knowing before we begin optimizing.

It is common in global optimization, and true for our problem, that the objective is a *black box* function: we do not have an expression of the objective function that we can analyze, and we do not know its derivatives. Evaluating the function is restricted to querying at a point $\mathbf{x}$ and getting a (possibly noisy) response of $f(\mathbf{x})$ or (anti-)derivatives of f evaluated at $\mathbf{x}$. Black box optimization also typically requires that all dimensions have bounds on the search space. In our case, we can safely make the simplifying assumption these bounds are all axis-aligned, so the search space is a hyperrectangle of dimension d.

A number of approaches exist for this kind of global optimization and have been well-studied in the literature (e.g., [Törn and Žilinskas, 1989; Mongeau *et al.*, 1998; Liberti and Maculan, 2006; Zhigljavsky and Žilinskas, 2008]). Deterministic approaches include interval optimization and branch and bound methods. *Stochastic approximation* is a popular idea for optimizing unknown objective functions in machine learning contexts [Kushner and Yin, 1997]. It is the core idea in most reinforcement learning algorithms [Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998], learning methods for Boltzmann machines and deep belief networks [Younes, 1989; Hinton and Salakhutdinov, 2006] and parameter estimation for nonlinear state space models [Poyiadjis *et al.*, 2005; Martinez–Cantin *et al.*, 2006]. However, these are generally unsuitable for our domain because they still require many samples, and in the active user-modelling domain drawing samples is *expensive*.

Even in a noise-free domain, evaluating an objective function with known Lipschitz continuity L on a d-dimensional unit hypercube, guaranteeing the best observation $f(\mathbf{x}^+) \geq f(\mathbf{x}^\star) - \epsilon$ requires a minimax approach, requiring $(L/2\epsilon)^d$ samples [Betrò, 1991]. This can be an incredibly expensive premium to pay for insurance against unlikely scenarios. As a result, the idea naturally arises to relax the guarantees against pathological worst-case scenarios. The goal, instead, is to use evidence and prior knowledge to maximize the posterior at each step, so that each new evaluation decreases the distance between the true global maximum and the expected maximum given the model. This is sometimes called "one-step" [Močkus, 1994] "average-case"

[Streltsov and Vakili, 1999] or "practical" [Lizotte, 2008] optimization. This average-case approach has weaker demands on computation than the worst-case approach. As a result, it may provide faster solutions in many practical domains where one does not believe the worst-case scenario is plausible.

Bayesian optimization uses the prior and evidence to define a posterior distribution over the space of functions. The Bayesian model allows for an elegant means by which informative priors can describe attributes of the objective function, such as smoothness or the most likely locations of the maximum, even when the function itself is not known. Optimizing follows the principle of *maximum expected utility*, or, equivalently, *minimum expected risk*. The process of deciding where to sample next requires the choice of a utility function and a way of optimizing the expectation of this utility with respect to the posterior distribution of the objective function. This secondary optimization problem is usually easier because the utility is typically chosen so that it is easy to evaluate, though still nonconvex. To make clear which function we are discussing, we will refer to this utility as the *acquisition function* (also sometimes called the *infill function*). In §2.2, we will discuss some common acquisition functions.

In practice, there is also the possibility of measurement noise, which we will assume is Gaussian. We define $\mathbf{x}_i$ as the ith sample and $y_i = f(\mathbf{x}_i) + \varepsilon_i$, with $\varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_{\text{noise}})$, as the noisy observation of the objective function at $\mathbf{x}_i$. We will discuss the noise model in §2.3. We focus here on real, Gaussian observations for ease of presentation, though this will be extended to discrete observations in Chapter 3.

The Bayesian optimization procedure is shown in Algorithm 1. As mentioned earlier, it has two components: the posterior distribution over the objective and the acquisition function. Let us focus on the posterior distribution first and come back to the acquisition function in §2.2. As we accumulate observations $\mathcal{D}_{1:t} = \{\mathbf{x}_{1:t}, y_{1:t}\}$, a prior distribution $P(f)$ is combined with the likelihood function $P(\mathcal{D}_{1:t}|f)$ to produce the posterior distribution: $P(f|\mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t}|f)P(f)$. The posterior captures the updated beliefs about the unknown objective function.

Algorithm 1 Bayesian Optimization

1: **for** $t = 1, 2, \ldots$ **do**
2: Find $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \operatorname{argmax}_{\mathbf{x}} u(\mathbf{x}|\mathcal{D}_{1:t-1})$.
3: Sample the objective function: $y_t = f(\mathbf{x}_t) + \varepsilon_t$.
4: Augment the data $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.
5: **end for**

2.1.1 Priors over functions

Any Bayesian method depends on a prior distribution, by definition. A Bayesian optimization method will converge to the optimum if (i) the acquisition function is continuous and approximately minimizes the risk (defined as the expected deviation from the global minimum at a fixed point $\mathbf{x}$); and (ii) conditional variance converges to zero (or appropriate positive minimum value in the presence of noise) if and only if the distance to the nearest observation is zero [Močkus, 1982; 1994]. Many models could be used for this prior—early work mostly used the Wiener process (§2.4). However, Gaussian process (GP) priors for Bayesian optimization date back at least to the late 1970s [O'Hagan, 1978; Žilinskas, 1980]. Močkus [1994] explicitly set the framework for the Gaussian process prior by specifying the additional "simple and natural" conditions that (iii) the objective is continuous; (iv) the prior is homogeneous; (v) the optimization is independent of the m^{th} differences. This includes a very large family of common optimization tasks, and Močkus showed that the GP prior is well-suited to the task.

A GP is a stochastic process for which any finite combination of samples will be normally distributed. However, for our purposes, it is often useful to intuitively think of a GP as analogous to a function, but instead of returning a scalar $f(\mathbf{x})$ for an arbitrary $\mathbf{x}$, it returns the mean and variance of a normal distribution (Figure 2.1) over the possible values of f at $\mathbf{x}$. Stochastic processes are sometimes called "random functions", by analogy to random variables.

Just as a Gaussian distribution is a distribution over a random variable, completely specified by its mean and covariance, a GP is a distribution over functions, completely specified by its mean function, m and covariance

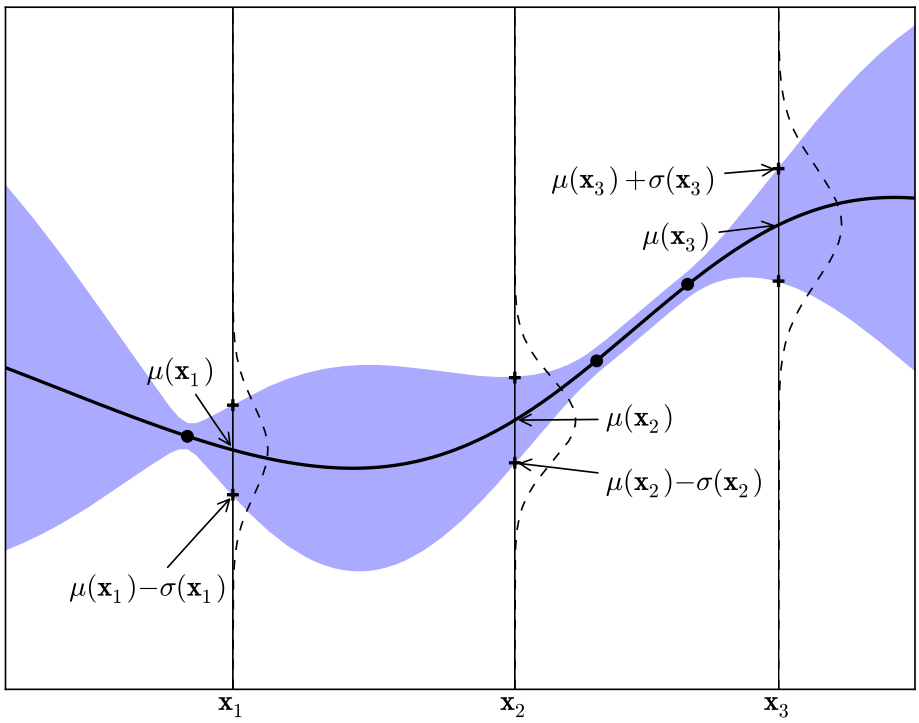


Figure 2.1: *Simple 1D Gaussian process with three observations. The solid black line is the GP surrogate mean prediction of the objective function given the data, and the shaded area shows the mean plus and minus the variance. The superimposed Gaussians correspond to the GP mean and standard deviation ($\mu(\cdot)$ and $\sigma(\cdot)$) of prediction at the points, $\mathbf{x}_{1:3}$.*

function, k:

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot)).$$

For convenience, we assume here that the prior mean is the zero function and the covariance is appropriately scaled; in §4.1 we present alternative priors for the mean and covariance. This leaves us the more interesting question of defining the covariance function k. A very popular choice is the

squared exponential function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right). \tag{2.1}$$

Note that this function approaches 1 as values get close together and 0 as they get further apart. Two points that are close together can be expected to have a very large influence on each other, whereas distant points have almost none. This is a necessary condition for convergence under the assumptions of [Močkus, 1994]. We will discuss more sophisticated kernels in §2.1.2.

If we were to sample from the prior, we would choose $\{\mathbf{x}_{1:t}\}$ and sample the values of the function at these indices to produce the pairs $\{\mathbf{x}_{1:t}, \mathbf{f}_{1:t}\}$, where $\mathbf{f}_{1:t} = f(\mathbf{x}_{1:t})$. The function values are drawn according to a multivariate normal distribution $\mathcal{N}(0, \mathbf{K})$, where the kernel matrix is given by:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \dots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}.$$

Of course, the diagonal values of this matrix are 1, which is only possible in a noise-free environment. We will discuss noise in §2.3. Also, recall that we have for simplicity chosen the zero mean function.

In our optimization tasks, however, we will use data from an external model to fit the GP and get the posterior. Assume that we already have the observations $\{\mathbf{x}_{1:t}, \mathbf{f}_{1:t}\}$, say from previous iterations, and that we want to use Bayesian optimization to decide what point $\mathbf{x}_{t+1}$ should be considered next. Let us denote the value of the function at this arbitrary point as $f_{t+1} = f(\mathbf{x}_{t+1})$. Then, by the properties of Gaussian processes, $\mathbf{f}_{1:t}$ and f_{t+1} are jointly Gaussian:

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \end{bmatrix}\right),$$

where

$$\mathbf{k} = \begin{bmatrix} k(\mathbf{x}_{t+1}, \mathbf{x}_1) & k(\mathbf{x}_{t+1}, \mathbf{x}_2) & \cdots & k(\mathbf{x}_{t+1}, \mathbf{x}_t) \end{bmatrix}$$

Using the Sherman-Morrison-Woodbury formula (see, e.g., [Rasmussen and Williams, 2006; Press *et al.*, 2007]), one can easily arrive at an expression for the predictive distribution:

$$P(f_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1}) = \mathcal{N}\left(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1})\right)$$

where

$$\begin{aligned} \mu_t(\mathbf{x}_{t+1}) &= \mathbf{k}^T \mathbf{K}^{-1} \mathbf{f}_{1:t} \\ \sigma_t^2(\mathbf{x}_{t+1}) &= k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k}. \end{aligned}$$

That is, $\mu_t(\mathbf{x}_{t+1})$ and $\sigma_t^2(\mathbf{x}_{t+1})$ model the predictive posterior distribution $P(f_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1})$. For legibility, we will omit the subscripts on μ and σ except where it might be unclear. In our sequential decision making setting, the number of query points is relatively small and, consequently, the GP predictions are easy to compute. In other problems, the number of data may be quite large, in which case some form of adaptive 'dropping' of data is required (see e.g. [Osborne *et al.*, 2010]).

Note that the idea of using a GP approximation to compute attributes of an expensive function is not unique to Bayesian optimization. For example, in the Hybrid Monte Carlo literature, the *GPHMC* algorithm [Rasmussen, 2003] uses a GP for most of the HMC computations, sampling the actual posterior only to ensure the approximation is reasonable.

2.1.2 Choice of covariance functions

The choice of covariance function for the Gaussian process is crucial, as it determines the smoothness properties of samples drawn from it. The squared exponential kernel in Eqn. (2.1) is actually a little naive, in that divergences of all features of $\mathbf{x}$ affect the covariance equally.

Typically, it is necessary to generalize by adding *hyperparameters*. In an isotropic model, this can be done with a single hyperparameter θ, which

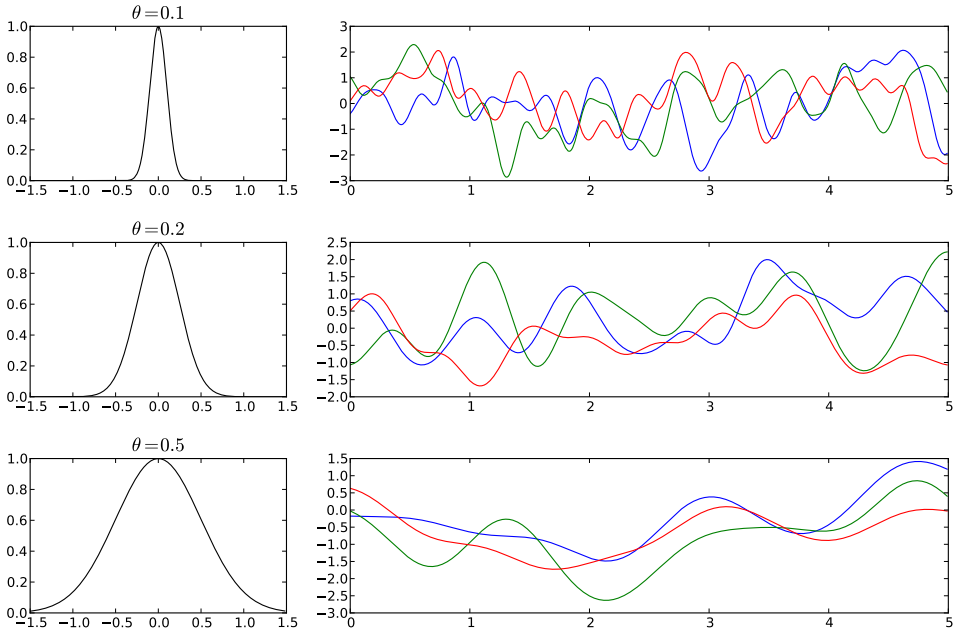


Figure 2.2: *The effect of changing the kernel hyperparameters. Shown are squared exponential kernels with $\theta = 0.1, 0.2, 0.5$. On the left is the function $k(0, \mathbf{x})$. On the right are some one-dimensional functions sampled from a GP with the hyperparameter value.*

controls the width of the kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\theta^2}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right).$$

For anisotropic models, a very popular choice is the squared exponential kernel with a vector of automatic relevance determination (ARD) hyperparameters $\boldsymbol{\theta}$ Neal:1996,Williams:1996:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\tfrac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \operatorname{diag}(\boldsymbol{\theta})^{-2}(\mathbf{x}_i - \mathbf{x}_j)\big),$$

where $\operatorname{diag}(\boldsymbol{\theta})$ is a diagonal matrix with d entries $\boldsymbol{\theta}$ along the diagonal. Intuitively, if a particular θ_ℓ has a small value, the kernel becomes independent of the ℓ^{th} input, effectively removing irrelevant dimensions. Figure 2.2

shows examples of different hyperparameter values on the squared exponential function and what functions sampled from those values look like. In §4.1.1, we will discuss how the hyperparameters can be learned.

Another important kernel for Bayesian optimization is the Matérn kernel [Matérn, 1960; Stein, 1999], which incorporates a smoothness parameter ς to permit greater flexibility in modelling functions:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2^{\varsigma-1}\Gamma(\varsigma)} \left(2\sqrt{\varsigma}\,\|\mathbf{x}_i - \mathbf{x}_j\|\right)^{\varsigma} H_{\varsigma}\left(2\sqrt{\varsigma}\,\|\mathbf{x}_i - \mathbf{x}_j\|\right),$$

where $\Gamma(\cdot)$ and $H_{\varsigma}(\cdot)$ are the Gamma function and the Bessel function of order ς. Note that as $\varsigma \to \infty$, the Matérn kernel reduces to the squared exponential kernel, and when $\varsigma = 0.5$, it reduces to the unsquared exponential kernel. As with the squared exponential, length-scale hyperparameter are often incorporated.

While the squared exponential and Matérn are the most common kernels for GPs, numerous others have been examined in the machine learning literature (see, e.g., [Genton, 2001] or [Rasmussen and Williams, 2006, Chapter 4] for an overview). Appropriate covariance functions can also be used to extend the model in other interesting ways. For example, the recent sequential sensor work of Osborne, Garnett and colleagues uses GP models with extensions to the covariance function to model the characteristics of changepoints [Osborne *et al.*, 2010] and the locations of sensors in a network [Garnett *et al.*, 2010a]. A common additional hyperparameter is simply a scalar applied to k to control the magnitude of the variance.

Determining which of a set of possible kernel functions to use for a problem typically requires a combination of engineering and automatic model selection, either hierarchical Bayesian model selection [Mackay, 1992] or cross-validation. However, these methods require fitting a model given a representative sample of data. In Chapter 4, we discuss how model selection can be performed using models believed to be similar. The techniques introduced in Chapter 5 could also be applied to model selection, though that is outside the scope of this thesis.

2.2 Acquisition functions for Bayesian optimization

Now that we have discussed placing priors over smooth functions and how to update these priors in light of new observations, we will focus our attention on the acquisition component of Bayesian optimization. The role of the acquisition function is to guide the search for the optimum. Typically, acquisition functions are defined such that high acquisition corresponds to *potentially* high values of the objective function, whether because the prediction is high, the uncertainty is great, or both. Maximizing the acquisition function is used to select the next point at which to evaluate the function. That is, we wish to sample f at $\operatorname{argmax}_{\mathbf{x}} u(\mathbf{x}|\mathcal{D})$, where $u(\cdot)$ is the generic symbol for an acquisition function.

The choice of acquisition function is nontrivial. In this chapter, we will discuss several different methods which been proposed in the literature, and show how they give rise to distinct sampling behaviour. Each works well for certain classes of functions, and it is often difficult or impossible to know which will perform best on an unknown function. In Chapter 5 we propose a solution to this problem, based on a hierarchical hedging approach for managing an adaptive portfolio of acquisition functions.

2.2.1 Improvement-based acquisition functions

The early work of Kushner [1964] suggested maximizing the *probability of improvement* over the incumbent $f(\mathbf{x}^+)$, where $\mathbf{x}^+ = \operatorname{argmax}_{\mathbf{x}_i \in \mathbf{x}_{1:t}} f(\mathbf{x}_i)$, so that

$$\begin{aligned} \mathrm{PI}(\mathbf{x}) &= P(f(\mathbf{x}) \geq f(\mathbf{x}^+)) \\ &= \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}\right), \end{aligned}$$

where $\Phi(\cdot)$ is the normal cumulative distribution function. This function is also sometimes called *MPI* (for "maximum probability of improvement") or "the P-algorithm" (since the utility is the probability of improvement).

The drawback, intuitively, is that this formulation is pure exploitation.

Points that have a high probability of being infinitesimally greater than $f(\mathbf{x}^+)$ will be drawn over points that offer potentially larger gains but less certainty. As a result, a modification is to add a trade-off parameter $\xi \geq 0$:

$$\begin{aligned} \mathrm{PI}(\mathbf{x}) &= P(f(\mathbf{x}) \geq f(\mathbf{x}^+) + \xi) \\ &= \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})}\right), \end{aligned} \tag{2.2}$$

The exact choice of ξ is left to the user, though Kushner recommended a schedule for ξ, so that it started fairly high early in the optimization, to drive exploration, and decreased toward zero as the algorithm continued. Several researchers have studied the empirical impact of different values of ξ in different domains [Törn and Žilinskas, 1989; Jones, 2001; Lizotte, 2008].

An appealing characteristic of this formulation for perceptual and preference models is that while maximizing $\mathrm{PI}(\cdot)$ is still greedy, it selects the point most likely to offer an improvement of at least ξ. This can be useful in psychoperceptual tasks, where there is a threshold of perceptual difference.

Jones [2001] notes that the performance of $\mathrm{PI}(\cdot)$

> "is truly impressive. It would be quite natural if the reader, like so many others, became enthusiastic about this approach. But if there is a single lesson to be taken away from this paper, it is that nothing in this response-surface area is so simple. There always seems to be a counterexample. In this case, the difficulty is that [the $\mathrm{PI}(\cdot)$ method] is extremely sensitive to the choice of the target. If the desired improvement is too small, the search will be highly local and will only move on to search globally after searching nearly exhaustively around the current best point. On the other hand, if [ξ] is set too high, the search will be excessively global, and the algorithm will be slow to fine-tune any promising solutions."

A somewhat more satisfying alternative acquisition function would be one that takes into account not only the probability of improvement, but the magnitude of the improvement a point can potentially yield. In particular,

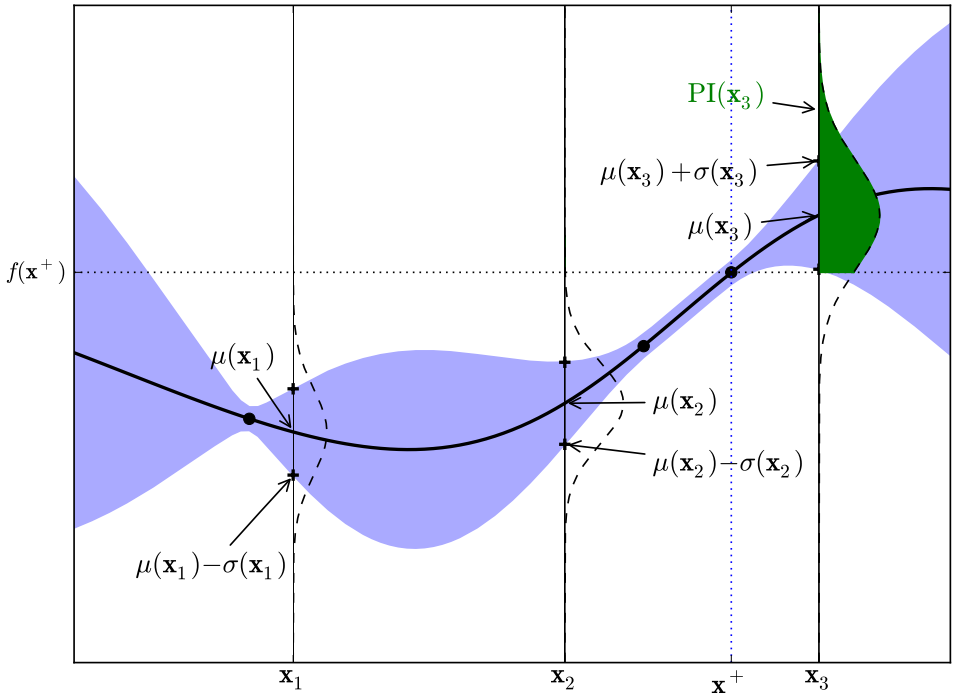


Figure 2.3: *Gaussian process from Figure 2.1, additionally showing the region of probable improvement. The maximum observation is at* $\mathbf{x}^+$*. The darkly-shaded area in the superimposed Gaussian above the dashed line can be used as a measure of improvement,* $I(\mathbf{x})$*. The model predicts almost no possibility of improvement by observing at* $\mathbf{x}_1$ *or* $\mathbf{x}_2$*, while sampling at* $\mathbf{x}_3$ *is more likely to improve on* $f(\mathbf{x}^+)$*.*

we want to minimize the expected deviation from the true maximum $f(\mathbf{x}^\star)$, when choosing a new trial point:

$$\begin{aligned}\mathbf{x}_{t+1} &= \operatorname*{argmin}_{\mathbf{x}} \mathbb{E}(\|f_{t+1}(\mathbf{x}) - f(\mathbf{x}^\star)\| \,|\mathcal{D}_{1:t}) \\ &= \operatorname*{argmin}_{\mathbf{x}} \int \|f_{t+1}(\mathbf{x}) - f(\mathbf{x}^\star)\| P(f_{t+1}|\mathcal{D}_{1:t}) df_{t+1},\end{aligned}$$

Note that this decision process is myopic in that it only considers one-step-ahead choices. However, if we want to plan two steps ahead, we can easily

apply recursion:

$$\mathbf{x}_{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbb{E}\left(\min_{\mathbf{x}'} \mathbb{E}(\| f_{t+2}(\mathbf{x}') - f(\mathbf{x}^\star)\| \mid \mathcal{D}_{t+1}) \mid \mathcal{D}_{1:t} \right)$$

One could continue applying this procedure of dynamic programming for as many steps ahead as desired. However, because of its expense, Močkus *et al.* [1978] proposed the alternative of maximizing the expected improvement with respect to $f(\mathbf{x}^+)$. Specifically, Močkus defined the improvement function as:

$$\mathrm{I}(\mathbf{x}) = \max\{0, f_{t+1}(\mathbf{x}) - f(\mathbf{x}^+)\}.$$

That is, $\mathrm{I}(\mathbf{x})$ is positive when the prediction is higher than the best value known thus far. Otherwise, $\mathrm{I}(\mathbf{x})$ is set to zero. The new query point is found by maximizing the expected improvement:

$$\mathbf{x} = \underset{\mathbf{x}}{\operatorname{argmax}}\, \mathbb{E}(\max\{0, f_{t+1}(\mathbf{x}) - f(\mathbf{x}^+)\} \mid \mathcal{D}_t)$$

The likelihood of improvement I on a normal posterior distribution characterized by $\mu(\mathbf{x}), \sigma^2(\mathbf{x})$ can be computed from the normal density function,

$$\frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left(-\frac{(\mu(\mathbf{x}) - f(\mathbf{x}^+) - \mathrm{I})^2}{2\sigma^2(\mathbf{x})} \right).$$

The expected improvement is the integral over this function:

$$\begin{aligned} \mathbb{E}(\mathrm{I}) &= \int_{\mathrm{I}=0}^{\mathrm{I}=\infty} \mathrm{I}\, \frac{1}{\sqrt{2\pi}\sigma(\mathbf{x})} \exp\left(-\frac{(\mu(\mathbf{x}) - f(\mathbf{x}^+) - \mathrm{I})^2}{2\sigma^2(\mathbf{x})} \right) d\,\mathrm{I} \\ &= \sigma(\mathbf{x}) \left[\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})} \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})} \right) + \phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})} \right) \right] \end{aligned}$$

The expected improvement can be evaluated analytically [Močkus *et al.*,

1978; Jones *et al.*, 1998], yielding:

$$\begin{aligned} \mathrm{EI}(\mathbf{x}) &= \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \\ Z &= \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})} \end{aligned} \tag{2.3}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the PDF and CDF of the standard normal distribution respectively. Figure 2.3 illustrates a typical expected improvement scenario.

Exploration-exploitation trade-off

The expectation of the improvement function with respect to the predictive distribution of the Gaussian process enables us to balance the trade-off of exploiting and exploring. When exploring, we should choose points where the surrogate variance is large. When exploiting, we should choose points where the surrogate mean is high.

It is highly desirable for our purposes to express $\mathrm{EI}(\cdot)$ in a generalized form which controls the trade-off between global search and local optimization (exploration/exploitation). Schonlau [1997] suggests adding a non-negative integer parameter γ, such that

$$I(\mathbf{x}_{t+1}) = \max\{0, (\mu(\mathbf{x}_{t+1}) - f(\mathbf{x}^+))^\gamma\}$$

which results in an expected improvement of

$$\mathrm{EI}_\gamma(\mathbf{x}) = \sigma^\gamma(\mathbf{x}) \sum_{j=0}^{\gamma} (-1)^j \binom{\gamma}{j} Z^{g-j} T_j,$$

where

$$\begin{aligned} T_0 &= \Phi(Z) \\ T_1 &= -\phi(Z) \\ T_{j>1} &= -Z^{j-1}\phi(Z) + (j-1)T_{j-2} \end{aligned}$$

When $\gamma = 1$, (which degenerates to Eqn. (2.3)), emphasis is placed on trying to improve near $\mathbf{x}^+$, unless the observations strongly suggest improvement in areas of high variance. As γ is increased, areas of larger, but less-likely, increase will be favoured. While there is no obvious way to select γ for an arbitrary function, Sasena [2002] proposes an annealing-type schedule to allow global search to smoothly collapse to local improvement.

Lizotte [2008] suggests an alternative $\xi \geq 0$ parameter such that:

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}, \quad (2.4)$$

where

$$Z = \begin{cases} \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}.$$

This ξ is very similar in flavour to the ξ used in Eqn. (2.2). Lizotte's experiments suggest that setting $\xi = 0.01$ (scaled by the signal variance if necessary) works well in almost all cases, and interestingly, setting a cooling schedule for ξ to encourage exploration early and exploitation later does *not* work well empirically, contrary to intuition (though Lizotte did find that a cooling schedule for ξ might slightly improve performance on short runs ($t < 30$) of PI optimization).

2.2.2 Confidence bound criteria

Cox and John [1992; 1997] introduce an algorithm they call "Sequential Design for Optimization", or *SDO*. Given a random function model, SDO

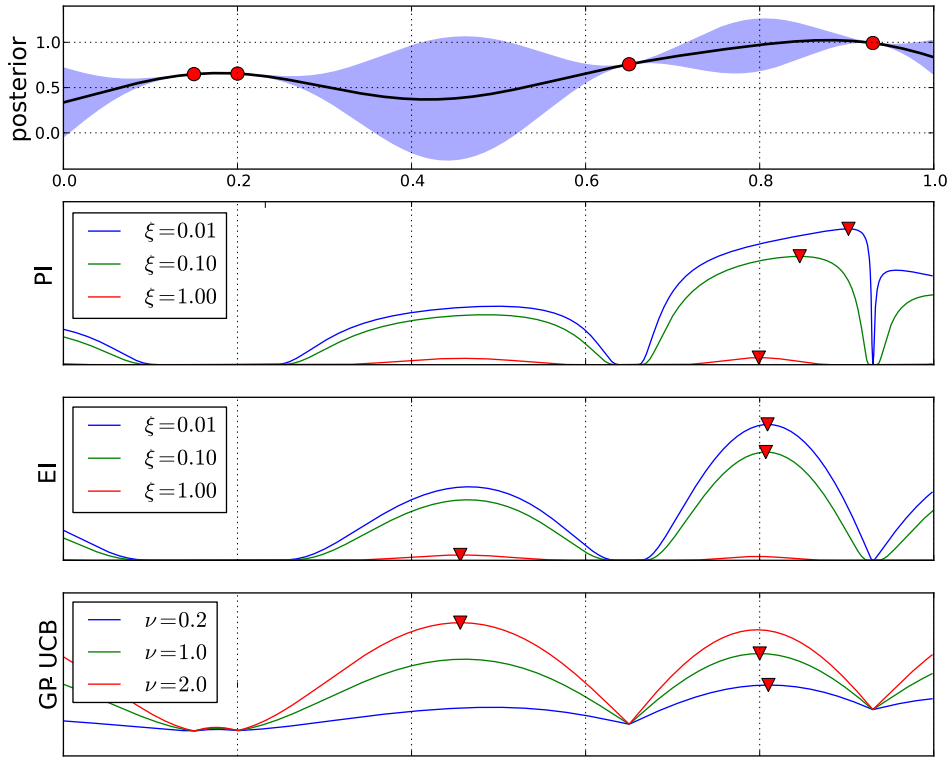


Figure 2.4: *Examples of acquisition functions and their settings. The GP posterior is shown at top. The other images show the acquisition functions for that GP. From the top: probability of improvement (Eqn. (2.2)), expected improvement (Eqn. (2.4)) and upper confidence bound (Eqn. (2.5)). The maximum of each function is shown with a triangle marker.*

selects points for evaluation based on the *lower confidence bound* of the prediction site:

$$\mathrm{LCB}(\mathbf{x}) = \mu(\mathbf{x}) - \kappa\sigma(x),$$

where $\kappa \geq 0$. While they are concerned with minimization, we can maximize by instead defining the upper confidence bound:

$$\mathrm{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(x).$$

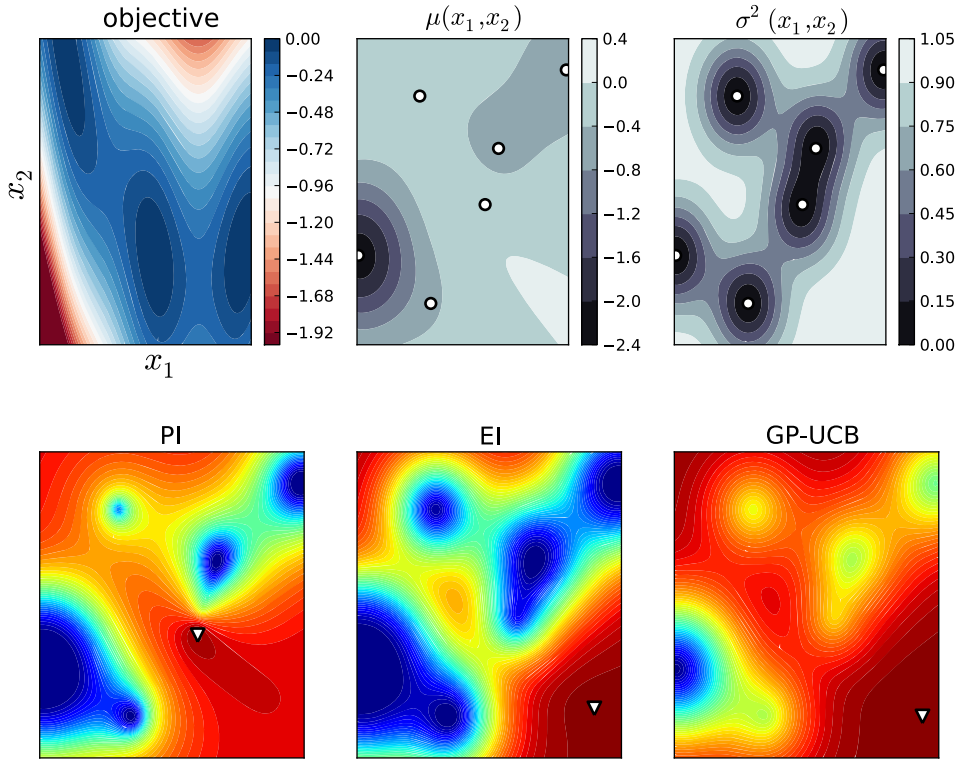


Figure 2.5: *Examples of acquisition functions and their settings in 2 dimensions. The top row shows the objective function (which is the scaled Branin function here), and the posterior mean and variance estimates $\mu(\cdot)$ and $\sigma^2(\cdot)$. The samples used to train the GP are shows with white dots. The second row shows the acquisition functions for the GP. From left to right: probability of improvement (Eqn. (2.2)), expected improvement (Eqn. (2.4)) and upper confidence bound (Eqn. (2.5)). The maximum of each function is shown with a triangle marker.*

Like other parameterized acquisition models we have seen, the parameter κ is left to the user. However, an alternative acquisition function has been proposed by Srinivas *et al.* [2010]. Casting the Bayesian optimization problem as a multi-armed bandit, the acquisition is the instantaneous regret function

$$r(\mathbf{x}) = f(\mathbf{x}^\star) - f(\mathbf{x}).$$

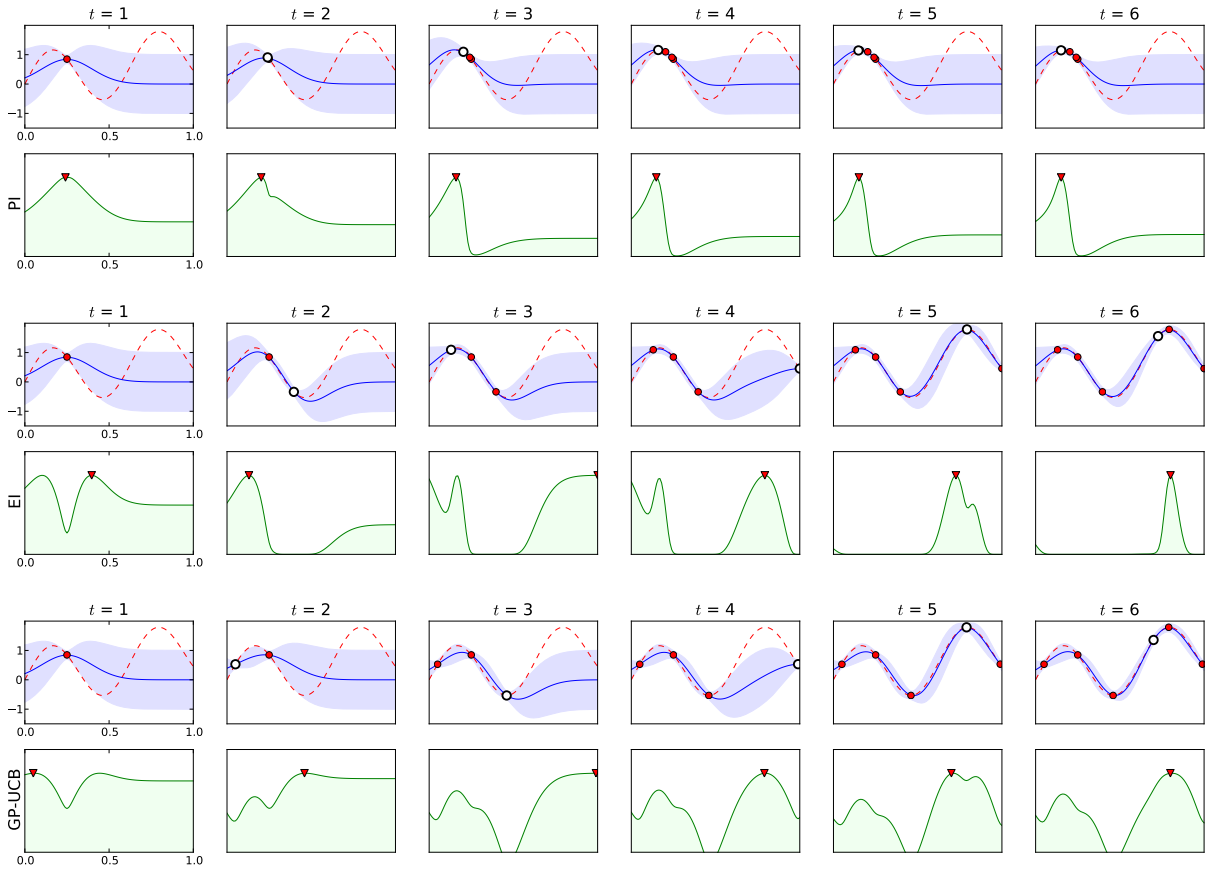


Figure 2.6: *Comparison of probability of improvement (top), expected improvement (middle) and upper confidence bound (bottom) acquisition functions on a toy 1D problem. In the upper rows, the objective function is shown with the dotted red line, the solid blue line is the GP posterior mean. In the lower rows, the respective acquisition functions are shown, with a star denoting the maximum. The optimizations are initialized with the same two points, but quickly follow different sampling trajectories. In particular, note that the conservative* PI *algorithm ignores the region outside the local maximum once it is determined there is minimal chance of improvement, while* EI *and* GP-UCB *continue to explore. This is a known problem with the* PI *algorithm.*

The goal of optimizing in the framework is to find:

$$\min \sum_{t}^{T} r(\mathbf{x}_t) = \max \sum_{t}^{T} f(\mathbf{x}_t),$$

where T is the number of iterations the optimization is to be run for.

Using the *upper confidence bound* selection criterion with $\kappa_t = \sqrt{\nu\tau_t}$ and the hyperparameter $\nu > 0$ Srinivas *et al.* define

$$\text{GP-UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\nu\tau_t}\sigma(\mathbf{x}). \tag{2.5}$$

It can be shown that this method has a cumulative regret which is bounded by $O(\sqrt{T\beta_T\gamma_T})$ with high probability. Here γ_T and β_T are problem-specific, depending upon the particular form of kernel-function used, but for many kernels their product can be shown to be sublinear in T. We refer the interested reader to the original paper [Srinivas *et al.*, 2010] for exact bounds.

The sublinear bound on cumulative regret implies that the method is *no-regret*, i.e., that $\lim_{T\to\infty} R_T/T = 0$. This in turn provides a bound on the convergence rate for the optimization process, since the regret at the maximum $f(\mathbf{x}^*) - \max_t f(\mathbf{x}_t)$ is upper bounded by the average regret

$$\frac{R_T}{T} = f(\mathbf{x}^*) - \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}_t).$$

Figures 2.4 and 2.5 show how with the same GP posterior, different acquisition functions with different maxima are defined. Figure 2.6 shows how PI, EI and GP-UCB give rise to distinct sampling behaviour over time.

2.2.3 Maximizing the acquisition function

To find the point at which to sample, we still need to maximize the constrained objective $u(\mathbf{x})$. *Unlike the original unknown objective function, $u(\cdot)$ can be cheaply sampled.* We optimize the acquisition function using *DIRECT* [Jones *et al.*, 1993], a deterministic, derivative-free optimizer. It uses the existing samples of the objective function to decide how to proceed to DIvide

the feasible space into finer RECTangles. A particular advantage in active learning applications is that DIRECT can be implemented as an "any-time" algorithm, so that as long as the user is doing something else, it continues to optimize, and when interrupted, the program can use the best results found to that point in time. Methods such as Monte Carlo and multistart have also been used, and seem to perform reasonably well [Močkus, 1994; Lizotte, 2008].

2.3 Noise

The model we've used so far assumes that we have perfectly noise-free observations. In real life, this is rarely possible, and instead of observing $f(\mathbf{x})$, we can often only observe a noisy transformation of $f(\mathbf{x})$.

The simplest transformation arises when $f(\mathbf{x})$ is corrupted with Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$ [Rasmussen and Williams, 2006]. If the noise is additive, we can easily add the noise distribution to the Gaussian distribution $\mathcal{N}(0, \mathbf{K})$ and define

$$y_i = f(\mathbf{x}_i) + \epsilon_i.$$

Since the mean is zero, this type of noise simply requires that we replace $\mathbf{K}$ with $\mathbf{K} + \sigma^2_{\text{noise}} I$.

This yields the predictive distribution:

$$P(y_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1}) = \mathcal{N}(\mu_t(\mathbf{x}_{t+1}), \sigma^2_t(\mathbf{x}_{t+1}) + \sigma^2_{\text{noise}}),$$

and the sufficient statistics

$$\begin{aligned}
\mu_t(\mathbf{x}_{t+1}) &= \mathbf{k}^T [\mathbf{K} + \sigma^2_{\text{noise}} I]^{-1} \mathbf{y}_{1:t} \\
\sigma^2_t(\mathbf{x}_{t+1}) &= k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T [\mathbf{K} + \sigma^2_{\text{noise}} I]^{-1} \mathbf{k}.
\end{aligned}$$

In a noisy environment, we also change the definition of the incumbent in the PI and EI acquisition functions. Instead of using the best observation, we use the distribution at the sample points, and define as the incumbent,

the point with the highest expected value,

$$\mu^+ = \underset{\mathbf{x}_i \in \mathbf{x}_{1:t}}{\operatorname{argmax}} \mu(\mathbf{x}_i).$$

This avoids the problem of attempting to maximize probability or expected improvement over an unreliable sample. It is also possible to resample potential incumbents to get more reliable estimates of the values in a noisy environment [Bartz-Beielstein *et al.*, 2005; Huang *et al.*, 2006; Hutter *et al.*, 2009], a process sometimes called *intensification*. Nonstationary noise models are also possible, such as autoregressive moving-average noise [Murray-Smith and Girard, 2001] and heteroskedastic Gaussian noise [Goldberg *et al.*, 1998].

2.4 A brief history of Bayesian optimization

The earliest work we are aware of resembling the modern Bayesian optimization approach is the early work of Kushner [1964], who used Wiener processes for unconstrained one-dimensional problems. Kushner's decision model was based on maximizing the probability of improvement (§2.2.1). He also included a parameter that controlled the trade-off between 'more global' and 'more local' optimization, in the same spirit as the exploration-exploitation trade-off. A key difference is that in a (one-dimensional) Wiener process, the intervals between samples are independent, and Kushner was concerned with the problem of selecting from a finite set of intervals. Later work extended Kushner's technique to multidimensional optimization, using, for example, interpolation in a Delauney triangulation of the space [Elder, 1992] or projecting Wiener processes between sample points [Stuckman, 1988].

Meanwhile, in the former Soviet Union, Močkus and colleagues developed a multidimensional Bayesian optimization method using linear combinations of Wiener fields. This was first published in English as [Močkus *et al.*, 1978]. This paper also, significantly, describes an acquisition function that is based on myopic expected improvement of the posterior, which has been widely

adopted in Bayesian optimization as the expected improvement function (§2.2.1). A more recent review of Močkus' approach is [Močkus, 1994].

At the same time, a large, related body of work emerged under the name *kriging* (§2.4.1), in honour of the South African student who developed this technique at the University of the Witwatersrand [Krige, 1951], though largely popularized by Matheron and colleagues (e.g. [Matheron, 1971]). In kriging, the goal is interpolation of a random field via a linear predictor. The errors on this model are typically assumed to *not* be independent, and are modelled with a Gaussian process.

More recently, Bayesian optimization using Gaussian processes has been successfully applied to derivative-free optimization and experimental design, where it is called Efficient Global Optimization, or *EGO* (§2.4.2).

There exist several consistency proofs for this algorithm in the one-dimensional setting [Locatelli, 1997] and one for a simplification of the algorithm using simplicial partitioning in higher dimensions [Žilinskas and Žilinskas, 2002]. The convergence of the algorithm using multivariate Gaussian processes has been recently established in [Vasquez and Bect, 2008].

2.4.1 Kriging

Kriging has been used in geostatistics and environmental science since the 1950s and remains important today. We will briefly summarize the connection to Bayesian optimization here. More detailed examinations can be found in, for example, [Stein, 1999; Sasena, 2002; Diggle and Ribeiro, 2007]. This section is primarily drawn from these sources.

In many modelling techniques in statistics and machine learning, it is assumed that samples drawn from a process with independent, identically distributed residuals, typically, $\varepsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$:

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon$$

In kriging, however, the usual assumption is that errors are *not* independent, and are, in fact, spatially correlated: where errors are high, it is expected that nearby errors will also be high. Kriging is a combination of a

linear regression model and a stochastic model fitted to the residual errors of the linear model. The residual is modelled with a zero-mean Gaussian process, so ε is actually parameterized by $\mathbf{x}$: $\varepsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma^2(\mathbf{x}))$.

The actual regression model depends on the type of kriging. In *simple kriging*, f is modelled with the zero function, making it a zero-mean GP model. In *ordinary kriging*, f is modelled with a constant but unknown function. *Universal kriging* models f with a polynomial of degree k with bases m and coefficients β, so that

$$y(\mathbf{x}) = \sum_{j=1}^{k} \beta_j m_j(\mathbf{x}) + \varepsilon(\mathbf{x}).$$

Other, more exotic types of kriging are also used.

Clearly, kriging and Bayesian optimization are very closely related. There are some key differences in practice, though. In Bayesian optimization, models are usually fit through maximum likelihood. In kriging, models are usually fit using a *variogram*, a measure of the average dissimilarity between samples versus their separation distance. Fitting is done using least squares or similar numerical methods, or interactively, by an expert visually inspecting the variogram plot with specially-designed software. Kriging also often restricts the prediction model to use only a small number of neighbours, making it fit locally while ignoring global information. Bayesian optimization normally uses all the data in order to learn a global model.

2.4.2 Experimental design

Kriging has been applied to experimental design under the name *DACE*, after "Design and Analysis of Computer Experiments", the title of a paper by Sacks *et al.* [1989] (and more recently a book by Santner *et al.* [2003]). In DACE, the regression model is a best linear unbiased predictor (BLUP), and the residual model is a noise-free Gaussian process. The goal is to find a design point or points that optimizes some criterion.

The "efficient global optimization", or *EGO*, algorithm is the combination of DACE model with the sequential expected improvement (§2.2.1) ac-

quisition criterion. It was published in a paper by Jones *et al.* [1998] as a refinement of the *SPACE* algorithm (Stochastic Process Analysis of Computer Experiments) [Schonlau, 1997]. Since EGO's publication, there has evolved a body of work devoted to extending the algorithm, particularly in adding constraints to the optimization problem [Audet *et al.*, 2000; Sasena, 2002; Boyle, 2007], and in modelling noisy functions [Bartz-Beielstein *et al.*, 2005; Huang *et al.*, 2006; Hutter *et al.*, 2009; Hutter, 2009].

In so-called "classical" experimental design, the problem to be addressed is often to learn the parameters $\boldsymbol{\zeta}$ of a function $g_{\boldsymbol{\zeta}}$ such that

$$y_i = g_{\boldsymbol{\zeta}}(\mathbf{x}_i) + \varepsilon_i, \forall i \in 1, \ldots, t$$

with noise ε_i (usually Gaussian) for scalar output y_i. $\mathbf{x}_i$ is the i^{th} set of experimental conditions. Usually, the assumption is that $g_{\boldsymbol{\zeta}}$ is linear, so that

$$y_i = \boldsymbol{\zeta}^T \mathbf{x}_i + \varepsilon_i.$$

An experiment is represented by a design matrix $\mathbf{X}$, whose rows are the independent inputs $\mathbf{x}_{1:t}$. If we let $\varepsilon \sim \mathcal{N}(0, \sigma)$, then for the linear model, the variance of the parameter estimate $\widehat{\boldsymbol{\zeta}}$ is

$$Var(\widehat{\boldsymbol{\zeta}}) = \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1},$$

and for an input $\mathbf{x}_i$, the prediction is

$$Var(\widehat{y}_t) = \sigma^2 \mathbf{x}_i^T (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{x}_i.$$

An optimal design is a design matrix that minimizes some characteristic of the inverse moment matrix $(\mathbf{X}^T \mathbf{X})^{-1}$. Common criteria include A-optimality, which minimizes the trace; D-optimality, which minimizes the determinant; and E-optimality, which minimizes the maximum eigenvalue.

Experimental design is usually non-adaptive: the entire experiment is designed before data is collected. However, *sequential design* is an important and active subfield (e.g. [Williams *et al.*, 2000; Busby, 2009]).

2.4.3 Active learning

Active learning is another area related to Bayesian optimization, and of particular relevance to our task. Active learning is closely related to experimental design and, indeed, the decision to describe a particular problem as active learning or experimental design is often arbitrary. However, there are a few distinguishing characteristics that most (but by no means all) active learning approaches use.

- Active learning is most often *adaptive*: a model exists that incorporates all the data seen, and this is used to sequentially select candidates for labelling. Once labelled, the data are incorporated into the model and new candidates selected. This is, of course, the same strategy used by Bayesian optimization.

- Active learning employs an oracle for data labelling, and this oracle is very often a human being, as is the case with interactive Bayesian optimization.

- Active learning is usually concerned with selecting candidates from a finite set of available data (*pool-based* sampling). Experimental design and optimization are usually concerned with continuous domains.

- Finally, active learning is usually used to learn a model for classification, or, less commonly, regression. Usually the candidate selection criterion is the maximization of some informativeness measure, or the minimization of uncertainty. These criteria are often closely related to the alphabetic criteria of experimental design.

An excellent recent overview of active learning is [Settles, 2010]. We are particularly interested in active learning because it often uses a human oracle for label acquisition. An example using GPs is the object categorization of Kapoor *et al.* [2007]. In this work, candidates are selected from a pool of unlabelled images so as to maximize the margin of the GP classifier and minimize the uncertainty. Osborne *et al.* [2010] also use active learning in a Gaussian process problem, deciding when to sample variables of interest

in a sequential sampling problem with faults and changepoints. Chu and Ghahramani [2005a] briefly discuss how active learning could be used for *ranking* GPs, by selecting sample points that maximize entropy gained with a new preference relation.

Interesting recent work with GPs that straddles the boundary between active learning and experimental design is the sensor placement problem of Krause *et al.* [2008]. They examine several criteria, including maximum entropy, and argue for using mutual information. Ideally, they would like to simultaneously select a set of points to place the entire set of sensors in a way that maximizes the mutual information. This is essentially a classical experimental design problem with maximum mutual information as the design criterion. However, maximizing mutual information over a set of samples is NP-complete, so they use an active learning approach. By exploiting the submodularity of mutual information, they are able to show that sequentially selecting sensor locations that greedily maximize mutual information, they can bound the divergence of the active learning approach from the experimental design approach. This work influenced the GP-UCB acquisition function (§2.2.2).

Finally, as an aside, in active learning, a common acquisition strategy is selecting the point of maximum uncertainty. This is called *uncertainty sampling* [Lewis and Gale, 1994]. GPs have the useful property that the posterior variance (interpreted as uncertainty) is independent of the actual observations! As a result, if this is the criterion, the entire active learning scheme can be designed before a single observation are made, making adaptive sampling unnecessary.

2.4.4 Applications

Bayesian optimization has recently begun to appear in the machine learning literature as a means of optimizing difficult black box optimizations. A few recent examples include:

- Lizotte *et al.* [2007; 2008] used Bayesian optimization to learn a set of robot gait parameters that maximize velocity of a Sony AIBO ERS-7

robot. As an acquisition function, the authors used maximum probability of improvement (§2.2.1). They show that the Bayesian optimization approach not only outperformed previous techniques, but used drastically fewer evaluations.

- Frean and Boyle [2008] use Bayesian optimization to learn the weights of a neural network controller to balance two vertical poles with different weights and lengths on a moving cart.

- Cora's MSc thesis [2008] uses Bayesian optimization to learn a hierarchical policy for a simulated driving task. At the lowest level, using the vehicle controls as inputs and fitness to a course as the response, a policy is learned by which a simulated vehicle performs various activities in an environment.

- Martinez–Cantin *et al.* [2009] also applied Bayesian optimization to policy search. In this problem, the goal was to find a policy for robot path planning that would minimize uncertainty about its location and heading, as well as minimizing the uncertainty about the environmental navigation landmarks.

- Hutter's PhD thesis [2009] studies methods of automatically tuning algorithm parameters, and presents several sequential approaches using Bayesian optimization.

- The work of Osborne, Garnett *et al.* [Osborne, 2010; Osborne *et al.*, 2010; Garnett *et al.*, 2010b] uses Bayesian optimization to select the locations of a set of (possibly heterogenous) sensors in a dynamic system. In this case, the samples are a function of the locations of the entire set of sensors in the network, and the objective is the root mean squared error of the predictions made by the sensor network.

2.5 Experiments

We have conducted a series of experiments to demonstrate the effectiveness of our methods. In the experiments of Chapters 3–5, we study the behaviour

function	d
Branin	2
Hartman 3	3
Shekel 10	4
Hartman 6	6

Table 2.1: *Standard test functions used in this thesis, and their dimensionality. The function expressions are described in Appendix A.*

of our methods applied to known mathematical functions. By not using human judgement and adopting functions whose optima are known, we are able to measure performance precisely. As part of the discussion of our applications in Chapter 6, we will also bring users into the loop to validate our approach. While we know our methods work well in a simulated environment, we want to ensure that some overlooked aspect of human psychology does not cause our assumptions to be violated for the application.

Most previous studies of Bayesian optimization (e.g., [Schonlau, 1997; Streltsov and Vakili, 1999; Sasena, 2002; Lizotte, 2008]) have been concerned with the number of evaluations required to reach the global optimum to within some error, or the error after some pre-determined number of samples. In interactive Bayesian optimization, however, we consider it more desirable to know how well the optimization performs doing at each iteration, which tells us whether a user could be seen to be making progress—that is, it is important to track the evolution of the optimization, not just the final result. To measure optimization efficiency at each time step, Huang *et al.* [2006] suggest using the "gap" metric:

$$G = \frac{f(\mathbf{x}^{\text{first}}) - f(\mathbf{x}^{+})}{f(\mathbf{x}^{\text{first}}) - f(\mathbf{x}^{\star})}, \tag{2.6}$$

where $f(\mathbf{x}^{\text{first}})$ is the value of the first function sample (or in the case where multiple samples were used on the first iteration, the max of that set), and $f(\mathbf{x}^{+})$ is the best value seen so far. G will therefore be a number between 0, indicating no improvement over the initial sample ($f(\mathbf{x}^{+}) = f(\mathbf{x}^{\text{first}})$), and 1, meaning that the incumbent is the maximum ($f(\mathbf{x}^{+}) = f(\mathbf{x}^{\star})$).

2.5.1 Test functions

The set of test functions we used is based on previous work with Bayesian optimization: the full set of functions used in the PhD theses of Schonlau [1997], Sasena [2002] and Lizotte [2008], excepting the 'mystery' function unique to Sasena, and two of the two-dimensional functions. In our experiments we found that the additional 2D functions (the Goldstein-Price and 6-Hump Camelback) performed similarly to the Branin, and in any case we wish to focus our attention on higher-dimensional functions instead. Instead, we use the Hartman 3 function, so we can see the impact of increasing dimensionality on performance. Details of the test functions can be found in Appendix A.

The standard optimization test functions we use are listed in Table 2.1. We are aware of no widely-used test functions for Bayesian optimization with dimensionality higher than the 6-dimensional version of the Hartman function, which Schonlau, Sasena, Lizotte, Streltsov & Vakili, Finkel [2003] and Osborne *et al.* [2009] all use. Boyle [2007] uses synthetic functions to test the impact of dimensionality on Bayesian optimization, but also stops at 6 dimensions. (Streltsov and Vakili test a 10-dimensional version of the Rastrigin function, however this function is pathologically multimodal and not very relevant to our work.)

It's not at all clear, though, that even if there were standard test functions for higher dimensions that we would want to use them. As discussed in Chapter 1, it is trivial, especially in high dimensions, to create a synthetic objective function that requires an unfeasible number of evaluations to optimize. We are not interested here in solving this problem. In an interactive optimization problem, we are interested in problems we already know can be optimized by a human being, because in the past a human being *has* optimized them, even if by finding a set of good parameters with a parameter-twiddling interface. Instead, the problem is optimizing *efficiently*.

As there is no generally-agreed-upon test functions for Bayesian optimization in higher dimensions, we seek to sample synthetic functions from a known GP prior. Lizotte follows a similar strategy. A GP prior is infinite-

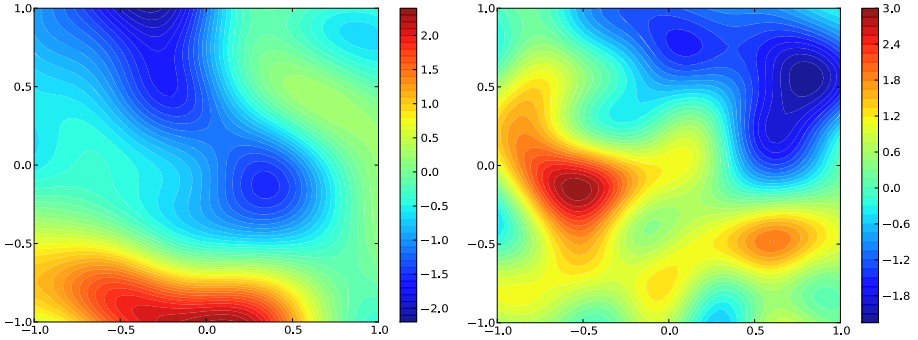


Figure 2.7: *Some randomly-generated test functions, sampled from a Gaussian ARD kernel with random hyperparameters.*

dimensional, so on a practical level for performing experiments we simulate this by sampling points from the prior. For each trial, we use an ARD kernel with $\boldsymbol{\theta}$ drawn randomly from $[0, 2]^d$ to get a mix of informative and uninformative dimensions. We then iteratively sample a fixed number of d-dimensional points using a Latin hypercube design, compute the covariance matrix $\mathbf{K}$, and sample $\mathbf{f} \sim \mathcal{GP}(0, \mathbf{K})$. Using the response surface as the objective function, we can then approximate the maximum using conventional global optimization techniques to get $f(\mathbf{x}^\star)$, which permits us to use the gap metric for performance. Note that these sample points are only used to construct the objective, and are not known to the optimization methods. Some examples of randomly-generated 2D synthetic functions are shown in Figure 2.7.

We will use these test functions in the machine experiments of Chapters 3–5 to test and verify our theoretical contributions. In Chapter 6, we will test our extensions with a real applications and human users.

Chapter 3

Preference galleries

The Bayesian optimization model described in Chapter 2 requires that each function evaluation have a continuous response. However, this is not always the case. In applications requiring human perception, for instance, preferences are often more accurate than ratings. Prospect theory, for example, employs utility models based on relation to a reference point, based on evidence that the human perceptual apparatus is attuned to evaluate differences rather than absolute magnitudes [Kahneman and Tversky, 1979; Tversky and Kahneman, 1992]. We present here a Bayesian optimization application based on discrete choice for a "preference gallery" application[1].

In the case of a person rating the suitability of a procedurally-generated animation or image, each sample of value involves creating an instance with the given parameters and asking a human to provide feedback, which is interpreted as the function response. This is a very expensive class of functions to evaluate! Furthermore, it is in general impossible to even sample the function directly and get a consistent response from users. Asking humans to rate an animation on a numerical scale has built-in problems—not only will scales vary from user to user, but human evaluation is subject to phenomena such as *drift*, where the scale varies over time, *anchoring*,

[1]This chapter is based on published work presented at SIGGRAPH (poster session) [Brochu *et al.*, 2007a] and Advances in Neural Information Processing Systems [Brochu *et al.*, 2007b]. Some experiments have been added to use the same experimental framework as the rest of this thesis.

in which early experiences dominate the scale [Siegel and Castellan, 1988; Payne *et al.*, 1993]. However, human beings *do* excel at comparing options and expressing a preference for one over others [Kingsley, 2006]. This insight allows us to approach the optimization function in another way. By presenting two or more realizations to a user and requiring only that they indicate preference, we can get more robust results with much less cognitive burden on the user [Kendall, 1975]. While this means we can't get responses for a value function directly, we model the value as a latent function, inferred from the preferences, which permits a Bayesian optimization approach.

3.1 Preferences

Learning preferences has been an active field of machine learning for some time. When we have a vocabulary of labels that could be applied and want to find an ordering, it is called *label preference*. However, what we are concerned with is the problem of *instance preference*, where we have data and we want to find a preference ordering over it. Much of the work in this field has involved casting instance preference as a binary classification problem [Wong *et al.*, 1988; Herbrich *et al.*, 1998], but we are instead going to cast the problem as one of probabilistic discrete choice over a latent function.

Probability models for learning from discrete choices have a long history in psychology and econometrics [Thurstone, 1927; Stern, 1990; McFadden, 2001]. They have been studied extensively for use in rating chess players, and the Elo Rating System [Élő, 1978] was adopted by the FIDE, the World Chess Federation, to model the probability of one player beating another. It has since been adopted to many other two-player games such as Go and Scrabble, and, more recently, online computer gaming [Herbrich and Graepel, 2006]. These methods differ from our work in that they are intended to predict the probability of a preference outcome over a finite set of possible pairs, whereas we work with infinite sets and are only incidentally interested in modelling outcomes. Glickman and Jensen [2005] use Bayesian optimal design for adaptively finding pairs for tournaments. Like our method, this

work uses the results of previous comparisons to select the next pair to be evaluated, and can be considered a form of active learning. It differs from our method in that it adopts different acquisition models and, more importantly, in the fact that the number of items being compared is finite.

Parts of our method are based on [Chu and Ghahramani, 2005b], which presents a preference learning method using probit models and Gaussian processes. They use a Thurstone-Mosteller model, but with an innovative nonparametric model of the utility. Note that the idea of using a latent Gaussian process prior and observations generated according to a generalized linear model has a much older history in spatial statistics (see, e.g. [Diggle *et al.*, 1998; Eidsvik *et al.*, 2009]).

3.2 Probit model for binary observations

The *probit model* allows us to deal with binary observations of $f(\cdot)$ in general. That is, every time we try a value of $\mathbf{x}$, we get back a binary variable, say either zero or one. From the binary observations, we have to infer the latent function $f(\cdot)$. In order to marry the presentation in this section to the user modelling applications discussed later, we will introduce probit models in the particular case of preference learning.

Assume we have shown the user M pairs of items from a set of N instances. For each pair, the user has identified a preference. The data set therefore consists of the ranked pairs:

$$\mathcal{D} = \{\mathbf{r}_i \succ \mathbf{c}_i; \;\; i = 1, \ldots, M\},$$

where the symbol $\succ$ indicates that the user prefers $\mathbf{r}$ to $\mathbf{c}$. We use $\mathbf{x}_{1:t}$ to denote the t distinct elements in the training data. That is, $\mathbf{r}_i$ and $\mathbf{c}_i$ correspond to two elements of $\mathbf{x}_{1:t}$. $\mathbf{r}_i \succ \mathbf{c}_i$ can be interpreted as a binary variable that takes value 1 when $\mathbf{r}_i$ is preferred to $\mathbf{c}_i$ and is 0 otherwise.

In the probit approach, we model the utility $v(\cdot)$ for items $\mathbf{r}$ and $\mathbf{c}$ as

follows:

$$\begin{aligned} v(\mathbf{r}_i) &= f(\mathbf{r}_i) + \varepsilon \\ v(\mathbf{c}_i) &= f(\mathbf{c}_i) + \varepsilon, \end{aligned} \tag{3.1}$$

where the noise terms are Gaussian: $\varepsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. Following [Chu and Ghahramani, 2005b], we assign a nonparametric Gaussian process prior to the unknown mean value: $f(\mathbf{x}') \sim \mathcal{GP}(0, k(\mathbf{x}', \mathbf{x}))$. That is, at the t training points:

$$P(\mathbf{f}) = |2\pi\mathbf{K}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f}\right),$$

where $\mathbf{f} = \{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_t)\}$.

Random utility models such as (3.1) have a long and influential history in psychology and the study of individual choice behaviour in economic markets. Daniel McFadden's Nobel Prize speech [McFadden, 2001] provides a glimpse of this history. More comprehensive treatments appear in classical economics books on discrete choice theory.

Under our Gaussian utility models, the probability that item $\mathbf{r}$ is preferred to item $\mathbf{c}$ is given by:

$$\begin{aligned} P(\mathbf{r}_i \succ \mathbf{c}_i | f(\mathbf{r}_i), f(\mathbf{c}_i)) &= P(v(\mathbf{r}_i) > v(\mathbf{c}_i) | f(\mathbf{r}_i), f(\mathbf{c}_i)) \\ &= P(\varepsilon - \varepsilon < f(\mathbf{r}_i) - f(\mathbf{c}_i)) \\ &= \Phi(Z_i), \end{aligned}$$

where

$$Z_i = \frac{f(\mathbf{r}_i) - f(\mathbf{c}_i)}{\sqrt{2}\sigma_{\text{noise}}}$$

and $\Phi(\cdot)$ is the CDF of the standard normal distribution. By including the noise term σ_{noise}, we allow for noisy inputs, including violations of the triangle inequality. This model, relating binary observations to a continuous latent function, is known as the Thurstone-Mosteller law of comparative judgement in psychology [Thurstone, 1927; Mosteller, 1951] or the intrapersonal random utility model in econometrics [McFadden, 1980]. In statistics

it goes by the name of binomial-probit regression. If the user had more than two choices one could adopt a polychotomous regression model [Holmes and Held, 2006]. This multi-category extension would, for example, enable the user to state no preference, or a degree of preference for any of the two items being presented. One could also easily adopt a logistic (sigmoidal) link function $\varphi(Z_i) = (1 + \exp(-Z_i))^{-1}$. In fact, such choice is known as the Bradley-Terry model [Stern, 1990].

Our goal is to estimate the posterior distribution of the latent utility function given the discrete data. That is, we want to maximize

$$P(\mathbf{f}|\mathcal{D}) \approx P(\mathbf{f}) \prod_{i=1}^{M} P(\mathbf{r}_i \succ \mathbf{c}_i | f(\mathbf{r}_i), f(\mathbf{c}_i)).$$

Although there exist sophisticated variational and Monte Carlo methods for approximating this distribution, we favour a common simple strategy: Laplace approximation [Diggle *et al.*, 1998; Chu and Ghahramani, 2005b]. Our motivation for doing this is the simplicity and computational efficiency of this technique. Moreover, given the amount of uncertainty in user valuations, we believe the choice of approximating technique plays a small role and hence we expect the simple Laplace approximation to perform reasonably in comparison to other techniques. For recent looks at more advanced techniques, we refer the reader to e.g. [Rue *et al.*, 2009; Eidsvik *et al.*, 2009].

The Laplace approximation follows from Taylor-expanding the log posterior about a set point $\widehat{\mathbf{f}}$:

$$\log P(\mathbf{f}|\mathcal{D}) = \log P(\widehat{\mathbf{f}}|\mathcal{D}) + \mathbf{g}^T(\mathbf{f} - \widehat{\mathbf{f}}) - \frac{1}{2}(\mathbf{f} - \widehat{\mathbf{f}})^T \mathbf{H} (\mathbf{f} - \widehat{\mathbf{f}}),$$

where $\mathbf{g} = \nabla_{\mathbf{f}} \log P(\mathbf{f}|\mathcal{D})$ and $\mathbf{H} = -\nabla_{\mathbf{f}} \nabla_{\mathbf{f}} \log P(\mathbf{f}|\mathcal{D})$. At the mode of the posterior ($\widehat{\mathbf{f}} = \mathbf{f}_{\mathrm{MAP}}$), the gradient $\mathbf{g}$ vanishes, and we obtain:

$$P(\mathbf{f}|\mathcal{D}) \approx P(\widehat{\mathbf{f}}|\mathcal{D}) \exp\left[-\frac{1}{2}(\mathbf{f} - \widehat{\mathbf{f}})\mathbf{H}(\mathbf{f} - \widehat{\mathbf{f}})\right]$$

In order to obtain this approximation, we need to compute the maximum a

posteriori (MAP) estimate $\mathbf{f}_{\mathrm{MAP}}$, the gradient $\mathbf{g}$ and the information matrix $\mathbf{H}$.

The gradient is given by:

$$\begin{aligned}
\mathbf{g} &= \nabla_{\mathbf{f}} \log P(\mathbf{f}|\mathcal{D}) \\
&= \nabla_{\mathbf{f}} \left[const - \frac{1}{2}\mathbf{f}^T \mathbf{K}^{-1}\mathbf{f} + \sum_{i=1}^{M} \log \Phi(Z_i) \right] \\
&= -\mathbf{K}^{-1}\mathbf{f} + \nabla_f \left[\sum_{i=1}^{M} \log \Phi(Z_i) \right] = -\mathbf{K}^{-1}\mathbf{f} + \mathbf{b},
\end{aligned}$$

where the j-th entry of the N-dimensional vector $\mathbf{b}$ is given by:

$$b_j = \frac{1}{\sqrt{2}\sigma_{\mathrm{noise}}} \sum_{i=1}^{M} \frac{\phi(Z_i)}{\Phi(Z_i)} \left[\frac{\partial}{\partial f(\mathbf{x}_j)} (f(\mathbf{r}_i) - f(\mathbf{c}_i)) \right],$$

where $\phi(\cdot)$ denotes the PDF of the standard normal distribution. Clearly, the derivative $h_i(\mathbf{x}_j) = \frac{\partial}{\partial f(\mathbf{x}_j)}(f(\mathbf{r}_i) - f(\mathbf{c}_i))$ is 1 when $\mathbf{x}_j = \mathbf{r}_i$, -1 when $\mathbf{x}_j = \mathbf{c}_i$ and 0 otherwise. Proceeding to compute the second derivative, one obtains the Hessian: $\mathbf{H} = \mathbf{K}^{-1} + \mathbf{C}$, where the matrix $\mathbf{C}$ has entries

$$\begin{aligned}
\mathbf{C}_{m,n} &= -\frac{\partial^2}{\partial f(\mathbf{x}_m)\partial f(\mathbf{x}_n)} \sum_{i=1}^{M} \log \Phi(Z_i) \\
&= \frac{1}{2\sigma^2} \sum_{i=1}^{M} h_i(\mathbf{x}_m) h_i(\mathbf{x}_n) \left[\frac{\phi(Z_i)}{\Phi^2(Z_i)} + \frac{\phi^2(Z_i)}{\Phi(Z_i)} Z_i \right]
\end{aligned}$$

The Hessian is a positive semi-definite matrix. Hence, one can find the MAP estimate with a simple Newton–Raphson recursion:

$$\mathbf{f}^{\mathrm{new}} = \mathbf{f}^{\mathrm{old}} - \mathbf{H}^{-1}\mathbf{g} \mid_{\mathbf{f}=\mathbf{f}^{\mathrm{old}}}.$$

At $\mathbf{f} = \mathbf{f}_{\mathrm{MAP}}$, we have

$$P(\mathbf{f}|\mathcal{D}) \approx \mathcal{N}\left(\mathbf{Kb}, (\mathbf{K}^{-1} + \mathbf{C})^{-1}\right).$$

preferences
$0.2 \succ 0.1$
$0.35 \succ 0.5$
$0.2 \succ 0.35$
$0.2 \succ 0.6$
$0.8 \succ 0.7$

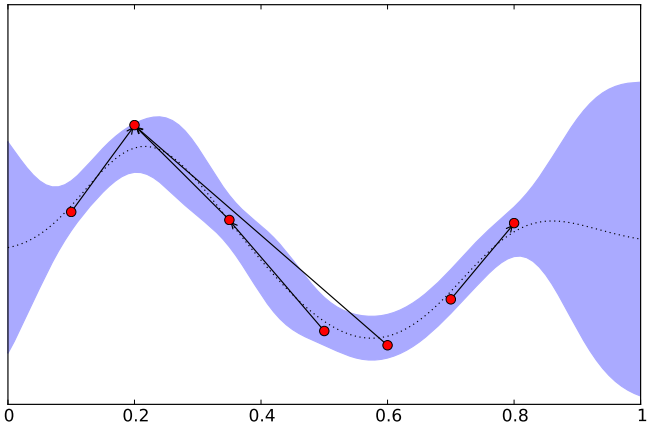


Figure 3.1: *Example of a set of preference relations (left table) used to infer a GP (right plot) on a toy problem. The preferences are indicated as a set of preferences between two points in the space, which serves as input to a function that finds a Gaussian process that takes into account all the available preference information, as well as prior information on the smoothness and noise.*

with $\mathbf{b} = \mathbf{K}^{-1}\mathbf{f}_{\text{MAP}}$. The goal of our derivation, namely the predictive distribution $P(f_{t+1}|\mathcal{D})$, follows by straightforward convolution of two Gaussians:

$$\begin{aligned} P(f_{t+1}|\mathcal{D}) &= \int P(f_{t+1}|\mathbf{f}_{\text{MAP}})P(\mathbf{f}_{\text{MAP}}|\mathcal{D})d\mathbf{f}_{\text{MAP}} \\ &\propto \mathcal{N}(\mathbf{k}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}}, k(\mathbf{x}_{t+1},\mathbf{x}_{t+1}) - \mathbf{k}^T(\mathbf{K}+\mathbf{C}^{-1})^{-1}\mathbf{k}). \end{aligned}$$

An example of the procedure on a toy 1D problem is shown in Figure 3.1. We can see that the inferred model explains the observed preferences, while also adhering to prior information about smoothness and noise.

3.3 Experiments: gallery labelling methods

To evaluate the preference gallery approach, we conducted a number of simulated gallery experiments using different techniques for modelling the GP. To do this, we use the known test functions described in Appendix A. In all cases, the experiments are run 25 times with random initialization. We use the gap metric (§2.5) to measure the evolution of the optimization.

We tested both simulated preference pairs (2-gallery) and a simulated environment in which pairs were defined over four instances at each iteration (4-gallery).

2-gallery

In the 2-gallery experiments, we iteratively selected pairs of data according to different methods. We then define preferences, which are added to the GP as training data (§3.2):

$$\begin{aligned} \mathbf{x}_i \succ \mathbf{x}_j \quad &\iff \quad f(\mathbf{x}_i) > f(\mathbf{x}_j) \\ \mathbf{x}_j \succ \mathbf{x}_i \quad &\iff \quad f(\mathbf{x}_j) > f(\mathbf{x}_i). \end{aligned}$$

Where f is the test function. In the very rare case where $f(\mathbf{x}_i) = f(\mathbf{x}_j)$, we randomly assign preference.

We tested three methods of sample selection[2]:

- $\operatorname{argmax} \mathrm{EI}, \operatorname{argmax} \mathrm{EI}_{+1}$. This method takes advantage of the fact that in a GP model, the covariance $\mathbf{K}$ is independent of the actual observations, so if we know $\mathbf{x}_{t+1}$, we can find $\sigma_{t+1}(\cdot)$ without knowing y_{t+1}. The first sample is obtained by maximizing EI (Eqn. (2.4)). We then update $\mathbf{K}$ for the sample and optimize EI again, using $\mu_t(\cdot)$ and $\sigma_{t+1}(\cdot)$. This method was also used by Schonlau [1997] to select sequential samples without observations.

- $\mathbf{x}^+, \operatorname{argmax} \mathrm{EI}$. In this method, the two samples for comparison are the incumbent and the argmax of EI. This is intended to directly seek improvement over the incumbent, though at the cost of displaying fewer unique samples to the user.

- random. The two points are drawn at random from the valid parameters, as a baseline comparison.

[2]Recent work by Azimi *et al.* [2011] has shown a novel and interesting method of batch Bayesian optimization by starting with a large set of candidates and iteratively pruning the least-promising until the desired batch size it achieved. While we do not test it here, it seems a promising direction.

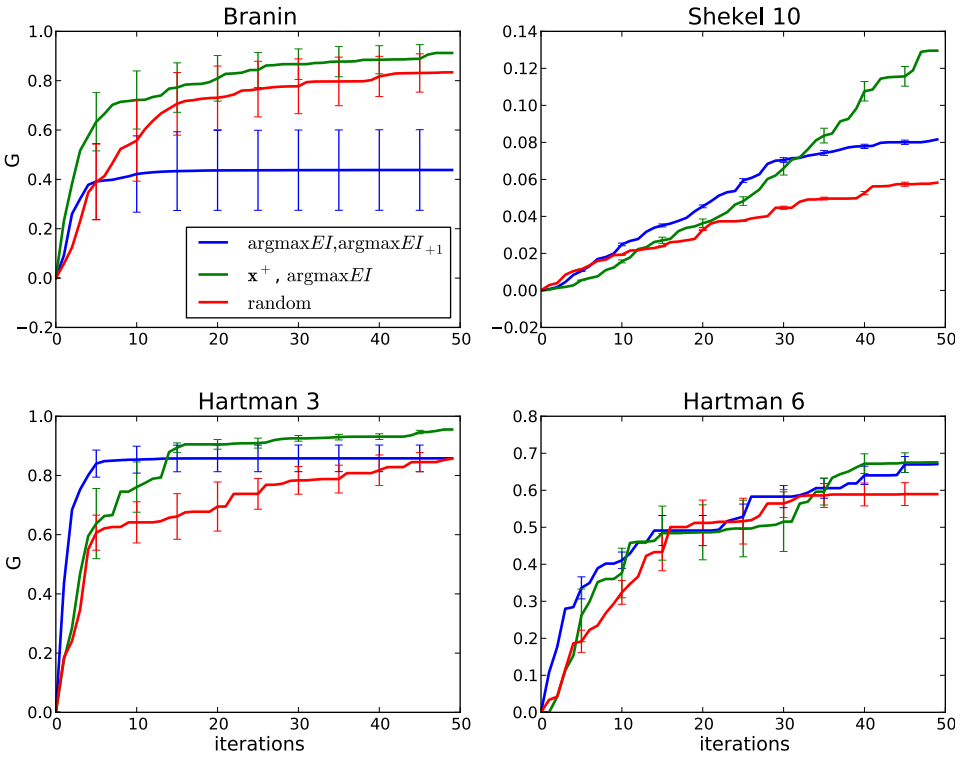


Figure 3.2: *Testing methods of selecting sample points for preferences on a set of test functions on a simulated 2-gallery. All methods draw two points at a time. The method labelled* $\operatorname{argmax}\mathrm{EI}, \operatorname{argmax}\mathrm{EI}_{+1}$ *selects the point of maximum EI, updates the covariance matrix and then samples the point of maximum EI on the updated model. The method labelled* $\mathbf{x}^+, \operatorname{argmax}\mathrm{EI}$ *used the incumbent and the point of maximum EI. The method labelled* `random` *selects two random points.*

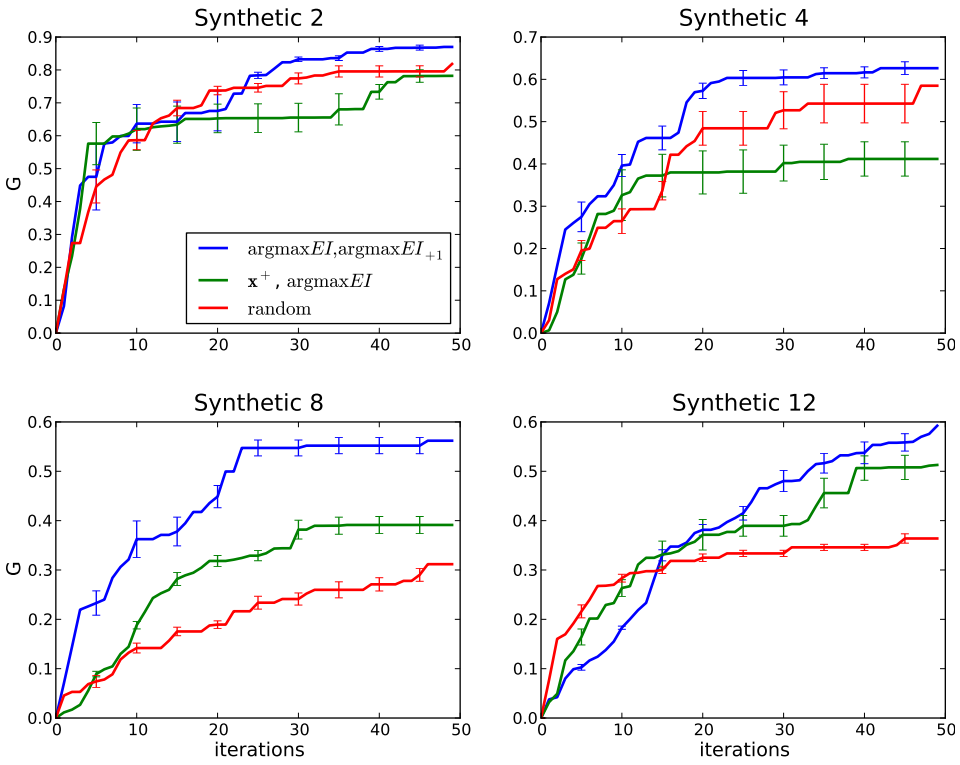


Figure 3.3: *Preference methods on the synthetic functions on a simulated 2-gallery, using the same methods used to generate Figure 3.2.*

We show the results of our experiments on functions from the literature (Appendix A) in Figure 3.2, and from synthetic functions (§2.5.1) in Figure 3.3. While there is no clear winner among the non-random methods, on the literature functions, using the $\mathbf{x}^+, \operatorname{argmax} \mathrm{EI}$ values is generally the best-performing overall, even though it involves only half as many distinct samples as the alternatives. Interestingly, on the Branin function, random selection performs very well, though all methods have high variance. Looking at the samples selected, we hypothesize that this is something of an artifact of the way the Branin function is constructed, where it is fairly easy for random samples to land close to the maximum, while more methodical techniques will tend to either start with a good sample and slowly improve, or first discount regions of high variance near the edges of the space first. On the synthetic functions, $\operatorname{argmax} \mathrm{EI}, \operatorname{argmax} \mathrm{EI}_{+1}$ tends to be the clear winner, particularly for higher-dimensional test functions. As dimensionality increases, and samples become further apart, exploring more of the space by drawing two previously unseen samples at each iteration rather than one sample and the incumbent becomes a more effective strategy.

4-gallery

In the 4-gallery experiment, we iteratively select a "gallery" of instances, $\mathbf{x}_{t+1}, \ldots, \mathbf{x}_{t+4}$ using one of the techniques described below. Preferences are then found for all pairs $\mathbf{x}_i, \mathbf{x}_j$ of instances in the gallery, where

$$\begin{aligned} \mathbf{x}_i \succ \mathbf{x}_j \quad &\Longleftrightarrow \quad f(\mathbf{x}_i) > f(\mathbf{x}_j) + \epsilon \\ \mathbf{x}_j \succ \mathbf{x}_i \quad &\Longleftrightarrow \quad f(\mathbf{x}_j) > f(\mathbf{x}_i) + \epsilon. \end{aligned}$$

($\epsilon \geq 0$ is a parameter to simulate two values so similar the user chooses to not distinguish them.) The GP is then updated with the preferences. The methods are extensions of the ones we use for the 2-gallery experiment.

- $\operatorname{argmax} \mathrm{EI}_{1:4}$. This method iteratively selects points, updates the covariance matrix, and optimizes EI to collect four samples for the gallery.

- $\mathbf{x}^+, \operatorname{argmax} \mathrm{EI}_{1:3}$. Compares the incumbent to three iterative samples taken by updating the covariance matrix and optimizing EI.
- `random`. The four points are drawn at random.

The results over the literature test functions are shown in Figure 3.4. In this case, the method of using the incumbent and three iterations of EI is clearly dominant. Studying the behaviour, we observe that using the incumbent allows for "anchoring" behaviour, in which highly-confident comparisons are made between local maxima until the best is found.

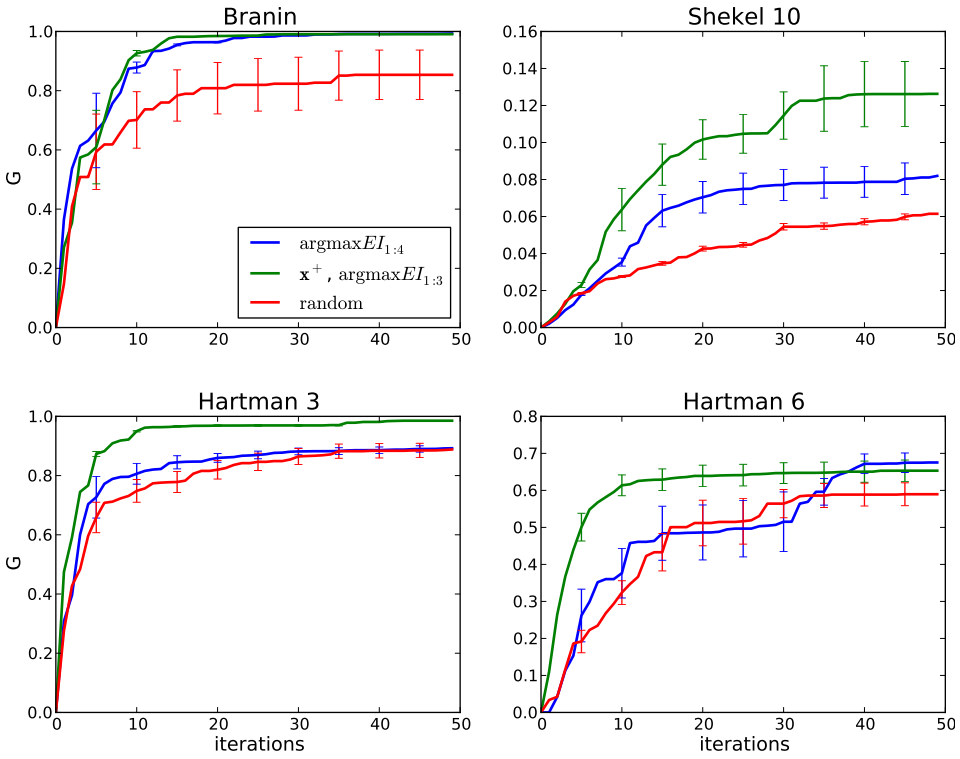


Figure 3.4: *Testing methods of selecting sample points for preferences on a set of test functions on a simulated 4-gallery. All methods draw four points at a time and then find preferences between every pair of points. The method labelled* $\operatorname{argmax} \mathrm{EI}_{1:4}$ *draws the four points by iteratively selecting the point of maximum* EI*, updating the covariance matrix and then sampling the point of maximum EI on the updated model, until 4 samples are selected. The method labelled* $\mathbf{x}^+, \operatorname{argmax} \mathrm{EI}_{1:3}$ *uses the same technique to find three samples, and uses the incumbent as the fourth. The method labelled* `random` *selects four random points.*

Chapter 4

Learning hyperparameters across user sessions

Our early work with Bayesian optimization proposed a technique to assist artists with parameter tuning for bidirectional reflectance distribution functions (*BRDF*s, see §6.3). Later, during the development of a procedural smoke animation system, we found ourselves with a parameterized system with 12 continuous parameters[1]. Setting these was a challenge for the developers, let alone other users, so we looked to adapt the methods we used previously [Brochu *et al.*, 2007a; 2007b]. In the process, though, we found that the settings of the kernel hyperparameters, which control the smoothness of the objective function on each of the parameters, has a very large impact on the quality of the optimization. We originally sought to have an expert set these, but our best attempts to do so were simply not good enough. Even worse, approaches in the literature for automatically setting these values require far more data than we have available [Jones, 2001; Santner *et al.*, 2003].

Furthermore, many regions of the parameter space (combinations of parameter ranges for the animation) never produce good animation instances.

[1]The animation problem is described in §6.4. That section and this chapter are based on published work presented at the ACM SIGGRAPH/Eurographics Symposium on Computer Animation [Brochu *et al.*, 2010a].

However, under the zero-mean Bayesian optimization model, until the user supplies evidence, parameter settings in these regions are no less likely to be selected than ones that are frequently chosen by other users. Particularly with the relatively large number of parameters we use, this requires users to repeatedly view and rate uninteresting settings until evidence is accrued, instead of first focussing on the most promising areas. In short, the earlier model does not offer an elegant way of incorporating user expertise into the prior. After even a small amount of time spent using the system, users had developed a good enough semantic understanding of certain parameters that they wanted to either set parameters to a specific value, or restrict the optimization to a user-defined region.

With these shortcomings in mind, we extended the interactive Bayesian optimization approach with several new features, which we incorporate into a novel application for assisting animators (§6.4). Our primary improvement, however, is a new hierarchical Bayesian method for incorporating data from previous users of the system to adaptively tune model hyperparameters. These hyperparameters control aspects such as the relative importance of changing settings of the different parameters, and the belief, before evidence is added, that an area of the space is unlikely to generate good animations. The problem of learning hyperparameters is usually difficult or impossible for small data sets, such as the ones generated in a single user session. We get around this by treating the problem as one of tracking the distributions of the hyperparameters across user sessions. As different users employ the system to find different animations, the system automatically improves and adapts.

We define a *session* as a single "use" of the system, starting from a model with no user data, and ending when the user has found a suitable instance or given up searching for it. While data such as the hyperparameter values might be tracked for some problems, at heart a session is set of samples and observations $\mathcal{D}_{1:T}$. Sessions are indexed by τ, so $\mathcal{D}_{1:3,\tau}$ refers to the first three instances of $\mathbf{x}, y$ for the τ^{th} session.

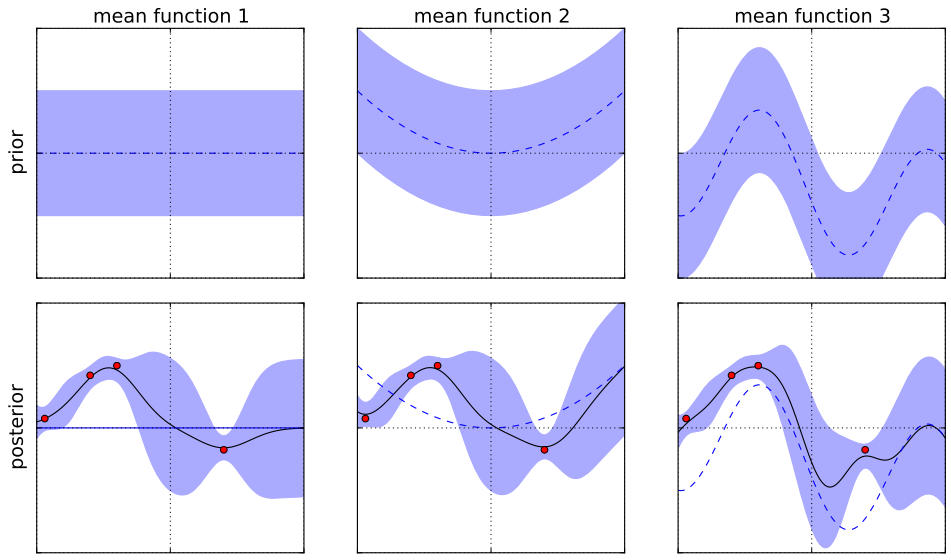


Figure 4.1: *Effect of different mean functions with and without evidence. The upper row shows the model with no observations. The lower row shows the model with four observations. Note that the models largely agree near the observations, but as the observations become more distant, the mean function becomes more strongly informative.*

4.1 Mean and covariance hyperparameters

In the previous sections, we assume very simple priors on the GP mean and covariance. In practice, however, we might want to use other priors, such as radial basis function networks. In this model, the mean $m(\cdot)$ is represented with radial basis functions (RBFs) with C centres $\mathbf{\Gamma} = \boldsymbol{\gamma}_{1:C}$ and coefficients $\boldsymbol{\beta} = \beta_{1:C}$:

$$m(\mathbf{x}) = \sum_{j=1}^{C} q(\|\mathbf{x} - \boldsymbol{\gamma}_j\|)\beta_j.$$

Where q is a squared exponential RBF of the form

$$q(z) = \exp(-z^2).$$

(If more flexibility were desired, this could introduce additional hyperparameters $\omega_{1:C}$, so that $q_j(z) = e^{-\omega_j z^2}$.) Note that a GP with a squared exponential covariance function is equivalent to this RBF model with $C = \infty$, and an alternate mean function would be to fit a GP to all prior user data and use the mean of that GP as the prior mean of the new GP. We refer the reader to, for example, [Bishop, 2006] for an introduction to RBF function approximation.

Adopting an informative mean function allows us to learn a general model of which areas of the space tend to generate high-value regions. The GP formulation allows an elegant way of trading off this prior knowledge with the observations. Near observation points, data dominates the prediction. Further from the observation points, the mean function dominates. Figure 4.1 shows some examples on a 1D model with some simple mean functions.

Subsequently, we describe the procedures for estimating the hyperparameters. To begin, we estimate of the locations of the basis centres $\mathbf{\Gamma}$ by, for example, C-means clustering the parameters of previous simulations. We can cluster because sample density near previous successful optimizations should be higher [Močkus, 1994], and we would like these cluster centroids to be the peaks of our mean function. Intuitively, the resulting clusters capture the different regions of the space that users are typically interested in.

By conditioning on $\mathbf{\Gamma}$, the coefficients $\boldsymbol{\beta}$ can be obtained analytically using standard Bayesian conjugate analysis. Specifically, we place a Gaussian hyperprior on them

$$\boldsymbol{\beta} \sim \mathcal{N}(0, \alpha\delta^2)$$

with regularizer δ^2. We place an inverse-gamma hyperprior on α with shape parameter $a > 0$ and scale parameter $b > 0$:

$$\alpha \sim \mathcal{IG}\left(\frac{a}{2}, \frac{b}{2}\right) \propto \left(\frac{1}{\alpha}\right)^{\frac{a}{2}+1} e^{-\frac{b}{2\alpha}}$$

We assume for now that the RBF centre estimates $\mathbf{\Gamma}$ are given, and turn

our attention to approximating the posteriors of $\boldsymbol{\beta}$ and α. We use as pseudo-measurement the function estimated by the Laplace method in §3.2. This two-stage approximation approach is motivated by computational simplicity. We model the pseudo-measurements with the following model:

$$P(\boldsymbol{\beta}, \alpha | \mathbf{X}, \mathbf{f}_{\text{MAP}}) \quad \propto \quad P(\mathbf{f}_{\text{MAP}} | \mathbf{X}, \boldsymbol{\beta}) P(\boldsymbol{\beta}) P(\alpha).$$

The posterior $P(\mathbf{f}_{\text{MAP}} | \mathbf{X}, \boldsymbol{\beta})$ follows from Bayes' rule:

$$\begin{aligned} P(\boldsymbol{\beta}, \alpha | \mathbf{X}, \mathbf{f}_{\text{MAP}}) &\propto \exp\left[-\frac{1}{2} (\mathbf{f}_{\text{MAP}} - \mathbf{Q}\boldsymbol{\beta})^T (\alpha \mathbf{K})^{-1} (\mathbf{f}_{\text{MAP}} - \mathbf{Q}\boldsymbol{\beta}) - \frac{1}{2\alpha\delta^2} \boldsymbol{\beta}^T \boldsymbol{\beta} \right] P(\alpha) \\ &= \exp\left[-\frac{1}{2\alpha} Z \right] P(\alpha) \end{aligned}$$

where $\mathbf{K}$ is the kernel matrix; $\mathbf{f}_{\text{MAP}}$ is the maximum a posteriori estimate of $\{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_t)\}$, which will be $\mu(\mathbf{x}_i)$ in the case of direct noisy observations or inferred from preferences in the binomial case (§3.2); $\mathbf{Q}$ is defined as

$$\mathbf{Q} = \begin{bmatrix} q(\mathbf{x}_1, \boldsymbol{\gamma}_1) & \ldots & q(\mathbf{x}_1, \boldsymbol{\gamma}_C) \\ \vdots & \ddots & \vdots \\ q(\mathbf{x}_t, \boldsymbol{\gamma}_1) & \ldots & q(\mathbf{x}_t, \boldsymbol{\gamma}_C) \end{bmatrix};$$

and

$$\begin{aligned} Z &= (\mathbf{f}_{\text{MAP}} - \mathbf{Q}\boldsymbol{\beta})^T \mathbf{K}^{-1} (\mathbf{f}_{\text{MAP}} - \mathbf{Q}\boldsymbol{\beta}) + \boldsymbol{\beta}^T \delta^{-2} \boldsymbol{\beta} \\ &= \mathbf{f}_{\text{MAP}}^T \mathbf{K}^{-1} \mathbf{f}_{\text{MAP}} + \boldsymbol{\beta}^T \mathbf{Q}^T \mathbf{K}^{-1} \mathbf{Q} \boldsymbol{\beta} - 2\mathbf{f}_{\text{MAP}}^T \mathbf{K}^{-1} \mathbf{Q} \boldsymbol{\beta} + \delta^{-2} \boldsymbol{\beta}^T \boldsymbol{\beta} \\ &= \boldsymbol{\beta}^T (\mathbf{Q}^T \mathbf{K}^{-1} \mathbf{Q} + \mathrm{I}\delta^{-2}) \boldsymbol{\beta} - 2\mathbf{f}_{\text{MAP}}^T \mathbf{K}^{-1} \mathbf{Q} \boldsymbol{\beta} + \mathbf{f}_{\text{MAP}}^T \mathbf{K}^{-1} \mathbf{f}_{\text{MAP}} \\ &= \boldsymbol{\beta}^T \mathbf{M}^{-1} \boldsymbol{\beta} - 2\mathbf{f}_{\text{MAP}}^T \mathbf{K}^{-1} \mathbf{Q} \boldsymbol{\beta} + \mathbf{f}_{\text{MAP}}^T \mathbf{K}^{-1} \mathbf{f}_{\text{MAP}}. \end{aligned}$$

For notational convenience, let the matrix $\mathbf{M}^{-1} = \mathbf{Q}^T \mathbf{K}^{-1} \mathbf{Q} + \mathrm{I}\delta^{-2}$ and

the vector $\mathbf{h} = \mathbf{M}\mathbf{Q}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}}$. Then

$$\begin{aligned} Z &= \boldsymbol{\beta}^T\mathbf{M}^{-1}\boldsymbol{\beta} - 2\mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{Q}\boldsymbol{\beta} + \mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} \\ &= \boldsymbol{\beta}^T\mathbf{M}^{-1}\boldsymbol{\beta} - 2\mathbf{h}^T\mathbf{M}^{-1}\boldsymbol{\beta} + \mathbf{h}^T\mathbf{M}^{-1}\mathbf{h} - \mathbf{h}^T\mathbf{M}^{-1}\mathbf{h} + \mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} \\ &= (\boldsymbol{\beta} - \mathbf{h})^T\mathbf{M}^{-1}(\boldsymbol{\beta} - \mathbf{h}) + \mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} - \mathbf{h}^T\mathbf{M}^{-1}\mathbf{h} \end{aligned}$$

We can use this to get an expression for the posterior:

$$\begin{aligned} P(\boldsymbol{\beta}, \alpha|\mathbf{X}, \mathbf{f}_{\text{MAP}}) \propto \quad & \exp\left[-\tfrac{1}{2\alpha}\mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} - \mathbf{h}^T\mathbf{M}^{-1}\mathbf{h}\right] |2\pi\alpha\mathbf{K}|^{-\frac{1}{2}} \\ & \times \exp\left[-\tfrac{1}{2\alpha}(\boldsymbol{\beta} - \mathbf{h})^T\mathbf{M}^{-1}(\boldsymbol{\beta} - \mathbf{h})\right] P(\alpha) \end{aligned}$$

Integrating out $\boldsymbol{\beta}$ and recalling that N is the number of data (and that $\mathbf{K}$ is therefore an N-by-N matrix), we get:

$$\begin{aligned} P(\alpha|\mathbf{X}, \mathbf{f}_{\text{MAP}}) &\propto P(\alpha)|2\pi\alpha\mathbf{K}|^{-\frac{1}{2}} \exp\left[-\frac{1}{2\alpha}(\mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} - \mathbf{h}^T\mathbf{M}^{-1}\mathbf{h})\right] \\ &\propto \alpha^{\frac{a}{2}+1+\frac{d}{2}} \exp\left[-\frac{1}{2\alpha}b + \mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} - \mathbf{h}^T\mathbf{M}^{-1}\mathbf{h}\right] \end{aligned}$$

This allows us to obtain analytical expressions for the posterior modes of these quantities:

$$\begin{aligned} \widehat{\boldsymbol{\beta}} &= \mathbf{h} = \frac{\mathbf{Q}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}}}{\mathbf{Q}^T\mathbf{K}^{-1}\mathbf{Q} + \delta^{-2}} \\ \widehat{\alpha} &= \frac{b'}{a'+1} = \frac{b + \mathbf{f}_{\text{MAP}}^T\mathbf{K}^{-1}\mathbf{f}_{\text{MAP}} - \mathbf{h}^T\mathbf{M}^{-1}\mathbf{h}}{a + N + 2} \end{aligned}$$

By approximating the posteriors of α and $\boldsymbol{\beta}$ with a Dirac-delta masses at $\widehat{\alpha}$ and $\widehat{\boldsymbol{\beta}}$, the mean and variance of the predictive distribution become:

$$\begin{aligned} \mu(\mathbf{x}) &= \sum_{i=1}^{C} q(\mathbf{x}, \boldsymbol{\gamma}_i)\widehat{\beta}_i + \mathbf{k}^T\mathbf{K}^{-1}\left(\mathbf{f}_{\text{MAP}} - \mathbf{Q}^T\widehat{\boldsymbol{\beta}}\right) \\ \sigma^2(\mathbf{x}) &= \left(k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k}\right)\widehat{\alpha} \end{aligned}$$

Note that this is somewhat similar to the process of universal kriging,

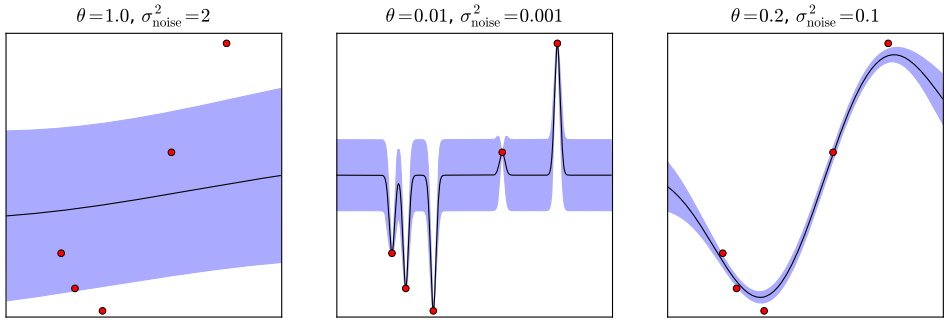


Figure 4.2: *Examples of the impact of kernel width (θ) and noise (σ^2_{noise}) hyperparameter settings on a toy 1D problem. All three models have the same data, but different hyperparameters. The leftmost model attributes most of the data to draws from a smooth but very noisy function. The centre model assumes the mean function holds everywhere except the close vicinity of the observations. The model on the right is more conservative, assuming a combination of smooth function and noise.*

in which a regression model is fit to the data with a GP used to model the residuals (§2.4.1). Here, however, we incorporate the regression model into the mean function and relax the universal kriging requirement that the regression model be unbiased.

4.1.1 Kernel hyperparameters

We now turn our attention to the process of inferring the kernel parameters $\boldsymbol{\theta}$ and σ^2_{noise}. In our user application, these have intuitive interpretations with regard to the impact on the user—$\boldsymbol{\theta}$ is the relationship between distance in the parameter space and the value function (§2.1.2), and σ^2_{noise} is the noise or uncertainty associated with the user's ratings (§2.3).

In many applications, the kernel hyperparameters can be learned by maximum a posteriori inference. This method works well for many application of GPs, but unfortunately for our application, it is known to work poorly when the number of training data is small [Hastie *et al.*, 2009]—exactly the situation we are in. The sparsity of data in the space can lead to some dimensions becoming very flat, or even monotonically increasing to infinity. This

can lead to low-quality models at best, and ill-conditioned covariance matrices at worst (Figure 4.2). In the experimental design literature, Bayesian optimization is often initialized using Latin hypercubes, which bypasses the need to learn parameters until enough data has been selected to make MAP feasible. However, the number of data required is significant, typically a minimum $10d$ [Jones, 2001; Santner *et al.*, 2003], though this can be aided with an informative hyperprior on the hyperparameters, often a log normal prior [Lizotte, 2008; Frean and Boyle, 2008].

4.2 Learning from the user base

Our central hypothesis is that the interests of users are regular. That is, there are some regions of the parameter space that are interesting to some (or all) users, and some that are never, or very rarely, of interest. Using the zero-function as the prior mean fails to capture this until data is added. We would like to use the results of previous trials (optimizations) to learn the hyperparameters of a semi-parametric prior for the current session. This is crucial to both reduce the number of user interactions and to scale the method to higher dimensions. Our insight is that *every time the application is used, a model is trained.* While different runs might involve users with different simulation goals, if we record the final user data $\mathcal{D}$ of each run, this can be used to generate a distribution over the hyperparameters, which identifies some hyperparameter settings as more likely than others, even before a new user starts using the system. It is important to emphasize that this does not directly affect the *actual* parameters of the animation—that is determined by the user's feedback. This is confined to the distributions of the hyperparameters. As more evidence is added to the model via user feedback, the hyperparameters can be updated to reflect that. The goal is not to restrict the hyperparameters, but to bring in knowledge gained from the results of previous users of the system.

We propose to learn the RBF hyperparameters (a, b) as well as the kernel hyperparameters $(\boldsymbol{\theta}, \sigma^2_{\text{noise}})$ using data gathered from previous optimizations. We will refer to these with the generic symbol $\boldsymbol{\rho}$.

We can model the sequence of optimizations conducted by the same user or different users with a dynamical state space model [Doucet *et al.*, 2001, Chapter 1]. In this dynamic model, the hyperparameters $\boldsymbol{\rho}$ correspond to the unknown states. The state space model consists of an initial belief $P(\boldsymbol{\rho}_0)$, a stochastic evolution process $P(\boldsymbol{\rho}_\tau|\boldsymbol{\rho}_{\tau-1})$ and an observation component $P(\mathbf{f}_\tau|\boldsymbol{\rho}_\tau)$. To be precise, the observation $\mathbf{f}_\tau$ is in fact the maximum a posteriori estimate $\mathbf{f}_{\text{MAP}}$ derived from the user feedback of run τ as outlined in the previous section, but we drop the MAP superscript to simplify the notation. Our goal is then to compute the optimal filtering distribution $P(\boldsymbol{\rho}_\tau|\mathbf{f}_{1:\tau})$ given that τ runs of the application have taken place. This distribution is intractable, but may be approximated with a Monte Carlo histogram estimator with samples obtained using a *particle filter* [Doucet *et al.*, 2001], as illustrated in Algorithm 2.

We treat the hyperparameters as independent, and use a separate dynamic Gaussian diffusion model for each hyperparameter[2]. The dynamic diffusion model allows the hyperparameters to converge to values that exhibit some random drift over time. We don't eliminate this drift as we believe it is useful to have a nonstationary model component in our application domain to take into account changes in the user, problem, application interface, etc.

At the beginning of each user session, we set the hyperparameters to the particle filter means estimated from previous trials. After each user session, we use the inferred function $\mathbf{f}$ to compute the fitness of each particle according to the following non-linear Gaussian observation likelihood model:

$$P(\mathbf{f}_\tau|\boldsymbol{\rho}_\tau) = |2\pi\mathbf{K}_{\rho_\tau}|^{-1/2} \exp\left[-\frac{1}{2}\mathbf{f}^T\mathbf{K}_{\rho_\tau}^{-1}\mathbf{f}\right].$$

In this expression, we have emphasized the dependency of $\mathbf{K}$ on the kernel hyperparameters $\boldsymbol{\rho}_\tau$ for clarity.

Algorithm 2 shows the particle filter method of updating for hyperparameters $\boldsymbol{\rho}$. This is a simple particle filter, where the importance sampling

[2]This is done for efficiency and interpretability—we could theoretically model the hyperparameters using a single particle filter if desired.

Algorithm 2 Particle Filter for Hyperparameter Learning

For $i = 1, \ldots, N$ sample $\boldsymbol{\rho}_0^{(i)} \sim P(\boldsymbol{\rho}_0)$.
$\tau = 1$
while True **do**
    For $i = 1, \ldots, N$ sample $\widetilde{\boldsymbol{\rho}}_\tau^{(i)} \sim P(\boldsymbol{\rho}_\tau | \boldsymbol{\rho}_{\tau-1}^{(i)})$
    For $i = 1, \ldots, N$ evaluate importance weights $\widetilde{w}_\tau^{(i)} = P(\mathbf{f}_\tau | \widetilde{\boldsymbol{\rho}}_\tau^{(i)})$ and normalize them.
    Resample with replacement N particles $\left(\boldsymbol{\rho}_{0:\tau}^{(i)}, i = 1, \ldots, N\right)$ from the set $\left(\widetilde{\boldsymbol{\rho}}_{0:\tau}^{(i)}, i = 1, \ldots, N\right)$ according to the importance weights $\widetilde{w}_\tau^{(i)}$.
    $\tau = \tau + 1$
end while

proposal is the transition prior $P(\boldsymbol{\rho}_\tau | \boldsymbol{\rho}_{\tau-1})$. If one has prior knowledge about the hyperparameters, it is possible to incorporate this into the design of more sophisticated proposal distributions [Doucet *et al.*, 2001]. A promising, and related, recent approach has been to marginalize over the hyperparameters using Bayesian Monte Carlo [Osborne *et al.*, 2008], which has been shown to outperform MAP methods [Garnett *et al.*, 2010a]. While combining this approach with ours is beyond the scope of this thesis, we feel this is a natural fit, and could be used to model much more sophisticated hyperparameter distributions effectively, though at some computational cost.

In essence, the model is continually learning from a variety of users doing a variety of tasks with it. The more it is used, the more representative and useful the priors and hyperpriors become.

4.3 Experiments: hyperparameter learning

In this section, we evaluate the performance of the particle filter (Algorithm 2) when optimizing a test function with known maxima: the Shekel 10 function (Appendix A). The function has 4 dimensions, 10 local maxima and 1 global maximum. Because of its dimensionality and fairly steep modes, it is difficult to optimize with naive techniques, but we can expect a well-designed general global optimizer to offer measurable improvement, even if it doesn't find the global maximum. This makes it ideal for study with the gap metric.

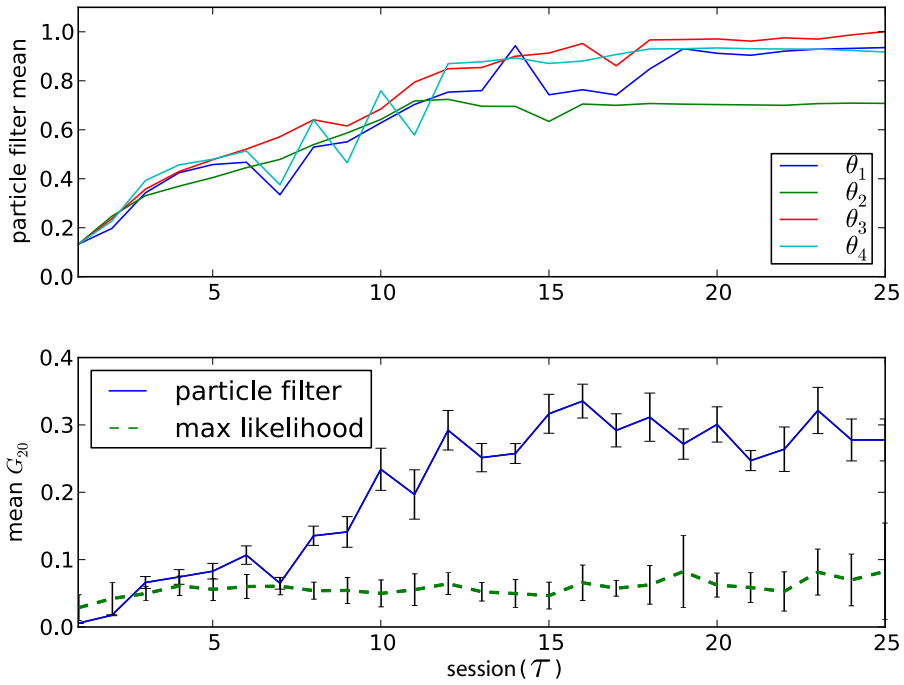


Figure 4.3: *Learning ARD kernel width hyperparameters on the Shekel 10 test function using a particle filter. The upper subfigure shows the evolution of $\theta_{1:4}$ over 25 sessions. The lower subfigure shows the performance measure G corresponding to these hyperparameters. It also shows the same measure using hyperparameters learned by maximum likelihood, using all the accumulated data. We can see that ML is fundamentally unable to improve above a low baseline due its inability to recover from poor initial estimates of the hyperparameters.*

We fix the GP mean to zero and focus on studying the effect of learning only the ARD kernel width hyperparameters, $\boldsymbol{\theta}_{1:4,\tau}$. In this optimization setting, the observations are noise-free (we will examine the effect of learning the mean function in §4.4, and the interaction between noise and kernel width hyperparameters has been well-studied elsewhere [Santner *et al.*, 2003; Lizotte, 2008]). We train the particle filters for sessions τ as shown in Algorithm 2. At each τ, we gather an observation vector $\mathbf{f}_\tau$ by running 20 iterations of Bayesian optimization, using the mean of the predictive distribution $p(\boldsymbol{\theta}_\tau|\mathbf{f}_{1:\tau-1})$. The simulation of $\mathbf{f}_\tau$ is not deterministic because

of random initialization. The procedure continues as detailed in Algorithm 2.

We store the 4 mean trajectories for 25 runs, $\widehat{\boldsymbol{\theta}}_{1:4,1:25}$, computed with the particle filter. For each of these trajectory values, we run 20 iterations of Bayesian optimization and record the mean and variance of G (Eqn. (2.6)), called here G_{20}. We adopted 20 iterations because this is roughly the point at which users of the animation system start to quit if they do not see significant improvement. We repeated the experiment 10 times to obtain confidence estimates. The evolution of G_{20} is shown in the lower plot of Figure 4.3. For comparison, we also show G_{20} for hyperparameters learned by maximizing the likelihood directly: for each session τ, data from *all previous sessions* $\mathcal{D}_{1:20,1:\tau-1}$ are used to find the MAP estimate of the hyperparameters, and these hyperparameter values are used for the τ^{th} session.

The results show that the particle filter not only converges at a reasonably fast rate, but also leads to significant improvements in optimization performance. After about 15 sessions, there is little further variation in the particle filter means, and optimization performance reaches a plateau of performance. The maximum likelihood method, meanwhile, forms a kind of performance baseline, and consistently performs very poorly on the Shekel 10 function. This was somewhat unanticipated, as ML might be expected to get better hyperparameter estimates with more data. What appears to be happening is that the estimates from the first session are so bad (all θ values are either close to 0 or extremely high) that the optimization on the next session draws unrepresentative samples (either repeatedly sampling the same tiny region, or sampling the corners of the space). With this data, it is nearly impossible to learn better θ values. This doesn't happen with the particle filter model, because there is a limit to the range of the hyperparameter values at each session, so a single disastrous estimate cannot propagate forward. We conjecture that using the ML method with a well-designed hyperprior would improve performance, though we did not study this scenario.

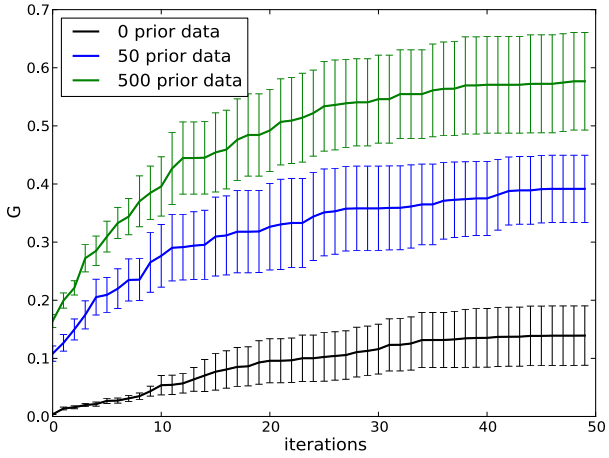


Figure 4.4: *Effect of adding more data to compute the RBF prior mean. We run Bayesian optimization to find the global maximum of the Shekel 10 function, with G a measure of the optimization performance. Solid lines are the mean performance at each iteration and error bars show the variance. We can see even 50 prior data offer a dramatic improvement in the speed and quality of the optimization, and 500 offers yet more improvement.*

4.4 Experiments: learning the mean function from an RBF network

To test the impact of the mean function on the optimization function, again use the Shekel 10 test function. We sample randomly and with noise, and use that data to learn the $\boldsymbol{\beta}$ and $\boldsymbol{\Gamma}$ values of the RBF network that we use as a prior. We then optimize the function iteratively by selecting the point of maximum EI and sampling from f and updating the GP. We ran tests using 50 and 500 data to train the RBF network, using Latin hypercube sampling and adding noise ε to the samples, where $\varepsilon \sim \mathcal{N}(0, 0.5)$. We set the size of the network (arbitrarily) at $p = 25$ bases. We run the experiments for 50 iterations, which we feel is the most most users would find acceptable.

Figure 4.4 shows the effect of the prior on optimizing the network. We can see that an RBF network trained on even 50 noisy samples offers striking improvement on the rate of optimization, and 500 samples even more.

Chapter 5

Portfolio strategies for acquisition selection

Bayesian optimization is efficient and can be used when very little is known about the objective function, making it popular for expensive black-box optimization scenarios. It is able to do this by sampling the objective using an acquisition function which incorporates the model's estimate of the objective and the uncertainty at any given point. However, we saw in §2.2 that there are several different parameterized acquisition functions in the literature, and it is often unclear which one to use. In this chapter, instead of using a single acquisition function, we adopt a portfolio of acquisition functions governed by an online multi-armed bandit strategy[1]. We describe the method, which we call GP-Hedge, which builds on recent developments in the field of online learning and multi-armed bandits [Cesa-Bianchi and Lugosi, 2006] and we will show that in our experiments GP-Hedge consistently outperforms the best individual acquisition function.

There is no choice of acquisition function that can guaranteed to perform best on an arbitrary, unknown objective[2]. In fact, it may be the case that no single acquisition function will perform the best over an entire

[1] This chapter is based on work published as an arXiv e-print [Brochu *et al.*, 2010b].

[2] While studying the objective might allow an expert to make an educated guess, even this can be difficult as Bayesian optimization is normally used specifically when sampling the objective is expensive.

Algorithm 3 Hedge [Auer *et al.*, 1998]

Select parameter $\eta \in \mathbb{R}^+$.
Set $g_0^i = 0$ for $i = 1, \ldots, K$.
for $t = 1, 2, \ldots$ **do**
    Choose action i_t with probability $p_t(i) = \exp(\eta g_{t-1}^i) / \sum_{\ell=1}^K \exp(\eta g_{t-1}^\ell)$.
    Receive rewards r_t^i.
    Update gains $g_t^i = g_{t-1}^i + r_t^i$.
end for

Algorithm 4 GP-Hedge

Select parameter $\eta \in \mathbb{R}^+$.
Set $g_0^i = 0$ for $i = 1, \ldots, K$.
for $t = 1, 2, \ldots$ **do**
    Nominate points from each acquisition function: $\mathbf{x}_t^i = \operatorname{argmax}_{\mathbf{x}} u_i(\mathbf{x} | \mathcal{D}_{1:t-1})$.
    Select nominee $\mathbf{x}_t = \mathbf{x}_t^j$ with probability $p_t(j) = \exp(\eta g_{t-1}^i) / \sum_{\ell=1}^K \exp(\eta g_{t-1}^\ell)$.
    Sample the objective function $y_t = f(\mathbf{x}_t) + \epsilon_t$.
    Augment the data $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.
    Receive rewards $r_t^i = \mu_t(\mathbf{x}_t^i)$ from the updated GP.
    Update gains $g_t^i = g_{t-1}^i + r_t^i$.
end for

optimization—a mixed strategy in which the acquisition function samples from a pool at each iteration might work better than any single acquisition.

Previous work in this problem involved a strategy that alternated exploration and exploitation phases, such as the $P^\star$-algorithm for one-dimensional optimization [Törn and Žilinskas, 1989]. However, we wish to automatically balance a portfolio of algorithms. This approach can be treated as a hierarchical multi-armed bandit problem, in which each of the K arms is an acquisition function, each of which is itself an infinite-armed bandit problem.

5.1 Hedge algorithms

Hedge (Algorithm 3) is an algorithm which at each time step t selects an action i with probability $p_t(i)$ based on the cumulative rewards (gain) for that action [Auer *et al.*, 1998]. After selecting an action the algorithm receives reward r_t^i for each action and updates the gain vector. In the Bayesian optimization setting, we can define K bandits each corresponding to a single acquisition function. Choosing action i corresponds to sampling from the point

Algorithm 5 Exp3 [Auer *et al.*, 1998]

1: Select parameters $\eta \in \mathbb{R}^+$, $\gamma \in (0, 1]$.
2: **for** $t = 1, 2, \ldots$ **do**
3: Get distribution $\mathbf{p}_t$ from Hedge (Algorithm 3).
4: Choose action i_t to be j with probability $\hat{p}(j) = (1 - \gamma)p_t(j) + \gamma/K$.
5: Receive reward $r_t^{i_t} \in [0, 1]$.
6: Return simulated reward vector $\hat{\mathbf{r}}_t$ to Hedge, where $\hat{r}_t^i = r_t^i / \hat{p}_t(i)$ if $j = i_t$, or $\hat{r}_t^i = 0$ otherwise.
7: **end for**

nominated by function u_i, i.e., $\mathbf{x}_t^i = \text{argmax}_{\mathbf{x}}\, u_i(\mathbf{x}|\mathcal{D}_{1:t-1})$ for $i = 1, \ldots, K$. While in the conventional Bayesian optimization setting the objective function is sampled only once per iteration, Hedge is a "full information game", meaning it requires a reward r_t^i for every action at every time step. We can achieve this by defining the reward at $\mathbf{x}_t^i$ as the expected value of the GP model at $\mathbf{x}_t^i$. That is, $r_t^i = \mu_t(\mathbf{x}_t^i)$. We refer to this method as GP-Hedge. Provided that the objective function is smooth, this reward definition is reasonable.

The GP-Hedge procedure is shown in Algorithm 4. Note that it is necessary to optimize K acquisition functions at each time step rather than 1. While this might seem expensive, this is unlikely to be a major problem in practice for small K, as (i) Bayesian optimization is typically employed when sampling the objective is so expensive as to dominate other costs, including acquisition function optimization; (ii) approximate optimization of u is usually sufficient [Močkus, 1994]; and (iii) it is straightforward to optimize the acquisitions functions in parallel on a modern multicore processor.

Auer *et al.* also propose *Exp3* (Algorithm 5), a variant of Hedge that applies to the partial information game. In this setting it is no longer assumed that rewards are observed for all actions. Instead at each iteration a reward is only associated with the particular action selected. The algorithm uses Hedge as a subroutine where rewards observed by Hedge at each iteration are $r_t^i / \hat{p}_t(i)$ for the action selected and zero for all other actions. Here $\hat{p}_t(i)$ is the probability that Hedge would have selected action i. The Exp3 algorithm, meanwhile, selects actions from a distribution that is a mixture between $\hat{p}_t(i)$ and the uniform distribution. Intuitively this ensures that the

Algorithm 6 NormalHedge [Chaudhuri *et al.*, 2009]

Initialize $R_0^i = 0, p_0^i = 1/K$.
for $t = 1, 2, \ldots$ **do**
  Each action i incurs loss ℓ_t^i.
  Learner incurs loss $\ell_t^A = \sum_{i=1}^K p_t^i \ell_t^i$.
  Update cumulative regrets $R_t^i = R_{t-1}^i + (\ell_t^A - \ell_t^i)$.
  Use line search to find $c_t > 0$ satisfying $1/K \sum_{i=1}^K \exp(([R_t^i]_+)^2/(2c_t)) = e$, where $[R_t^i]_+ = \max\{0, R_t^i\}$.
  $p_{t+1}^i \propto [R_t^i]_+ / c_t \exp(([R_t^i]_+)^2/(2c_t))$.
end for

algorithm does not miss good actions because the initial rewards were low (i.e., it continues exploring all strategies, albeit less frequently).

In comparing these two hedging strategies we will first note that GP-Hedge is somewhere "in between" a full and a partial information game. For example, the method collapses to a partial information game if all sample points are too distant, such that the kernels of these points exert negligible influence on each other. However, this behaviour is not observed in practical situations because of smoothness and our particular selection of acquisition functions.

Another possible strategy is the *NormalHedge* algorithm [Chaudhuri *et al.*, 2009], shown in Algorithm 6. We adapt NormalHedge by at each step sampling a single action i_t to be j with (normalized) probability p_t^j and defining the loss as

$$\ell_t^i = \frac{1}{1 + \exp\left(\mu(\mathbf{x}_t^i)\right)}$$

to meet the form required by NormalHedge. The algorithm has the appealing quality that non-zero weights are only applied to actions that perform better than the mean performance, so unproductive acquisition functions are quickly discarded. However, NormalHedge is also designed to take advantage of situations where the number of bandit arms (acquisition functions) is large, and may not be a good match to problems where K is relatively small.

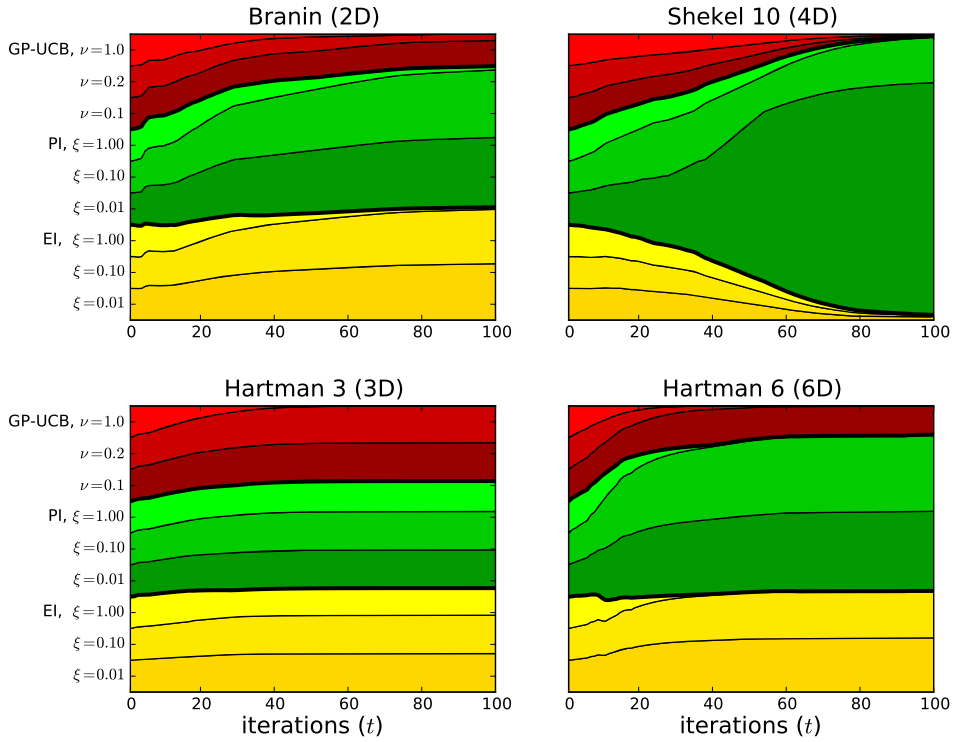


Figure 5.1: *Examples of typical evolution of GP-Hedge's portfolio with $K = 9$ for each objective function. The height of each band corresponds to the probability $p_t(i)$ at each iteration.*

5.2 Experiments: acquisition functions and acquisition strategies

We first tested performance using our standard suite of test functions: the Branin, Shekel 10, Hartman 3 and Hartman 6 functions (Appendix A). For each experiment, we optimized 25 times with random initialization and computed the mean and variance of the gap metric (§2.5) over time. In these experiments we first learned the hyperparameters $\boldsymbol{\theta}$ by sampling $10d$ points with the Latin hypercube method and maximizing the log marginal likelihood of the data given $\boldsymbol{\theta}$. These samples were then discarded. When also tested on synthetic functions (§2.5). For the synthetic functions, for

each trial, we sample a different function and optimize it using the different techniques, so each plot shows the mean result over 25 different functions, with one trial each.

We compared the standard acquisition functions using parameters suggested by previous authors, i.e., $\xi = 0.01$ for EI and PI, $\delta = 0.1$ and $\nu = 0.2$ for GP-UCB [Lizotte, 2008; Srinivas *et al.*, 2010]. The plots for EI, PI and GP-UCB shown all use these parameters. For the GP-Hedge trials, we tested performance under using both 3 acquisition functions and 9 acquisition functions. For the 3-function variant we use the standard acquisition functions with default hyperparameters. The 9-function variant uses these same three as well as 6 additional acquisition functions consisting of: both PI and EI with $\xi = 0.1$ and $\xi = 1.0$, GP-UCB with $\nu = 0.1$ and $\nu = 1.0$. Used alone, these values are not expected to perform as well as the defaults and our experiments confirmed this hypothesis. However, we are curious to see if adding known suboptimal acquisition functions will help or hinder GP-Hedge in practice.

Results for the gap measure G_t are shown in Figure 5.2. While the improvement GP-Hedge offers over the best single acquisition function varies, there is almost no combination of function and time step in which the 9-function GP-Hedge variant is not the best-performing method. The results suggest that the extra acquisition functions assist GP-Hedge in exploring the space in the early stages of the optimization process. Figure 5.2 also displays, for a single example run, how the the arm probabilities $p_t(i)$ used by GP-Hedge evolve over time. We have observed that the distribution becomes more stable when the acquisition functions come to a general consensus about the best region to sample. As the optimization progresses, exploitation becomes more rewarding than exploration, resulting in more probability being assigned to methods that tend to exploit. However, note that if the initial portfolio had consisted only of these more exploitative acquisition functions, the likelihood of entrapment at suboptimal points would have been higher.

In Figures 5.3 and 5.4 we compare against the other Hedging strategies introduced in §5.1 under both the gap measure and mean average regret. We

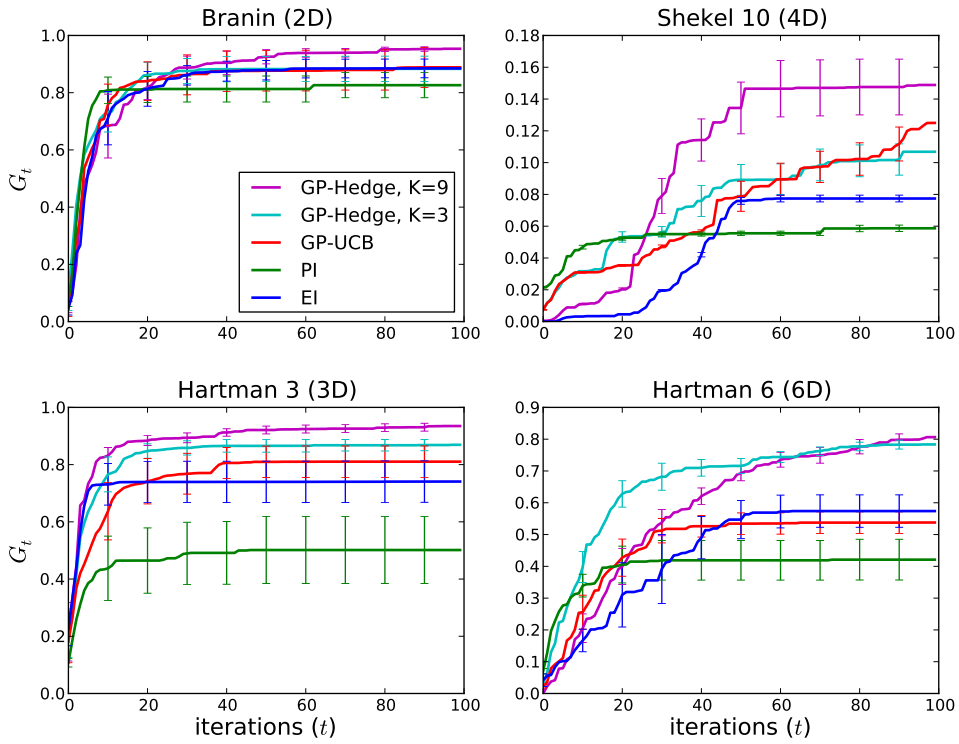


Figure 5.2: *Comparison of different acquisition approaches on four commonly-used test functions. The lines and error bars in the subplots show the mean and variance of the gap metric averaged over 25 trials. With $K = 3$ acquisition functions, GP-Hedge beats the best-performing acquisition function in almost all cases. With $K = 9$, we add additional instances of the three acquisition functions, but with different parameters. Despite the fact that these additional functions individually perform worse than the ones with default parameters, adding them to GP-Hedge improves performance in the long run.*

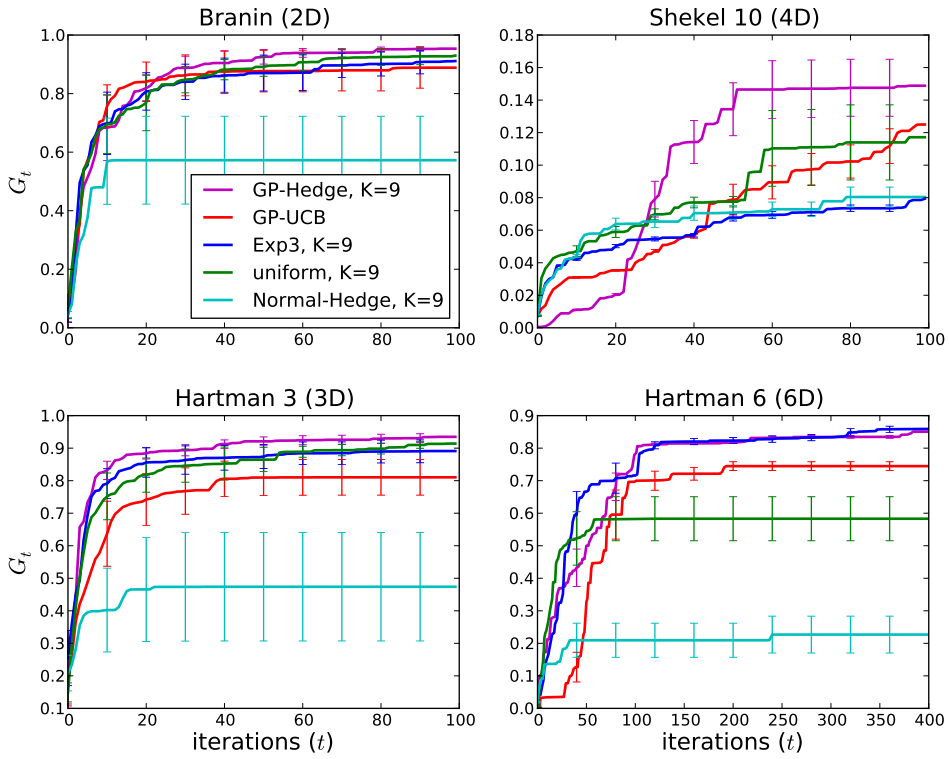


Figure 5.3: *Comparison of different hedging strategies on three commonly-used test functions. The subplots show the mean and variance of the gap metric averaged over 25 trials.*

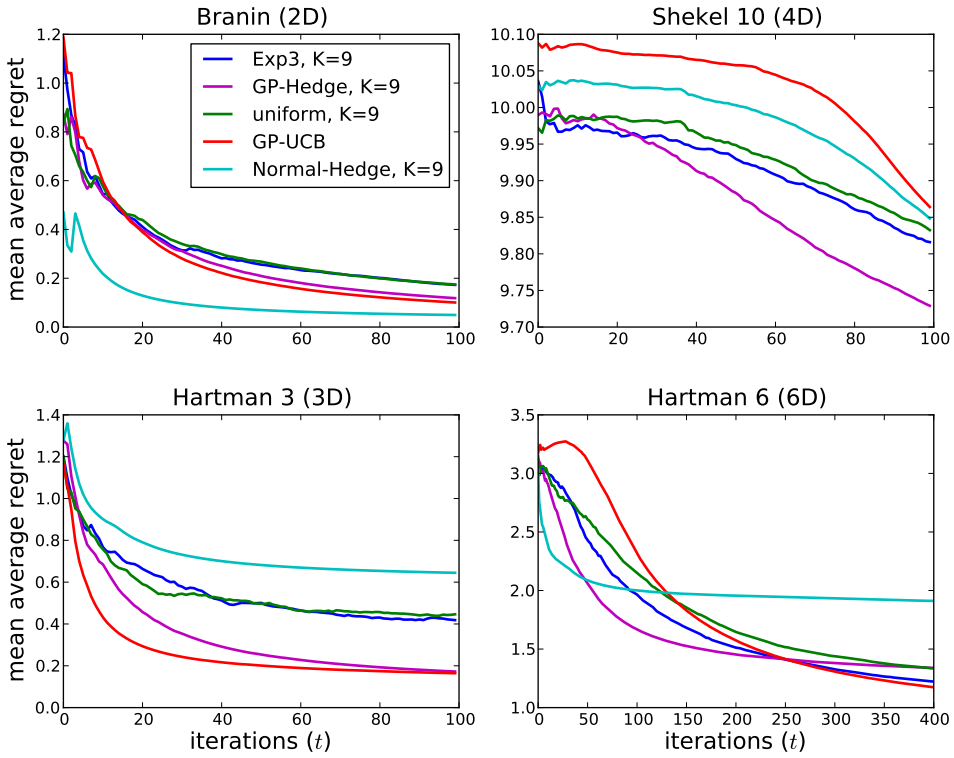


Figure 5.4: *Comparison of different hedging strategies on four commonly used literature functions. The subplots plots show the mean average regret for each method (lower is better). We see that mixed strategies (i.e., GP-Hedge) perform comparably to GP-UCB under the regret measure and outperform this individual strategy under the gap measure (Figure 5.3). As the problems get harder, and with higher dimensionality, GP-Hedge outperforms other mixed strategies.*

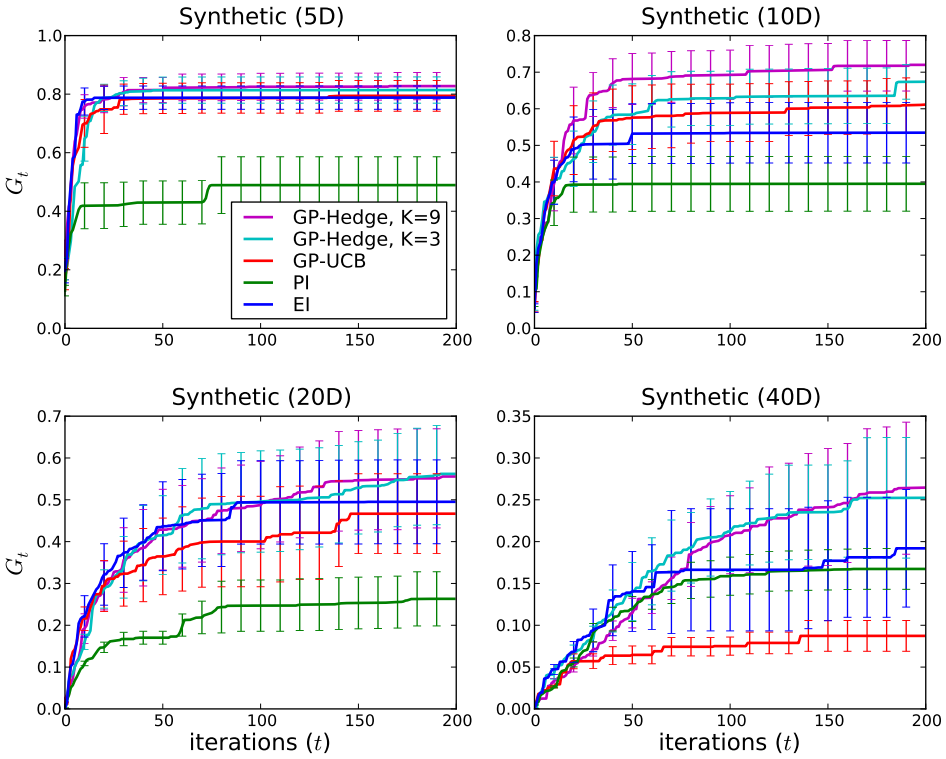


Figure 5.5: *Comparison of performance of the acquisition approaches on synthetic functions sampled from a GP prior with randomly initialized hyperparameters. Shown are the mean and variance of the gap metric over 25 sampled functions.* ***Note, the variance in this plot is very different from the variance shown in previous plots.*** *Here, the variance is a relative measure of how well the various algorithms perform while the functions themselves are varied. While the variance is high (which is to be expected over diverse functions), we can see that Hedge is at least comparable to the best acquisition functions and usually better. This confirms our thesis that, in the absence of strong knowledge about the best acquisition to use, it is sensible to use Hedge.*

also introduce a baseline strategy which utilizes a portfolio uniformly distributed over the same acquisition functions. The results show that mixing across multiple acquisition functions provides significant performance benefits under the gap measure, and as the problems' difficulty/dimensionality increases we see that GP-Hedge outperforms other mixed strategies. The uniform strategy performs well on the easier test functions, as the individual acquisition functions are reasonable. However, for the hardest problem (Hartman 6) we see that the performance of the naive uniform strategy degrades. NormalHedge performs particularly poorly on this problem. We observed that this algorithm very quickly collapses to an exclusively exploitative portfolio which becomes very conservative in its departures from the incumbent. We again note that this strategy is intended for large values of K, which may explain this behaviour.

In the case of the regret measure we see that the hedging strategies perform comparable to GP-UCB, a method designed to optimize this measure. We further note that although the average regret can be seen as a lower-bound on the convergence of Bayesian optimization methods, this bound can be loose in practice. Further, in the setting of Bayesian optimization we are typically concerned not with the cumulative regret during optimization, but instead with the regret incurred by the incumbent after optimization is complete—that is, the instantaneous regret of the best observation, independent of the regrets of the other observations. Similar notions of "simple regret" have been studied in [Audibert *et al.*, 2010; Bubeck *et al.*, 2009]. Based on the performance in these experiments, we use Hedge as the underlying algorithm for GP-Hedge in the remainder of the experiments.

As can be seen in Figure 5.5, GP-Hedge with $K = 9$ is again the best-performing method, which becomes even more clear as the dimensionality increases. Interestingly, the *worst*-performing function changes as dimensionality increases. In the 40-D experiments, GP-UCB, which generally performed well in other experiments, does quite poorly. Examining the behaviour, it appears that by trying to minimize regret instead of maximizing improvement, GP-UCB favours regions of high variance. However, since

a 40-D space remains extremely sparsely populated even with hundreds of samples, the vast majority of the space still has high variance, and thus high acquisition value.

5.3 Experiment: control of a particle simulation

We also applied these methods to optimize the behaviour of a simulated physical system in which the trajectories of falling particles are controlled via a set of repelling forces. This is a difficult, nonlinear control task whose resulting objective function exhibits fairly isolated regions of high value surrounded by severe plateaus. Briefly, the four-dimensional state-space in this problem consists of a particle's 2D position and velocity $(\mathbf{p}, \dot{\mathbf{p}})$ with two-dimensional actions consisting of forces which act on the particle. Particles are also affected by gravity and a frictional force resisting movement. The goal is to direct the path of the particle through regions of the state space with high reward $r(\mathbf{p})$ so as to maximize the *total reward* accumulated over many time-steps. In our experiments we use a finite, but large, time-horizon H. In order to control this system we employ a set of "repellers" each of which is located at some position $C_i = (A_i, B_i)$ and has strength W_i (see the upper plot of Figure 5.6). The force on a particle at position $\mathbf{p}$ is a weighted sum of the individual forces from all repellers, each of which is inversely proportional to the distance $\mathbf{p} - C_i$. For further details on this model we refer the reader to [Hoffman *et al.*, 2009]. The use of Bayesian optimization for control problems has previously been proposed by Frean and Boyle [2008], who used it for a control problem of balancing two poles on a cart (§2.4.4).

This problem can be formulated in the setting of Bayesian optimization by defining the vector of repeller parameters $\mathbf{x} = (W_1, A_1, B_1, \dots)$. In the experiments shown in Figure 5.6 we will utilize three repellers, resulting in a 9D optimization task. We can then define our objective as the total H-step expected reward

$$f(\mathbf{x}) = \mathbb{E}\big[\textstyle\sum_{t=0}^{H} r(p_t)|\mathbf{x}\big].$$

Finally, since the model defines a probability distribution $P_{\mathbf{x}}(\mathbf{p}_{0:H})$ over particle trajectories we can obtain a noisy estimate of this objective function

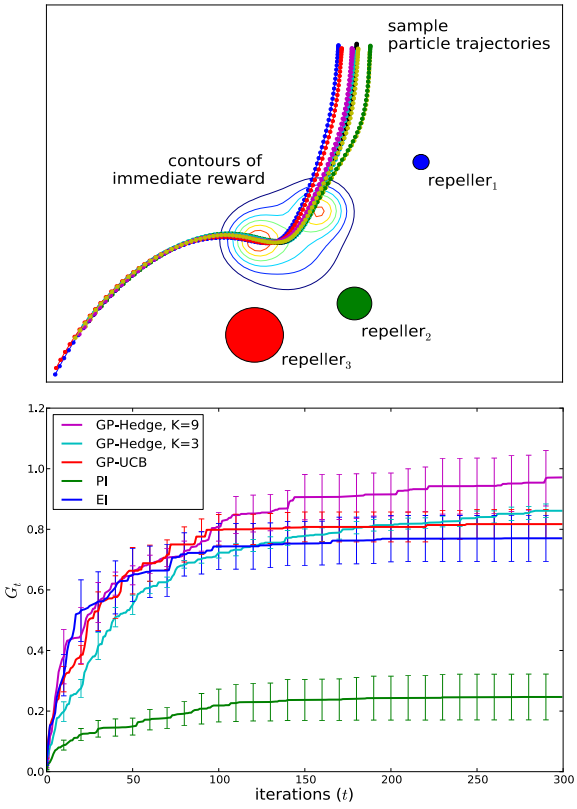


Figure 5.6: *Results of experiments on the repeller control problem. The upper plot displays 10 sample trajectories over 100 time-steps for a particular repeller configuration (not necessarily optimal). The lower plot shows the progress of each of the described Bayesian optimization methods on a similar model, averaged over 25 runs.*

by sampling a single trajectory and evaluating the sum over its immediate rewards.

Results for this optimization task are shown in Figure 5.6. As with the previous synthetic examples GP-Hedge outperforms each of its constituent methods on average. We further note the particularly poor performance of the PI acquisition function on this example, which in part we hypothesize is a result of plateaus in the resulting objective function. In particular PI has trouble exploring after it has "locked on" to a particular mode, a fact

that seems exacerbated when there are large regions with very little change in objective.

5.4 Conclusions and future work

Hedging strategies are a powerful tool in the design of acquisition functions for Bayesian optimization. In this paper we have shown that strategies that adaptively modify a portfolio of acquisition functions often perform substantially better—and almost never worse—than the best-performing individual acquisition function. Our experiments have shown that full-information strategies are able to outperform partial-information strategies in many situations. However, partial-information strategies can be beneficial in instances of high K or in situations where the acquisition functions provide very conflicting advice. Evaluating these tradeoffs is an interesting area of future research.

As noted earlier (and in §2.2.2) the cumulative regret provides a method for lower-bounding the convergence properties of these algorithms. Bounding GP-Hedge is left for future work. Bounds for the underlying GP-UCB algorithm have been shown [Srinivas *et al.*, 2010], as well as bounds for related but distinct bandit problems [Grunewalder *et al.*, 2010]. Ultimately we would also like to show similar bounds for the gap measure on GP-Hedge, but this result seems much more difficult.

Finally, while outside the scope of this work, we feel this portfolio-based approach could be used for model selection (§2.1.2), as well, possibly in conjunction with the cross-session model introduced in Chapter 4, and/or Bayesian Monte Carlo model integration [Osborne *et al.*, 2008].

Chapter 6

Applications: active preference learning with Bayesian optimization

In this chapter, we will present several applications that use interactive Bayesian optimization to find parameters for graphics and animation problems[1]. We also present results of experiments done with users, which demonstrate the effectiveness of interactive Bayesian optimization.

6.1 Related work

The work of Talton *et al.* [2009] is a notable recent example of a design gallery approach in the computer graphics literature. They introduce a collaborative system which uses data from a body of users to learn spatially-varying parameters, though their work is still quite distinct from ours. Their model is based on density estimation in high-dimensional spaces, whereas we are interested in optimizing individual user value model. Their approach is also intended as a novel interface to aid users who are unfamiliar with the system, while our approach is intended to work in conjunction with more

[1]This chapter is based on previously published work [Brochu *et al.*, 2007a; 2007b; 2010a]. See the Co-Authorship Statement for details on co-author contributions.

traditional slider manipulation approaches to finding parameters.

The gallery interface of [Marks *et al.*, 1997] is perhaps the best known assistance tool for animators. It is a browsing tool where a set of animations is displayed on a 2D layout using multi-dimensional scaling. It uses heuristics to find the set of input parameters to be used in the generation of the display. We depart from this heuristic treatment and instead present a principled probabilistic decision making approach to model the design process.

Psychoperceptual preference elicitation has been previously done in graphics in the context of image-based rendering [Kuang *et al.*, 2004], as well as evaluation of tone-mapping operators for high dynamic range (HDR) images [Ledda *et al.*, 2005] and displays [Akyüz *et al.*, 2007]. Ledda *et al.* used preference to conduct psychoperceptual experiments to evaluate tone mapping operators. Participants were presented with pairs of images and asked to indicate which they thought most closely resembled a reference scene. This approach is very similar in spirit to our system, though it uses a finite set of data and does not interactively select pairs.

In this chapter, we detail the applications we have developed to date using interactive Bayesian optimization: an interactive smoke simulation (§6.2), a Bidirectional Radial Basis Function gallery tool (§6.3) and a procedural fluid animation generator (§6.4). On the latter two, most complex applications, we detail the results of user studies we have conducted to evaluate its use as a tool.

6.2 Active preferences for 2D smoke simulation

The initial motivation for this work was an animation tool based on Jos Stam's smoke simulation [Stam, 2003]. We present it here as a motivating example only—we did not use study it experimentally. This is a 2D simulation, which is not necessarily intended to be physically accurate, but rather to generate smoke effects that look "realistic". It uses the familiar Navier-Stokes equations[2] and a grid of samples on a density field. At each time

[2]We use here the "standard" symbols for Navier-Stokes, not to be confused with symbols used elsewhere in this thesis.

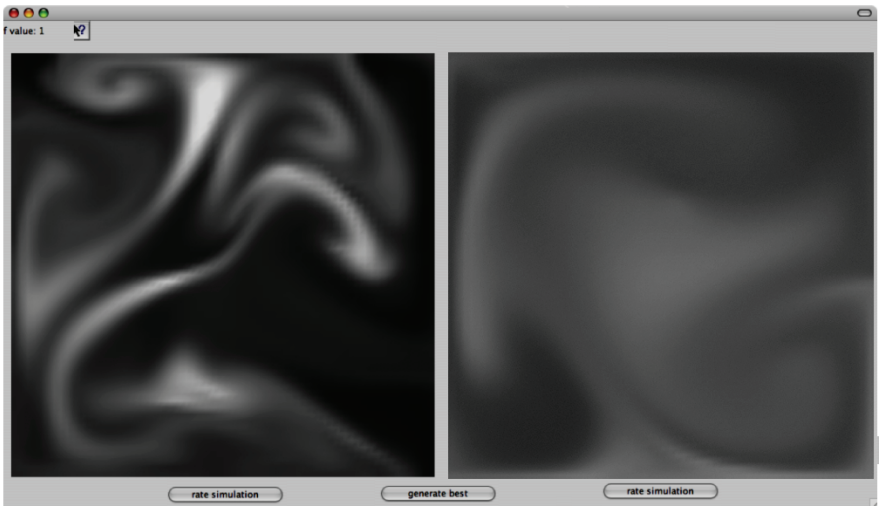


Figure 6.1: *An example of the interactive Bayesian optimization smoke simulation comparison tool (§6.2). The simulation engine is interactive and real-time, but controlling the simulation environment requires the setting of five interacting parameters. Our system allows a user with no prior experience to find the desired simulation parameters by generating simulations for the user to compare. At any point, the user can also ask the system to predict the current estimated best simulation parameters based on his or her feedback.*

step, the density field is updated by applying a velocity field (representing air flow or other interference, also sampled on a grid), uniform diffusion, and additional sources of density (that is, smoke added to the system from a source point).

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f} \\ \frac{\partial \varrho}{\partial t} &= -(\mathbf{u} \cdot \nabla) + \kappa \nabla^2 \varrho + S \end{aligned}$$

The simulation has a vector $\mathbf{x}$ of five parameters, all of which take positive real values: *viscosity*, *diffusion*, *time step*, *source* (the amount of smoke generated in the simulation when the user clicks the simulation window) and *force* (the amount of disruption the user's mouse click creates in the

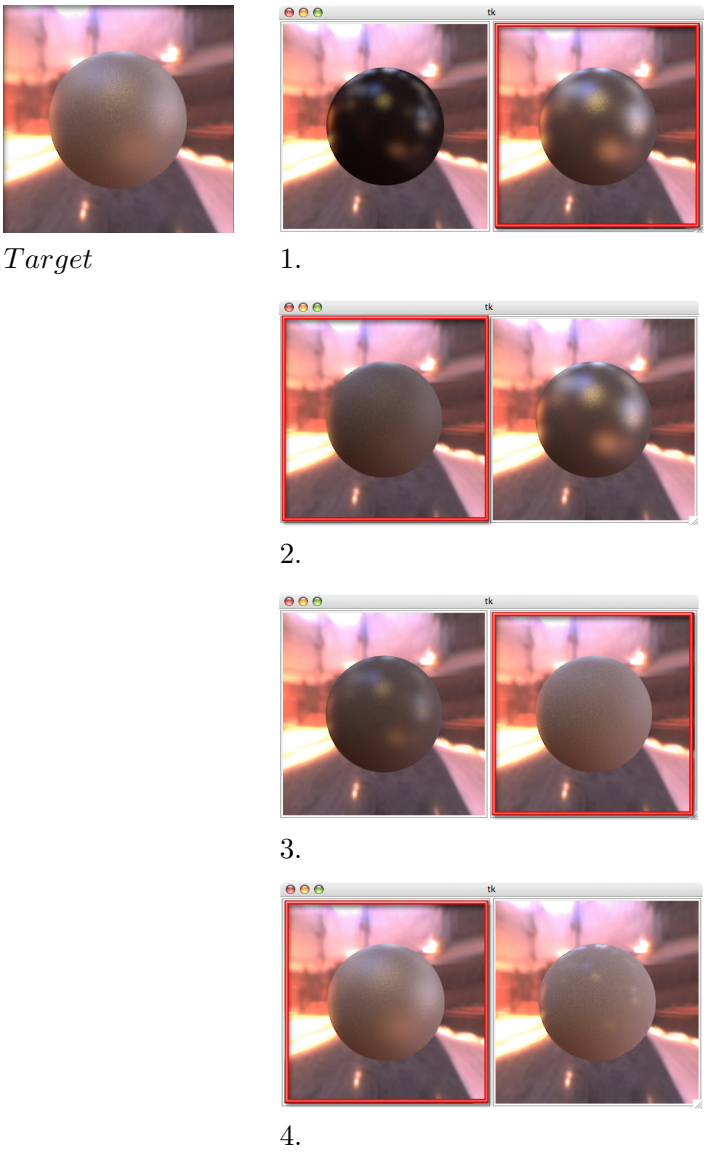
simulation). Clearly, some of these are dependent (*force* and *time step*, for example). The system is extremely robust, and will not blow up even when given bad values, but large parts of the parameter space generate simulations that look disappointing, at best. Furthermore, this real-time simulation can generate a variety of legitimate smoke, from wispy curlicues like cigarette smoke to thick, viscous smoke resembling factory exhaust.

Our simulation-rating environment is a GUI application shown in Figure 6.1. The user-feedback portion of our application allows the user to interact with two simulation environments and indicate preference using whatever desired value model they have in mind, typically a specific smoke effect. By clicking mouse buttons, the user can add and move smoke within either simulation window. A window button allows the user to indicate preference, at which point our algorithm is run again and selects two more simulations to show the user. At any time, the user can click a button in the GUI to get the "best" simulation, which is computed by running DIRECT to find an $\mathbf{x}$ (set of parameters) that maximizes the value rather than $\mathrm{EI}(\cdot)$.

If this is not the final result the user sought, it can be rated as usual, and another iteration of our algorithm is performed.

6.3 Interactive Bayesian optimization for material design

Properly modelling the appearance of a material is a necessary component of realistic image synthesis. The appearance of a material is formalized by the notion of the Bidirectional Reflectance Distribution Function (BRDF). In computer graphics, BRDFs are most often specified using various analytical models. Analytical models that are of interest to realistic image synthesis are the ones that observe the physical laws of reciprocity and energy conservation while typically also exhibiting shadowing, masking and Fresnel reflectance phenomenon. Realistic models are therefore fairly complex with many parameters that need to be adjusted by the designer for the proper material appearance. Unfortunately these parameters can interact in non-intuitive ways, and small adjustments to certain settings may result in



Target 1. 2. 3. 4.

Figure 6.2: *A shorter-than-average but otherwise typical run of the BRDF preference gallery tool (§6.3). At each (numbered) iteration, the user is provided with two images generated with parameter instances and indicates the one they think most resembles the target image (top-left) they are looking for. The red boxes indicate the user's selections at each iteration.*

non-uniform changes in the appearance. This can make the material design process quite difficult for the artist end user, who is not expected to be an expert in the field. To alleviate this problem, Ngan *et al.* [2006] presented an interface for navigation in a perceptually uniform BRDF space based on a metric derived from user studies. However, this is still somewhat constraining as the user has to develop an understanding of the various aspects of material appearance such as varying degrees of diffuseness, glossiness, specularity, Fresnel effects and/or anisotropy in order to navigate such an interface. An artist often knows the look that she desires for a particular application without necessarily being interested in understanding the various subtleties of reflection. We attempt to deal with this using a preference gallery approach, in which users are simply required to view two or more images rendered with different material properties and indicate which they prefer, in an iterative process.

We use the interactive Bayesian optimization preference model described in §3.2 on an example gallery application for helping users find a BRDF. For the purposes of this example, we limit ourselves to isotropic materials and ignore wavelength dependent effects in reflection. The gallery uses the Ashikhmin-Shirley Phong model [Ashikhmin *et al.*, 2000] for the BRDFs, which has been shown to be well-suited for representing real materials [Ngan *et al.*, 2005]. The BRDFs are rendered on a sphere under high-frequency natural illumination as this has been shown to be the desired setting for human perception [Fleming *et al.*, 2001]. Our gallery demonstration presents the user with two BRDF images at a time. The GP model is updated after each preference is indicated. We use parameters of real measured materials from the MERL database [Ngan *et al.*, 2005] for seeding the parameter space, but can draw arbitrary parameters after that.

By querying the user with a paired comparison, one can estimate statistics of the value function at the query point, but only at considerable expense. Thus, we wish to make sure that the samples we do draw will generate the maximum possible improvement.

Our method for achieving this goal iterates between the following steps:

1. **Present the user with a new set of instances and record preferences**

from the user: Augment the training set of paired choices with the new user data.

2. **Infer the value function**: Here we use a Thurstone-Mosteller model with Gaussian processes [Chu and Ghahramani, 2005b]. See §3.2 for details. Note that in this application, the value function is the objective of Bayesian optimization. We will use the terms interchangeably.

3. **Optimize the acquisition function of the value to obtain the query points for the next gallery**: Methods for selecting a set of instances are described in §3.3.

6.3.1 User study

To evaluate the performance of our application, we have run a simple user study in which the generated images are restricted to a subset of 38 materials from the MERL database that we deemed to be representative of the appearance space of the measured materials. The user is given the task of finding a single randomly-selected image from that set by indicating preferences. Figure 6.2 shows a typical user run, where we ask the user to use the preference gallery to find a provided target image. At each step, the user need only indicate the image they think looks most like the target. This would, of course, be an unrealistic scenario if we were to be evaluating the application from an HCI stance, but here we limit our attention to the model, where we are interested here in demonstrating that with human users maximizing value is preferable to learning the entire latent function.

Using five subjects, we compared 50 trials using the expected improvement function (§2.2.1) to select the images for the gallery ($\max_{\text{EI}}$), 50 trials using maximum variance ($\max_{\sigma}$), and 50 trials using samples selected using a randomized Latin hypercube algorithm. In each case, one of the gallery images was the incumbent and the other was selected by the algorithm. The algorithm type for each trial was randomly selected by the computer and neither the experimenter nor the subjects knew which of the three algorithms was selecting the images. The results are shown in Table 6.1. N is the number clicks required of the user to find the target image. Clearly $\max_{\text{EI}}$ dominates, with a mean N less than half that of the competing al-

algorithm	trials	N (mean $\pm$ std)
random	50	18.40 $\pm$ 7.87
$\max_\sigma$	50	17.87 $\pm$ 8.60
$\max_{\text{EI}}$	50	8.56 $\pm$ 5.23

Table 6.1: *Results of the user study on the BRDF gallery (§6.3.1).*

gorithms. Interestingly, selecting images using maximum variance does not perform much better than random. We suspect that this is because $\max_\sigma$ has a tendency to select images from the corners of the parameter space, which adds limited information to the other images, whereas the random (Latin hypercube) algorithm at least guarantees that the selected images fill the space.

6.4 Procedural fluid animation

Procedural methods for generating animation have long been used in visual effects and games studios due to their efficiency and artist controllability. However, this control comes with a cost: a set of often unintuitive parameters confronts the user of a procedural animation system. The desired end result is often identifiable by the user, but these parameters must be tuned in a tedious trial-and-error process.

For example, realistic animation of smoke can be achieved by driving a particle system through a simple combination of vortex rings and curl noise [Bridson *et al.*, 2007]. However even these two relatively simple procedural methods are influenced by several parameters: the velocity, radius and magnitude of the vortex rings, and the length scale and magnitude of the curl noise. Adding more procedural "flow primitives", such as uniform and vortical flows, sources and sinks [Sims, 1990; Wejchert and Haumann, 1991], turbulent wind [Stam and Fiume, 1993], vortex particles [Selle *et al.*, 2005], and vortex filaments [Angelidis and Neyret, 2005] can produce a wider variety of animations, but each of these primitives carries its own set of associated parameters. These parameters can interact in subtle and non-intuitive ways, and small adjustments to certain settings may result in

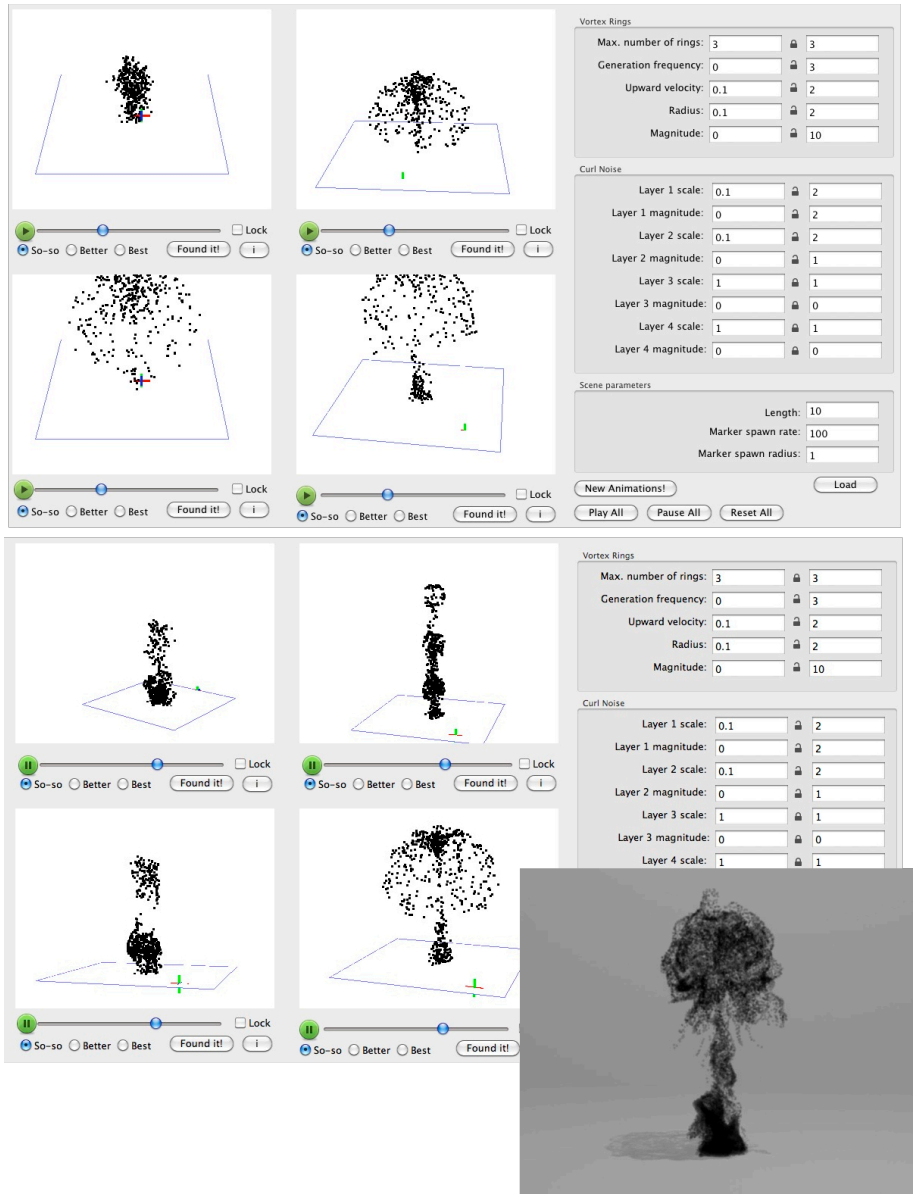


Figure 6.3: *The animation gallery in action. The upper and middle images show intermediate stages of the animation gallery. The inset image in the lower right is a frame from the final animation, generated after the artists has found the desired parameter settings.*

non-uniform changes in the appearance.

We apply the Bayesian optimization model as the learning engine for our procedural animation design tool. Our contribution to the design problem is a "gallery" approach in which users can view several animations generated from multiple parameters, and provide feedback in the form of real-valued ratings indicating how close an animation is to what they are looking for. In practice, the first few examples presented to the user will be points of high uncertainty, since little of the space is explored (that is, the model is very uncertain about the user's value criteria). Later galleries include more examples of high predicted value, as a model of the user's interest is learned.

6.4.1 Procedural animation

We produce smoke animation by driving a set of passive marker particles through a procedurally-generated velocity field. This velocity field is generated by taking the curl of a (vector-valued) potential function, which automatically ensures that the resulting velocity field is divergence-free, an important characteristic of fluid motion. There are two main components to this potential field, which we linearly combine: the contribution due to a set of vortex rings, and a spatially varying noise function.

The potential function of a vortex ring perpendicular to the y axis with centre $\mathbf{u}$, radius R, at a point in space $\mathbf{z}$ is given by

$$\boldsymbol{\psi}_v(\mathbf{z}) = \frac{1}{(R-D)^2 + 2RD} < \mathbf{z}_3, 0, -\mathbf{z}_1 >,$$

where $D = \|\mathbf{z} - \mathbf{u}\|$. The potential function associated with curl noise [Bridson *et al.*, 2007] is a spatially and temporally continuous noise function $\boldsymbol{\psi}_n(\mathbf{z}) = g(\mathbf{z}, L)$, where L is the length scale of noise.

$$\boldsymbol{\psi}_g(\mathbf{z}) = g(\mathbf{z}, L).$$

Our examples use Flow Noise [Perlin and Neyret, 2001] for this function. The velocity field is then the curl of the linear combination of these two

potential fields:

$$\mathbf{v}(\mathbf{z}) = \nabla \times (A\boldsymbol{\psi}_v(\mathbf{z}) + B\boldsymbol{\psi}_n(\mathbf{z})).$$

This simple model results in at least four parameters which must be tuned: the radius of a vortex ring, R, the length-scale of the noise function, L, and the relative strengths of each potential function, $A, B \in \mathbb{R}^+$. Additionally, our examples use vortex rings which move upward with some velocity, and are generated at the origin with some frequency, resulting in two additional parameters per ring. We model a total of four distinct curl noise layers, for an additional 8 parameters, though the use of the curl noise layers is not required for all animations, and they can be disabled by setting both parameters to 0. Since this method is procedural, and not a simulation, the variety of animations capable of production is fundamentally limited, though still quite large. Further parameters which could be added to the system include the spawn rate of marker particles, orientation of vortex rings, and the time derivative of any of the parameters mentioned above.

6.4.2 Gallery

The gallery interface (Figure 6.3) is our user-facing parameter-optimization tool. The user has the option of setting any or all of the parameters manually, either to a fixed value, or to a range of values. Otherwise, they may leave them at the default range of acceptable parameters. Fixing or setting the ranges of parameters directly sets the bounds of the optimization of the EI function.

Four animations are shown at a time. Based on the results of our gallery-selection experiments (§3.3), one of these is the previously-seen animation with the highest predicted value, and one is the point of maximum expected improvement. Remaining windows in the gallery are selected based on Schonlau's method [Schonlau, 1997] of simulating updates to the GP by iteratively maximizing the EI and updating the covariance matrix of the max EI function.

Note that at any stage, the user can set the parameters to a fixed value

or change the range. This permits the user to set up useful work-flows. For example, users can start with several free parameters and view examples until they find one similar to their target and fix most of the parameters, using the model to help set one or two "tricky" remaining values. This is a frequently-observed process in human decision making called "filtration" or "anchoring and adjustment". Alternatively, the user can adjust parameters until they reach a point where they are frustrated with one or more and then use the system to help find it. This is important, as Payne *et al.* [1993] demonstrate that users will change decision-making strategies depending on previous experience and information extracted and the task at hand. Forcing users into a particular mode can result in poor performance. In any case, the goal is not to remove or restrict the process of manually setting parameters, but to augment it.

Animations are generated using the procedural system described above, during a non-interactive "animation" phase. At each frame of the animation, the flow primitives are updated, and new ones are spawned if necessary. New particles are spawned at a source, advected according to the set of flow primitives, and all particle positions for a given frame are written to disk. The animation can then be previewed in an OpenGL window by streaming the particle data from disk.

After each run of the application, the final data vector $\mathcal{D}_{1:t}$ for the run is logged and the RBF parameters and distributions of the hyperparameters $(a, b, \boldsymbol{\theta}, \sigma^2_{\text{noise}})$ are updated (§4.2). The user has the option to skip this step if they do not wish to have the results logged.

The application uses the hyperparameters and RBF network mean function described above, trained on all available data from previous users. The hyperparameter updates are computed in the background or at an appropriate time when the user's computer has free cycles.

6.4.3 Experiment: hyperparameter learning

One of the main innovations of our work is the particle-filter updates of the hyperparameters. Particle filters are known to be an effective way of

modelling a distribution, but to our knowledge the concept of using them as a consistent model of hyperparameters across GPs trained with different data is novel. We need to confirm that the method will work well on our problem.

To track the fit of the particle filter to the model, we used the gallery to test one parameter at a time, with several users. Each user was given the task of using the gallery to find a target animation, which they were shown. The gallery was used with all the dimensions except one set to the target animation parameters. Users had to find the remaining parameter setting by rating generated animations using the gallery tool. After each run, the particle filters were updated for the noise hyperparameter and the kernel width hyperparameter of the free parameter. Figure 6.4 shows the results of the running the experiment on the σ^2_{noise} and $\boldsymbol{\theta}$ hyperparameters over 20 runs with 4 users. It is interesting to see that it takes only a few runs for the estimate to move from the initial value to a fairly consistent "region" of values. While the hyperparameters fluctuate a little with different users and use cases, the change is slow after the first few trials. This suggests that in the hyperparameter-learning scenario, particle filters are both quick to learn the hyperparameters and robust to noise.

6.4.4 Gallery interface performance

The experiments of Chapter 4 show clearly that particle filters and RBF models for the GP mean result in significant improvements in Bayesian optimization of known mathematical functions. To test the performance of components of the system with human beings in the loop, we would like to simulate the task of an animator looking for a specific animation. The difficulty with measuring performance in these animation tasks is that we don't know the animator's precise intentions. That is, we don't know the objective function's global maximum. To overcome this, an expert generated a set of five distinct animations, for which the target parameters were known. Users were shown one of the target animations, picked at random, and asked to find the corresponding target parameters using different variations of the

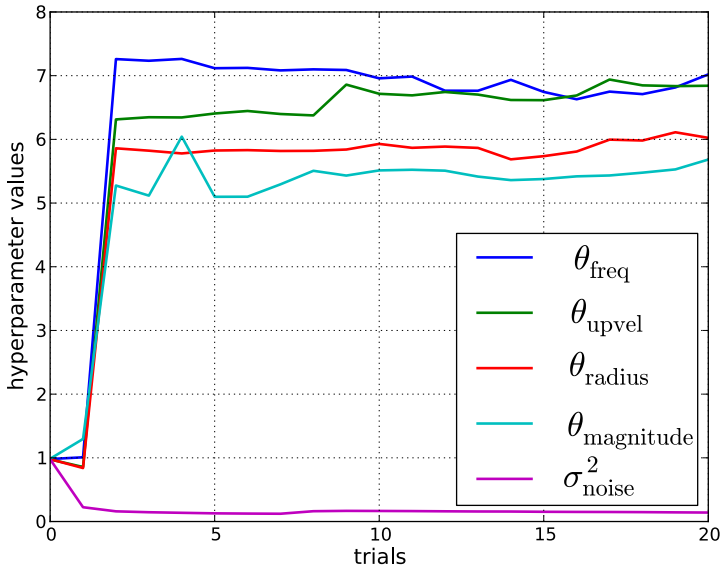


Figure 6.4: *Impact of learning hyperparameters from user data. Users were asked to perform tasks on the application and the hyperparameters were updated after each run using our particle filter (§4.2). The results show the evolution of 5 hyperparameter settings over 20 runs, involving 4 users. In all cases, the initial values, which are set to the common default of 1, were very quickly corrected and became increasingly stable thereafter, despite being used by different users for different tasks. This indicates that after an initial "burn-in" period where the particles are fairly scattered, they tend to congregate around the mean and are resistant to rapid change caused by simple variance from run to run. This suggests our learning model is both quick to learn and robust.*

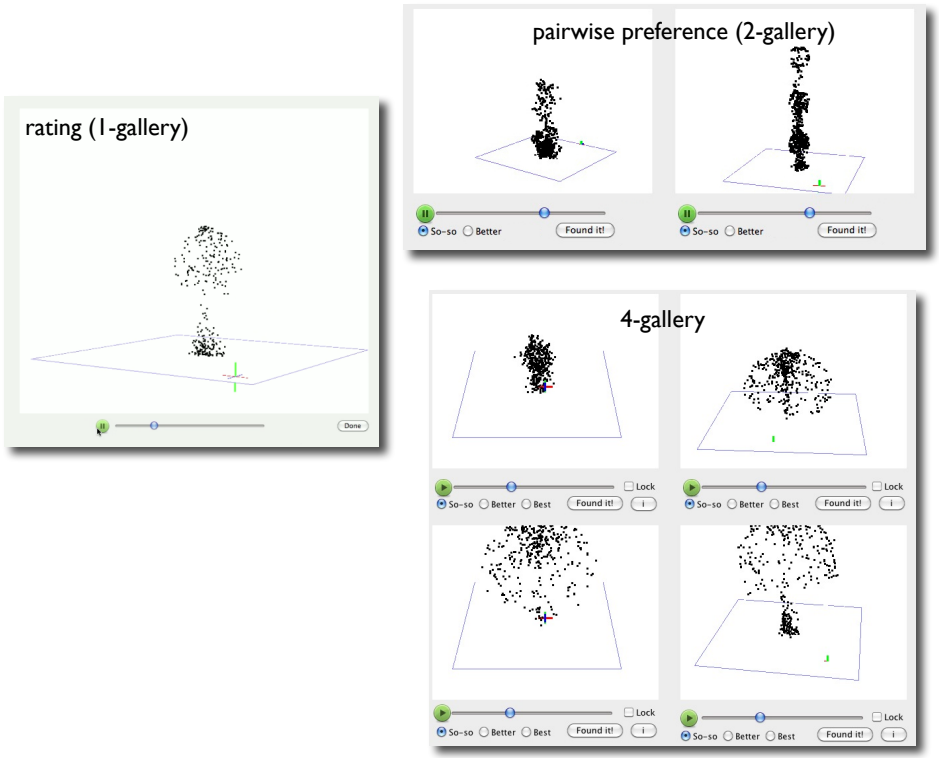


Figure 6.5: *Gallery interfaces used for the user studies described in §6.4.4, slightly edited to remove extraneous interface details. The 1-gallery functions by having the user directly rate animations with numerical values. The 2-gallery uses pairwise preferences: the user indicates which of the pair is preferred. The 4-gallery uses degrees of preference to allow richer input. All instances with the highest preference are preferred to all other instances, all instances with the middle preference are preferred to all instances with the lowest preference.*

interface (Figure 6.5). When users found an animation they felt was "close enough", they could select it. Subsequently, the application terminated and logged the number of iterations, distinct animations viewed, and the error. A GP zero mean function was used so as to avoid giving unfair advantage to the target animations. We tested the following scenarios:

- **expert-set hyperparameters (expertHP)** We tested interactive Bayesian optimization on our BRDF interface task in §6.3.1, and so we wish to compare our work to that as directly as possible, even though the applications are different. To do this, we (a) set the kernel hyperparameters $\boldsymbol{\theta}$ and σ^2_{noise} to expert-selected values; (b) restricted the gallery to 2 windows, one generated from $\text{argmax}_{\mathbf{x}} \text{EI}(\mathbf{x})$ and one from $\text{argmax}_{\mathbf{x}} \mu(\mathbf{x})$. To set θ we had an expert set an initial $\boldsymbol{\theta}^{\text{E}}$. We then ran 5 sessions with those hyperparameters, and maximized the likelihood to learn $\boldsymbol{\theta}^{\text{LL}}$ for each of those sessions. The final value for each hyperparameter θ_ℓ was either θ^{E}_ℓ or the median of θ^{LL}_ℓ, whichever seemed more reasonable to our expert.

- **particle filter hyperparameters (PFHP)** We compared the expert-set hyperparameter model to a pairwise model which is identical except that it uses hyperparameters learned with a particle filter trained on user sessions, as described in §4.1.

- **ratings** We repeated the rating experiment of §6.3.1 on our application to see if there was any undiscovered aspect of our problem that might make it easier to find the target by rating instances numerically. In this case, the user was shown a single animation at a time, corresponding to $\text{argmax}_{\mathbf{x}} \text{EI}(\mathbf{x})$ and asked to rate it with a "score" between 0 and 1.

- **4-gallery** Using the same hyperparameters as PFHP, for comparison purposes, we generated a gallery of 4 instances over which the user could enter preferences. Once all preferences were entered, they were added to the model, and a new set of gallery parameters selected.

As shown in Table 6.2 and Figure 6.6, the hyperparameters learned by PFHP result in a statistically significant ($p < 0.05$) improvement in both the

scenario	gallery size	param.	sessions	iterations	animations	error
expertHP	2	4	20	12.43 ± 4.45	22.66 ± 7.35	0.44 ± 0.30
PFHP	2	4	20	8.45 ± 2.81	**14.44 ± 5.03**	0.36 ± 0.34
ratings	1	4	20	28.35 ± 5.13	28.35 ± 5.13	0.31 ± 0.26
4-gallery	4	4	30	**7.57 ± 4.67**	24.85 ± 15.48	0.22 ± 0.17*
manual *(novice)*	1	12	3	35.33 ± 7.13	35.33 ± 7.13	2.02 ± 0.35
manual *(expert)*	1	12	5	28.40 ± 5.95	28.40 ± 5.95	0.91 ± 0.30*
4-gallery + manual	4	12	20	**5.38 ± 2.63**	**19.14 ± 7.02**	1.23 ± 0.74*

Table 6.2: *Results of experiments of §6.4.4 and 6.4.5 in which users were shown target animations and asked to find them using only specific methods.* **gallery size** *is the number of simultaneous animations instances in the interface used.* **param.** *is the number of free parameters the user was trying to find.* **sessions** *is the total number of sessions the scenario was run to collect data.* **iterations**, **animations** *and* **error** *are the mean and standard deviation of the number of interface iterations, the total number of different animations viewed, and the divergence of the user's selected parameter values from the target animation, respectively.* ***Bold results*** *indicate that for the indicated attribute, the scenario is the statistically-significant best-performing ($p < 0.05$). Asterisks indicate significance $0.05 < p < 0.2$. The upper part of the table illustrates some of the trade-offs involved (§6.4.4). The* 4-gallery *approach requires viewing more animations than pairwise, but is the most accurate, while direct rating performs poorly, as predicted. The pairwise preferences produced higher error, but required fewer total animation views. The lower part shows how a task (§6.4.5) that is nearly impossible for novice users with the manual interface becomes relatively easy when the manual interface is combined with the 4-gallery approach.*

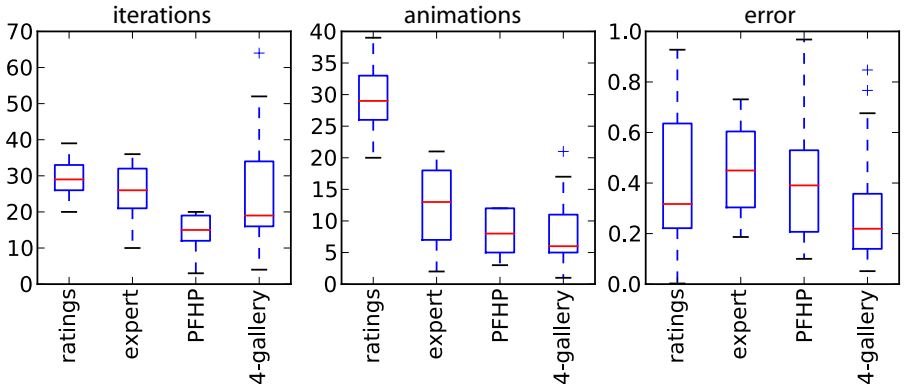


Figure 6.6: *Results of the experiments of §6.4.4. This is the same data as the upper part of Table 6.2, shown as boxplots to visualize the magnitudes of the mean performance and range of results.*

number of iterations and animations viewed over the expert hyperparameters (*expertHP*), and result in a slightly lower error. This represents a substantial savings in human effort, *and* eliminates the risk involved in having a human expert attempt the difficult but crucial task of setting the hyperparameters.

The *4-gallery* approach requires viewing more animations than pairwise, but is the most accurate (with significance $p < 0.2$), while direct rating performs poorly, as predicted. (Of course, it should be no surprise that the 4-gallery involved more instances, as at each iteration, 4 animations are generated, rather than 2.) The pairwise preferences produced higher error, but required fewer total animation views. Anecdotally, users reported difficulty in using the pairwise preferences due to the limited feedback available. We suspect this lead to the very large standard deviations on the error. Based on these results, the case for using a gallery of several instances is significant, but not as clear-cut as the case for learning the hyperparameters and mean function. If animating is cheap and the cognitive effort of rating is low (as in our application), a large gallery offers lower error. If the goal is to minimize the total number of animations generated and rated, using pairs is preferable. Numerically rating individual animations, however, suffers in both accuracy and the number of instances. Clearly, if we want to make the best use of user time, preferences are the most suitable choice.

6.4.5 Full application interface compared to parameter twiddling

Next, we test whether our "real" application—in which users can restrict ranges and fix parameters in addition to using a gallery of machine-selected animations—is more effective than more traditional "parameter twiddling" applications:

- **manual** We implemented a single-window GUI with no machine learning: the user sets all parameters manually. We distinguished between novices, who have no real understanding of what the parameters mean, and an expert, who is very familiar with the procedural animation system, and who also has a commanding knowledge of fluid animation in general.

- **4-gallery + manual** This is the full interface, in which all parameters start off unrestricted, and the user can indicate preference, and can also restrict the range of any parameter, including setting it to a single value. This directly controls the bounds of the EI maximization, so the next set of parameters will be in the indicated ranges.

The hyperparameters are again learned with a particle filter, and the prior mean function is set to zero. We found that we had to discontinue the initial manual-tweaking experiment. Our first subjects became so frustrated with trying to set the parameters that they frequently expressed a desire to give up. The numbers show the results when the experiment was terminated. We include the final results by means of comparison. We also included a small number of runs for an expert user who was familiar with the manual interface. These are shown as *manual (expert)* in Table 6.2. We consider the numbers for the manual tool very unreliable—the significant result is that for non-experts, using the gallery made completing the task feasible at all! With that caveat, though, we find it interesting that the expert trials with the manual interface were only slightly better than the non-expert users using the gallery-plus-manual interface.

We used interactive Bayesian optimization on the problem described in §6.3.1 so we wish to compare the procedural fluid gallery to that as directly as possible, even though the applications (BRDF parameters versus procedural fluid animation parameters) are different. To do this, we set up what we will refer to as the *expertHP* model, which uses the method described there. This consists of (a) setting the prior mean to be the zero function (i.e., $\boldsymbol{\beta} = 0$); (b) setting the kernel hyperparameters θ to expert-selected "reasonable" values and keeping them there; and (c) restricting the gallery to 2 windows, one generated from $\operatorname{argmax}_{\mathbf{x}} \mathrm{EI}(\mathbf{x})$ and one from $\operatorname{argmax}_{\mathbf{x}} \mu(\mathbf{x})$. To set θ we had an expert set an initial $\boldsymbol{\theta}^{\mathtt{E}}$. We then ran 5 sessions with those hyperparameters, and used the log likelihood method to learn $\boldsymbol{\theta}^{\mathtt{LL}}$ for each of those sessions. The final value for each hyperparameter θ_ℓ was either $\theta_\ell^{\mathtt{E}}$ or the median of $\theta_\ell^{\mathtt{LL}}$, whichever seemed more reasonable to our expert. We collected data using 4 users for a total of 20 sessions.

We then compared those results to 20 sessions in which the hyperparameters were trained using the particle filter method described in §4.1. We initially trained the hyperparameters using the results of the *expertHP* sessions. Note that in both scenarios, we use the zero function for the mean, so as not to make animations more likely simply because they happened to be in the pool of available targets. The results of our observations in §4.3 indicate this would be a sufficient number of sessions to learn a fairly stable model. The trained particle filters were then used for a second set of sessions, *PFHP*. The gallery here was still pairwise, and the user's task the same, but we used hyperparameters that had been automatically tuned on 30 sessions of the system. The results are shown in Table 6.2 and Figure 6.6. Our particle filter method reduced the mean number of animations viewed by almost half, with a comparable error. The mean number of iterations (pairs viewed, in this case) was also substantially reduced, though not by quite as much, as our method is more likely to show the user the "best" animations as a component of different pairs. Nevertheless, this represents a substantial savings in human effort.

mean f'n	iterations	Q1	Q2
zero	11.25 ± 3.60	6.64 ± 1.76	7.00 ± 1.41
trained	**6.50 ± 2.15**	**7.42 ± 1.10**	**7.36 ± 0.81**

Table 6.3: *Comparison of users using the system with a zero function as the mean, and with a mean trained on previous user data. Users were asked to think of an animation and use the system to try to find it. We measured the number of iterations before the user terminated and collected responses to a two-question survey. Q1 was "how close is your selected animation to what you had in mind". Q2 was "independent of how close it was to your target, how much did you like the selected animation". The trained mean function offers improvement in all metrics, but especially in reducing the number of iterations of the system.*

6.4.6 Discovery

To test the effect of learning the mean function (§4.1), we designed a "discovery" experiment, where a user has an idea of what they want, but no ground truth. This also allows us to assess the impact of learning the mean function in a realistic setting. In this experiment, users (familiar with the system, but non-experts) are simply asked to have a rough idea in mind and use the system to find an animation they are satisfied with. In the first 15 trials, we used the zero-mean function. In the second 15, we learned the mean function using data from the first trials. Users were not told which mean function they were using. At the end of each session, users were asked to answer, using a subjective scale of 1–10, the following questions: (Q1) "how close is your selected animation to what you had in mind?"; and (Q2) "independent of how close it was to your target, how much did you like the selected animation?". The goal of these two questions is to measure the effect the mean function had on helping the user find an instance, and also to determine whether the trained mean might cause the user to favour instances that are appealing but not what they were looking for. This could happen, for instance, if the mean function had too much influence, biasing the search toward animations it was trained over exploration.

The results are shown in Table 6.3. The most dramatic difference is in the number of iterations required for users to find a target—from an average

of 11.25 to just 6.5. Moreover, the higher responses to Q1 indicate that users were, indeed, finding what they were looking for, and were not just settling for instances suggested due to the mean function.

Chapter 7

Conclusion

This thesis has sought to show how Bayesian optimization with Gaussian processes can be applied to solve a real problem: helping animators and artists find parameters for procedural graphics and animation problems. To this end, we have identified several unique aspects of the problem and suggested solutions.

The first issue is that human beings are poor at rating psychoperceptual experiences with absolute magnitudes. This makes optimizing a scalar function directly a difficult proposition. However, humans excel at indicating preferences. In Chapter 3, we showed how a probit model can be used in a gallery application to elicit user feedback. This allows users to respond in a more natural way, using preferences, from which we infer a GP model.

Another problem with interactive applications is that typically very little data is collected in a user session, which makes setting hyperparameters and the mean function challenging. Learning hyperparameters in a single user session is impractical, but the nature of the task itself offers a solution: while each individual session with the application might involve different users looking for different animations, if we treat the hyperparameter-learning problem as one of state estimation in a hierarchical Bayesian model, we can use particle filters to make the process automatic. In Chapter 4, we show how this can be done. By only updating the distribution, the model is guided toward good settings without being overly restricted. The continuous

learning and updating ensures that the more the gallery application is used, the better it will become at meeting the needs of its users, even if those needs change over time.

Finally, determining the acquisition function for an unknown objective is a difficult task. In Chapter 5, we show that strategies that adaptively modify a portfolio of acquisition functions often perform substantially better—and almost never worse—than the best-performing individual acquisition function. Our experiments have shown that full-information strategies are able to outperform partial-information strategies in many situations.

In Chapter 6, we put our theoretical work into practice with a series of applications. We demonstrate the performance of applications in a set of user studies in a controlled environment. However, user expertise is a valuable resource, and if a user knows the ranges or exact value of a parameter, or if they wish to use conventional slider twiddling as part of their workflow, it is important that our tool not interfere. We present a framework in which our techniques can be used in conjunction with existing methods.

7.1 Discussion and advice to practitioners

Interactive Bayesian optimization is a powerful tool for machine learning, where the problem is often not acquiring data, but acquiring labels. In many ways, it is like conventional active learning, but instead of acquiring training data for classification or regression, it allows us to develop frameworks to efficiently solve novel kinds of learning problems such as those discussed in Chapters 3–6. It provides us with an efficient way to learn the solutions to problems, and to collect data, all within a Bayesian framework.

However, Bayesian optimization is also a fairly recent addition to the machine learning community, and not yet extensively studied on user applications. Here, we wish to describe some of the shortcomings we have experienced in our work with Bayesian optimization, both as caveats and as opportunities for other researchers.

A particular issue is that the design of the prior is absolutely critical to efficient Bayesian optimization. Gaussian processes are not always the

best or easiest solution, but even when they are, great care must be taken in the design of the kernel. In many cases, though, little is known about the objective function, and, of course, it is expensive to sample from (or we wouldn't need to use Bayesian optimization in the first place). The practical result is that in the absence of (expensive) data, either strong assumptions are made without certainty that they hold, or a weak prior must be used. It is also often unclear how to handle the trade-off between exploration and exploitation in the acquisition function. Too much exploration, and many iterations can go by without improvement. Too much exploitation leads to local maximization.

These problems are exacerbated as dimensionality is increased—more dimensions means more samples are required to cover the space, and more parameters and hyperparameters may need to be tuned, as well. In order to deal with this problem effectively, it may be necessary to do automatic feature selection, or assume independence and optimize each dimension individually.

Another limitation of Bayesian optimization is that the acquisition is currently both myopic and permits only a single sample per iteration. Looking forward to some horizon would be extremely valuable for reinforcement learning problems, as well as in trying to optimize within a known budget of future observations. Recent work [Garnett *et al.*, 2010b; Azimi *et al.*, 2011] has indicated very promising directions for this work to follow. Being able to efficiently select entire sets of samples to be labelled at each iteration would be a boon to design galleries and other batch-incremental problems.

Finally, there are many extensions that will need to be made to Bayesian optimization for particular applications—feature selection, time-varying models, censored data, heteroskedasticity, nonstationarity, non-Gaussian noise, *etc.* In many cases, these can be dealt with as extensions to the prior—in the case of Gaussian processes, for example, a rich body of literature exists in which such extensions have been proposed. However, these extensions need to take into account the adaptive and iterative nature of the optimization problem, which can vary from trivial to impossible.

Clearly, there is a lot of work to be done in Bayesian optimization, but

we feel that the doors it opens make it worthwhile. It is our hope that as Bayesian optimization proves itself useful in the machine learning domain, the community will embrace the fascinating new problems and applications it opens up.

Bibliography

[Akyüz *et al.*, 2007] A. O. Akyüz, R. Fleming, B. E. Riecke, E. Reinhard, and H. H. Bülthoff. Do HDR displays support LDR content?: a psychophysical evaluation. *ACM Transactions on Graphics*, 26(3):38, 2007.

[Angelidis and Neyret, 2005] A. Angelidis and F. Neyret. Simulation of smoke based on vortex filament primitives. In *ACM-SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2005.

[Ashikhmin *et al.*, 2000] M. Ashikhmin, S. Premośe, and P. Shirley. A microfacet-based BRDF generator. *ACM Transactions on Graphics*, pages 65–74, 2000.

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

[Audibert *et al.*, 2010] J. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. In *Proc. of the Conference on Learning Theory (COLT)*, 2010.

[Auer *et al.*, 1998] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. Technical Report NC2-TR-1998-025, NeuroCOLT2 Technical Report Series, 1998.

[Azimi *et al.*, 2011] J. Azimi, A. Fern, and X. Z. Fern. Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems 24*, 2011.

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

[Bertsekas and Tsitsiklis, 1996] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Betrò, 1991] B. Betrò. Bayesian methods in global optimization. *J. Global Optimization*, 1:1–14, 1991.

[Bishop, 2006] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

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

[Bridson *et al.*, 2007] R. Bridson, J. Houriham, and M. Nordenstam. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics*, 26(3):46, 2007.

[Brochu *et al.*, 2007a] E. Brochu, N. de Freitas, and A. Ghosh. Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems 21*, 2007.

[Brochu *et al.*, 2007b] E. Brochu, A. Ghosh, and N. de Freitas. Preference galleries for material design. In *ACM SIGGRAPH 2007 Posters*, page 105, 2007.

[Brochu *et al.*, 2009] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions. Technical Report TR-2009-023, University of British Columbia, Department of Computer Science, 2009.

[Brochu *et al.*, 2010a] E. Brochu, T. Brochu, and N. de Freitas. A Bayesian interactive optimization approach to procedural animation design. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*, 2010.

[Brochu *et al.*, 2010b] E. Brochu, M. Hoffman, and N. de Freitas. Hedging strategies for Bayesian optimization. eprint arXiv:1009.5419, arXiv.org, September 2010.

[Brochu *et al.*, 2011] E. Brochu, V. M. Cora, and N. de Freitas. *2008 Machine Learning Summer School*, volume 6368 of *LNCS/LNAI Tutorial Series*, chapter A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. Springer, 2011. To appear.

[Bubeck *et al.*, 2009] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory*, pages 23–37. Springer, 2009.

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

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

[Chaudhuri *et al.*, 2009] K. Chaudhuri, Y. Freund, and D. Hsu. A parameter-free hedging algorithm. In *Advances in Neural Information Processing Systems 23*, 2009.

[Chu and Ghahramani, 2005a] W. Chu and Z. Ghahramani. Extensions of Gaussian processes for ranking: semi-supervised and active learning. In *Learning to Rank workshop at NIPS-18*, 2005.

[Chu and Ghahramani, 2005b] W. Chu and Z. Ghahramani. Preference learning with Gaussian processes. In *Proc. 22nd International Conf. on Machine Learning*, 2005.

[Cora, 2008] V. M. Cora. Model-based active learning in hierarchical policies. Master's thesis, University of British Columbia, Vancouver, Canada, April 2008.

[Cox and John, 1992] D. D. Cox and S. John. A statistical method for global optimization. In *Proc. IEEE Conference on Systems, Man and Cybernetics*, volume 2, pages 1241–1246, 1992.

[Cox and John, 1997] D. D. Cox and S. John. SDO: A statistical method for global optimization. In M. N. Alexandrov and M. Y. Hussaini, editors, *Multidisciplinary Design Optimization: State of the Art*, pages 315–329. SIAM, 1997.

[Diggle and Ribeiro, 2007] P. J. Diggle and P. J. Ribeiro. *Model-Based Geostatistics*. Springer Series in Statistics. Springer, 2007.

[Diggle *et al.*, 1998] P. J. Diggle, J. A. Tawn, and R. A. Moyeed. Model-based geostatistics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47(3):299–350, 1998.

[Doucet *et al.*, 2001] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Statistics for Engineering and Information Science. Springer, 2001.

[Eidsvik *et al.*, 2009] J. Eidsvik, S. Martinao, and H. Rue. Approximate Bayesian inference in spatial generalized linear mixed models. *Scandinavian Journal of Statistics*, 36:1–22, 2009.

[Elder, 1992] J. F. Elder, IV. Global R^d optimization when probes are expensive: The GROPE algorithm. In *Proc. IEEE International Conference on Systems, Man and Cybernetics*, 1992.

[Élő, 1978] Á. Élő. *The Rating of Chess Players: Past and Present*. Arco Publishing, New York, 1978.

[Finkel, 2003] D. E. Finkel. *DIRECT Optimization Algorithm User Guide*. Center for Research in Scientific Computation, North Carolina State University, 2003.

[Fleming *et al.*, 2001] R. Fleming, R. Dror, and E. Adelson. How do humans determine reflectance properties under unknown illumination? In *CVPR Workshop on Identifying Objects Across Variations in Lighting*, 2001.

[Frean and Boyle, 2008] M. Frean and P. Boyle. Using Gaussian processes to optimize expensive functions. In W. Wobcke and M. Zhang, editors, *AI 2008: Advances in Artificial Intelligence*, volume 5360 of *Lecture Notes in Computer Science*, pages 258–267. Springer Berlin / Heidelberg, 2008.

[Garnett *et al.*, 2010a] R. Garnett, M. Osborne, S. Reece, A. Rogers, and S. Roberts. Sequential Bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 2010.

[Garnett *et al.*, 2010b] R. Garnett, M. Osborne, and S. Roberts. Bayesian optimization for sensor set selection. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 209–219. ACM, 2010.

[Genton, 2001] M. G. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2:299–312, 2001.

[Glickman and Jensen, 2005] M. E. Glickman and S. T. Jensen. Adaptive paired comparison design. *J. Statistical Planning and Inference*, 127:279–293, 2005.

[Goldberg *et al.*, 1998] P. W. Goldberg, C. K. I. Williamsn, and C. M. Bishop. Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems 10*, 1998.

[Grunewalder *et al.*, 2010] S. Grunewalder, J. Audibert, M. Opper, and J. Shawe-Taylor. Regret bounds for Gaussian process bandit problems. In *Proceedings of the Conference on Artificial Intelligence and Statistics*, 2010.

[Hastie *et al.*, 2009] T. Hastie, R. Tibrishani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, second edition, 2009.

[Herbrich and Graepel, 2006] R. Herbrich and T. Graepel. Trueskill: A Bayesian skill rating system. Technical Report MSR-TR-2006-80, Microsoft Research, June 2006.

[Herbrich *et al.*, 1998] R. Herbrich, T. Graepel, P. Bollman-Sdorra, and K. Obermayer. Learning preference relations for information retreival. Technical report, AAAI Technical Report, 1998.

[Hinton and Salakhutdinov, 2006] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[Hoffman *et al.*, 2009] M. Hoffman, H. Kück, N. de Freitas, and A. Doucet. New inference strategies for solving Markov decision processes using reversible jump MCMC. In *Uncertainty in Artificial Intelligence*, 2009.

[Holmes and Held, 2006] C. Holmes and L. Held. Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Analysis*, 1(1):145–168, 2006.

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

[Hutter *et al.*, 2007] F. Hutter, D. Babić, H. H. Hoos, and A. Hu. Boosting verification by automatic tuning of decision procedures. In *Proc. of Formal Methods in Computer Aided Design (FMCAD'07)*, 2007.

[Hutter *et al.*, 2009] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy. An experimental investigation of model-based parameter optimisation: SPO and beyond. In *Proc. GECCO'09*, 2009.

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

[Jones *et al.*, 1993] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *J. Optimization Theory and Apps*, 79(1):157–181, 1993.

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

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

[Kahneman and Tversky, 1979] D. Kahneman and A. Tversky. Prospect theory: an analysis of decision making under risk. *Econometrica*, 47:263–291, 1979.

[Kapoor *et al.*, 2007] A. Kapoor, K. Grauman, R. Urtasun, and T. Sarrell. Active learning with Gaussian processes for object categorization. In *Proc. International Conference on Computer Vision (ICCV)*, 2007.

[Kendall, 1975] M. Kendall. *Rank Correlation Methods*. Griffin Ltd, 1975.

[Kingsley, 2006] D. C. Kingsley. Preference uncertainty, preference refinement and paired comparison choice experiments. Working Paper 06-06, Center for Economic Analysis, University of Colorado at Boulder, 2006. Dept. of Economics, University of Colorado.

[Krause *et al.*, 2008] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *J. Machine Learning Research*, 9:235–284, 2008.

[Krige, 1951] D. G. Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *J. the Chemical, Metallurgical and Mining Soc. of South Africa*, 52(6), 1951.

[Kuang *et al.*, 2004] J. Kuang, H. Yamaguchi, G. Johnson, and M. Fairchild. Testing HDR image rendering algorithms. In *Proc. IS and T/SID 12th Color Imaging Conference*, 2004.

[Kushner and Yin, 1997] H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, 1997.

[Kushner, 1964] H. J. Kushner. A new method of locating the maximum of an arbitrary multipeak curve in the presence of noise. *J. Basic Engineering*, 86:97–106, 1964.

[Ledda *et al.*, 2005] P. Ledda, A. Chalmers, T. Troscianko, and H. Seetzen. Evaluation of tone mapping operators using a high dynamic range display. *ACM Transactions on Graphics*, 24(3):640–648, August 2005.

[Lewis and Gale, 1994] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proc. ACM SIGIR Conference on Research and Development in Information Retreival*, 1994.

[Liberti and Maculan, 2006] L. Liberti and N. Maculan, editors. *Global Optimization: From Theory to Implementation*. Springer Nonconvex Optimization and Its Applications. Springer, 2006.

[Lizotte *et al.*, 2007] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans. Automatic gait optimization with Gaussian process regression. In *IJCAI*, 2007.

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

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

[Mackay, 1992] D. J. C. Mackay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472, 1992.

[Marks *et al.*, 1997] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. *Computer Graphics*, 31, 1997.

[Martinez–Cantin *et al.*, 2006] R. Martinez–Cantin, N. de Freitas, and J. Castellanos. Analysis of particle methods for simultaneous robot localization and mapping and a new algorithm: Marginal-SLAM. In *Proc. IEEE International Conference on Robots and Automation*, 2006.

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

[Matérn, 1960] B. Matérn. *Spatial Variation*. Springer-Verlag, 2nd (1986) edition, 1960.

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

[McFadden, 1980] D. McFadden. Econometric models for probabilistic choice among products. *Journal of Business*, 53(3):13–29, 1980.

[McFadden, 2001] D. McFadden. Economic choices. *The American Economic Review*, 91:351–378, 2001.

[Močkus *et al.*, 1978] J. Močkus, V. Tiesis, and A. Žilinskas. *Toward Global Optimization*, volume 2, chapter The Application of Bayesian Methods for Seeking the Extremum, pages 117–128. Elsevier, 1978.

[Močkus, 1982] J. Močkus. The Bayesian approach to global optimization. In R. Drenick and F. Kozin, editors, *System Modeling and Optimization*, volume 38, pages 473–481. Springer Berlin / Heidelberg, 1982.

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

[Mongeau *et al.*, 1998] M. Mongeau, H. Karsenty, V. Rouzé, and J.-B. Hiriart-Urruty. Comparison of public-domain software for black-box global optimization. Technical Report LAO 98-01, Universite Paul Sabatier, Toulouse, France, 1998.

[Mosteller, 1951] F. Mosteller. Remarks on the method of paired comparisons: I. the least squares solution assuming equal standard deviations and equal correlations. *Psychometrika*, 16:3–9, 1951.

[Murray-Smith and Girard, 2001] R. Murray-Smith and A. Girard. Gaussian process priors with ARMA noise models. In *Irish Signals and Systems Conference*, 2001.

[Ngan *et al.*, 2005] A. Ngan, F. Durand, and W. Matusik. Experimental analysis of BRDF models. In *Proc. Eurographics Symposium on Rendering*, pages 117–226, 2005.

[Ngan *et al.*, 2006] A. Ngan, F. Durand, and W. Matusik. Image-driven navigation of analytical BRDF models. In T. Akenine-Möller and W. Heidrich, editors, *Eurographics Symposium on Rendering*, 2006.

[O'Hagan, 1978] A. O'Hagan. On curve fitting and optimal design for regression. *Journal of the Royal Statistical Society B*, 40:1–42, 1978.

[Osborne *et al.*, 2008] M. Osborne, S. Roberts, A. Rogers, S. Ramchurn, and N. Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 109–120. IEEE Computer Society, 2008.

[Osborne *et al.*, 2009] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimization. In *3rd International Conference on Learning and Intelligent Optimization (LION3)*, 2009.

[Osborne *et al.*, 2010] M. Osborne, R. Garnett, and S. Roberts. Active data selection for sensor networks with faults and changepoints. In *IEEE International Conference on Advanced Information Networking and Applications*, 2010.

[Osborne, 2010] M. Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimization and Quadrature*. PhD thesis, University of Oxford, 2010.

[Payne *et al.*, 1993] J. W. Payne, J. R. Bettman, and E. J. Johnson. *The Adaptive Decision Maker*. Cambridge University Press, 1993.

[Perlin and Neyret, 2001] K. Perlin and F. Neyret. Flow noise. In *ACM SIGGRAPH 2001 Technical Sketches and Applications*, page 187, Aug 2001.

[Poyiadjis *et al.*, 2005] G. Poyiadjis, A. Doucet, and S. S. Singh. Particle methods for optimal filter derivative: Application to parameter estimation. In *IEEE ICASSP*, pages 925–928, 2005.

[Press *et al.*, 2007] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

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

[Rasmussen, 2003] C. E. Rasmussen. Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals. In *Bayesian Statistics 7*, pages 651–659. Oxford University Press, 2003.

[Rue *et al.*, 2009] H. Rue, S. Martino, and N. Chopin. Approximate Bayesian inference for latent Gaussian process models using integrated Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71:319–392, 2009.

[Russo and Leclerc, 1991] J. E. Russo and F. Leclerc. Characteristics of successful information programs. *J. Social Issues*, 17:759–769, 1991.

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

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

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

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

[Selle *et al.*, 2005] A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Transactions on Graphics*, 24(3):910–914, 2005.

[Settles, 2010] B. Settles. Active learning literature survey. Computer Science Technical Report 1648, University of Wisconsin-Madison, January 2010.

[Siegel and Castellan, 1988] S. Siegel and N. J. Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 1988.

[Simon, 1978] H. A. Simon. Rationality as process and as product of thought. *American Economic Review*, 68:1–16, 1978.

[Sims, 1990] K. Sims. Particle animation and rendering using data parallel computation. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 405–413, New York, NY, USA, 1990. ACM.

[Srinivas *et al.*, 2010] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. International Conference on Machine Learning (ICML)*, 2010.

[Stam and Fiume, 1993] J. Stam and E. Fiume. Turbulent wind fields for gaseous phenomena. In *SIGGRAPH '93*, pages 369–376, 1993.

[Stam, 2003] J. Stam. Real-time fluid dynamics for games. In *Proceedings of the Game Developer Conference*, 2003.

[Stein, 1999] M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer, 1999.

[Stern, 1990] H. Stern. A continuum of paired comparison models. *Biometrika*, 77:265–273, 1990.

[Streltsov and Vakili, 1999] S. Streltsov and P. Vakili. A non-myopic utility function for statistical global optimization algorithms. *J. Global Optimization*, 14:283–298, 1999.

[Stuckman, 1988] B. Stuckman. A global search method for optimizing nonlinear systems. *IEEE Transactions on Systems, Man and Cybernetics*, 18(6):965–977, 1988.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Talton *et al.*, 2009] J. O. Talton, D. Gibson, L. Yang, P. Hanrahan, and V. Koltun. Exploratory modeling with collaborative design spaces. In *Proc. 2nd Annual ACM SIGGRAPH Conf. and Exhibition in Asia*, 2009.

[Thurstone, 1927] L. Thurstone. A law of comparative judgement. *Psychological Review*, 34:273–286, 1927.

[Törn and Žilinskas, 1989] A. Törn and A. Žilinskas. *Global Optimization*. Springer-Verlag, 1989.

[Train, 2003] K. Train. *Discrete Choice Methods with Simulation*. Cambridge University Press, 2003.

[Tversky and Kahneman, 1992] A. Tversky and D. Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. *J. Risk and Uncertainty*, 5:297–323, 1992.

[Vasquez and Bect, 2008] E. Vasquez and J. Bect. On the convergence of the expected improvement algorithm. Technical Report arXiv:0712.3744v2, arXiv.org, Feb 2008.

[Wejchert and Haumann, 1991] J. Wejchert and D. Haumann. Animation aerodynamics. In *SIGGRAPH '91*, pages 19–22, New York, NY, USA, 1991. ACM.

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

[Wong *et al.*, 1988] S. K. M. Wong, Y. Y. Yao, and P. Bollmann. Linear structure in information retrieval. In *ACM SIGIR Conference on Research and Development in Information Retreival*, 1988.

[Younes, 1989] L. Younes. Parameter estimation for imperfectly observed Gibbsian fields. *Prob. Theory and Rel. fields*, 82:625–645, 1989.

[Zhigljavsky and Žilinskas, 2008] A. Zhigljavsky and A. Žilinskas. *Stochastic Global Optimization*. Springer Optimization and Its Applications. Springer, 2008.

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

[Žilinskas, 1980] A. Žilinskas. *Lecture Notes in Control and Information Sciences*, chapter On the Use of Statistical Models of Multimodal Functions for the Construction of Optimization Algorithms. Number 23. Springer-Verlag, 1980.

Appendix A

Test functions

In this appendix, we describe the forms of the literature test functions used for experiments in Chapters 3, 4 and 5. Note that while we give the conventional minimization forms here, when running the experiments we maximize $-f(\mathbf{x})$.

Branin

Two-dimensional Branin function (Figure A.1). Has 3 global minima $f(\mathbf{x}) = 0.397887$, at $\mathbf{x} = [-\pi, 12.275], [\pi, 2.275], [9.42478, 2.475]$.

$$f(\mathbf{x}) = \left(x_2 - 1.275\left(\frac{x_1}{\pi}\right)^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

$$-5 \leq x_1 \leq 10$$
$$0 \leq x_2 \leq 15$$

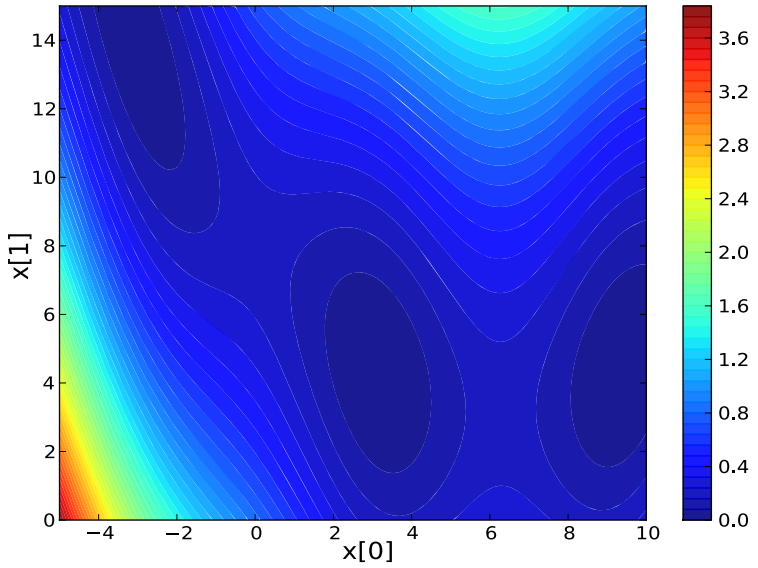


Figure A.1: *The Branin function.*

Hartman 3

Three-dimensional Hartman function. Minimum: $f([0.1, 0.5559, 0.8522]^T) = -3.8628$.

$$
\begin{aligned}
f(\mathbf{x}) &= -\sum_{i=1}^{4} C_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - B_{ij})^2\right) \\
A &= \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix} \\
B &= \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \end{bmatrix} \\
C &= [1 \quad 1.2 \quad 1 \quad 3]
\end{aligned}
$$

$$0 \leq x_i \leq 1, \qquad i = 1, \ldots, 3$$

Hartman 6

Six-dimensional Hartman function. Minimum $f(\mathbf{x}) = -3.3224$, when $\mathbf{x} = [0.2017, 0.15, 0.4769, 0.2753, 0.3117, 0.6573]^T$.

$$
\begin{aligned}
f(\mathbf{x}) &= -\sum_{i=1}^{4} C_i \exp\left(-\sum_{j=1}^{6} A_{ij}(x_j - B_{ij})^2\right) \\
A &= \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix} \\
B &= \begin{bmatrix} 0.1313 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix} \\
C &= [1 \quad 1.2 \quad 3 \quad 3.2]
\end{aligned}
$$

$$0 \leq x_i \leq 1, \qquad i = 1, \ldots, 6$$

Shekel 10

Four-dimensional function with 10 minima, and one global minimum $f(\mathbf{x}) = -10.1532$, when $\mathbf{x} = [4, 4, 4, 4]$.

$$
\begin{aligned}
f(\mathbf{x}) &= \sum_{i=1}^{10} \frac{1}{(\mathbf{x} - A_i)(\mathbf{x} - A_i)^T + C_i} \\
A &= \begin{bmatrix}
4 & 4 & 4 & 4 \\
1 & 1 & 1 & 1 \\
8 & 8 & 8 & 8 \\
6 & 6 & 6 & 6 \\
3 & 7 & 3 & 7 \\
2 & 9 & 2 & 9 \\
5 & 5 & 3 & 3 \\
8 & 1 & 8 & 1 \\
6 & 2 & 6 & 2 \\
7 & 3.6 & 7 & 3.6
\end{bmatrix} \\
C &= \begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.4 & 0.4 & 0.6 & 0.3 & 0.7 & 0.5 & 0.5 \end{bmatrix}
\end{aligned}
$$

$$
0 \leq x_i \leq 10, \qquad i = 1, \ldots, 4
$$