Chapter 3
Bayesian Optimization for Materials Design

Peter I. Frazier and Jialei Wang

Abstract We introduce Bayesian optimization, a technique developed for optimizing time-consuming engineering simulations and for fitting machine learning models on large datasets. Bayesian optimization guides the choice of experiments during materials design and discovery to find good material designs in as few experiments as possible. We focus on the case when materials designs are parameterized by a low-dimensional vector. Bayesian optimization is built on a statistical technique called Gaussian process regression, which allows predicting the performance of a new design based on previously tested designs. After providing a detailed introduction to Gaussian process regression, we describe two Bayesian optimization methods: expected improvement, for design problems with noise-free evaluations; and the knowledge-gradient method, which generalizes expected improvement and may be used in design problems with noisy evaluations. Both methods are derived using a value-of-information analysis, and enjoy one-step Bayes-optimality.

3.1 Introduction

In materials design and discovery, we face the problem of choosing the chemical structure, composition, or processing conditions of a material to meet design criteria. The traditional approach is to use iterative trial and error, in which we (1) choose some material design that we think will work well based on intuition, past experience, or theoretical knowledge; (2) synthesize and test the material in physical experiments; and (3) use what we learn from these experiments in choosing the material design to try next. This iterative process is repeated until some combination of success and exhaustion is achieved.

P.I. Frazier (✉) · J. Wang
School of Operations Research & Information Engineering, Cornell University,
Ithaca, NY14853, USA
e-mail: pf98@cornell.edu

T. Lookman et al. (eds.), *Information Science for Materials Discovery and Design*, Springer Series in Materials Science 225,
DOI 10.1007/978-3-319-23871-5_3

While trial and error has been extremely successful, we believe that mathematics and computation together promise to accelerate the pace of materials discovery, not by changing the fundamental iterative nature of materials design, but by improving the choices that we make about which material designs to test, and by improving our ability to learn from previous experimental results.

In this chapter, we describe a collection of mathematical techniques, based on Bayesian statistics and decision theory, for augmenting and enhancing the trial and error process. We focus on one class of techniques, called Bayesian optimization (BO), or Bayesian global optimization (BGO), which use machine learning to build a predictive model of the underlying relationship between the design parameters of a material and its properties, and then use decision theory to suggest which design or designs would be most valuable to try next. The most well-developed Bayesian optimization methods assume that (1) the material is described by a vector of continuous variables, as is the case, e.g., when choosing ratios of constituent compounds, or choosing a combination of temperature and pressure to use during manufacture; (2) we have a single measure of quality that we wish to make as large as possible; and (3) the constraints on feasible materials designs are all known, so that any unknown constraints are incorporated into the quality measure. There is also a smaller body of work on problems that go beyond these assumptions, either by considering discrete design decisions (such as small molecule design), multiple competing objectives, or by explicitly allowing unknown constraints.

Bayesian optimization was pioneered by [1], with early development through the 1970s and 1980s by Mockus and Zilinskas [2, 3]. Development in the 1990s was marked by the popularization of Bayesian optimization by Jones, Schonlau, and Welch, who, building on previous work by Mockus, introduced the Efficient Global Optimization (EGO) method [4]. This method became quite popular and well-known in engineering, where it has been adopted for design applications involving time-consuming computer experiments, within a broader set of methods designed for optimization of expensive functions [5]. In the 2000s, development of Bayesian optimization continued in statistics and engineering, and the 2010s have seen additional development from the machine learning community, where Bayesian optimization is used for tuning hyperparameters of computationally expensive machine learning models [6]. Other introductions to Bayesian optimization may be found in the tutorial article [7] and textbooks [8, 9], and an overview of the history of the field may be found in [10].

We begin in Sect. 3.2 by introducing the precise problem considered by Bayesian Optimization. We then describe in Sect. 3.3 the predictive technique used by Bayesian Optimization, which is called Gaussian Process (GP) regression. We then show, in Sect. 3.4, how Bayesian Optimization recommends which experiments to perform. In Sect. 3.5 we provide an overview of software packages, both freely available and commercial, that implement the Bayesian Optimization methods described in this chapter. We offer closing remarks in Sect. 3.6.

3.2 Bayesian Optimization

Bayesian optimization considers materials designs parameterized by a d-dimensional vector x. We suppose that the space of materials designs in which x takes values is a known set $A \subseteq \mathbb{R}^d$.

For example, $x = (x(1), \ldots, x(d))$ could give the ratio of each of d different constituents mixed together to create some aggregate material. In this case, we would choose A to be the set $A = \{x : \sum_{i=1}^{d} x(i) = 1\}$. As another example, setting $d = 2$, $x = (x(1), x(2))$ could give the temperature $(x(1))$ and pressure $(x(2))$ used in material processing. In this case, we would choose A to be the rectangle bounded by the experimental setup's minimum and maximal achievable temperature on one axis, $T_{\min}$ and $T_{\max}$, and the minimum and maximum achievable pressure on the other. As a final example, we could let $x = (x(1), \ldots, x(d)$ be the temperatures used in some annealing schedule, assumed to be decreasing over time. In this case, we would set A to be the set $\{x : T_{\max} \geq x(1) \geq \cdots \geq x(d) \geq T_{\min}\}$.

Let $f(x)$ be the quality of the material with design parameter x. The function f is unknown, and observing $f(x)$ requires synthesizing material design x and observing its quality in a physical experiment. We would like to find a design x for which $f(x)$ is large. That is, we would like to solve

$$\max_{x \in A} f(x). \tag{3.1}$$

This is challenging because evaluating $f(x)$ is typically expensive and time-consuming. While the time and expense depends on the setting, synthesizing and testing a new material design could easily take days or weeks of effort and thousands of dollars of materials.

In Bayesian optimization, we use mathematics to build a predictive model for the function f based on observations of previous materials designs, and then use this predictive model to recommend a materials design that would be most valuable to test next. We first describe this predictive model in Sect. 3.3, which is performed using a machine learning technique called Gaussian process regression. We then describe, in Sect. 3.4, how this predictive model is used to recommend which design to test next.

3.3 Gaussian Process Regression

The predictive piece of Bayesian optimization is based on a machine learning technique called Gaussian process regression. This technique is a Bayesian version of a frequentist technique called kriging, introduced in the geostatistics literature by South-African mining engineer Daniel Krige [11], and popularized later by Matheron and colleagues [12], as described in [13]. A modern monograph on

Gaussian process regression is [14], and a list of software implementing Gaussian process regression may be found at [15].

In Gaussian process regression, we seek to predict $f(x)$ based on observations at previously evaluated points, call them $x_1, \ldots, x_n$. We first treat the case where $f(x)$ can be observed exactly, without noise, and then later treat noise in Sect. 3.3.5. In this noise-free case, our observations are $y_i = f(x_i)$ for $i = 1, \ldots, n$.

Gaussian process regression is a Bayesian statistical method, and in Bayesian statistics we perform inference by placing a so-called *prior probability distribution* on unknown quantities of interest. The prior probability distribution is often called, more simply, the *prior distribution* or, even more simply, the *prior*. This prior distribution is meant to encode our intuition or domain expertise regarding which values for the unknown quantity of interest are most likely. We then use Bayes rule, together with any data observed, to calculate a *posterior probability distribution* on these unknowns. For a broader introduction to Bayesian statistics, see the textbook [16] or the research monograph [17].

In Gaussian process regression, if we wish to predict the value of f at a single candidate point x^*, it is sufficient to consider our unknowns to be the values of f at the previously evaluated points, $x_1, \ldots, x_n$, and the new point x^* at which we wish to predict. That is, we take our unknown quantity of interest to be the vector $(f(x_1), \ldots, f(x_n), f(x^*))$. We then take our data, which is $f(x_1), \ldots, f(x_n)$, and use Bayes rule to calculate a posterior probability distribution on the full vector of interest, $(f(x_1), \ldots, f(x_n), f(x^*))$, or, more simply, just on $f(x^*)$.

To calculate the posterior, we must first specify the prior, which Gaussian process regression assumes to be multivariate normal. It calculates the mean vector of this multivariate normal prior distribution using a function, called the *mean function* and written here as $\mu_0(\cdot)$, which takes a single x as an argument. It applies this mean function to each of the points $x_1, \ldots, x_n, x^*$ to create an $n+1$-dimensional column vector. Gaussian process regression creates the covariance matrix of the multivariate normal prior distribution using another function, called the *covariance function* or *covariance kernel* and written here as $\Sigma_0(\cdot, \cdot)$, which takes a pair of points x, x' as arguments. It applies this covariance function to every pair of points in $x_1, \ldots, x_n, x$ to create an $(n+1) \times (n+1)$ matrix.

Thus, Gaussian process regression sets the prior probability distribution to,

$$\begin{bmatrix} f(x_1) \\ \cdots \\ f(x_n) \\ f(x^*) \end{bmatrix} \sim \text{Normal}\left(\begin{bmatrix} \mu_0(x_1) \\ \cdots \\ \mu_0(x_n) \\ \mu_0(x^*) \end{bmatrix}, \begin{bmatrix} \Sigma_0(x_1, x_1) & \cdots & \Sigma_0(x_1, x_n) & \Sigma_0(x_1, x^*) \\ \vdots & \ddots & \vdots & \vdots \\ \Sigma_0(x_n, x_1) & \cdots & \Sigma_0(x_n, x_n) & \Sigma_0(x_n, x^*) \\ \Sigma_0(x^*, x_1) & \cdots & \Sigma_0(x^*, x_n) & \Sigma_0(x^*, x^*) \end{bmatrix} \right) \tag{3.2}$$

The subscript "0" in μ_0 and Σ_0 indicate that these functions are relevant to the prior distribution, before any data has been collected.

We now discuss how the mean and covariance functions are chosen, focusing on the covariance function first because it tends to be more important in getting good results from Gaussian process regression.

3.3.1 Choice of Covariance Function

In choosing the covariance function $\Sigma_0(\cdot, \cdot)$, we wish to satisfy two requirements.

The first is that it should encode the belief that points x and x' near each other tend to have more similar values for $f(x)$ and $f(x')$. To accomplish this, we want the covariance matrix in (3.2) to have entries that are larger for pairs of points that are closer together, and closer to 0 for pairs of points that are further apart.

The second is that the covariance function should always produce positive semi-definite covariance matrices in the multivariate normal prior. That is, if Σ is the covariance matrix in (3.2), then we require that $a^T \Sigma a \geq 0$ for all column vectors a (where a is assumed to have the appropriate length, $n + 1$). This requirement is necessary to ensure that the multivariate normal prior distribution is a well-defined probability distribution, because if θ is multivariate normal with mean vector μ and covariance matrix Σ, then the variance of $a \cdot \theta$ is $a^T \Sigma a$, and we require variances to be non-negative.

Several covariance functions satisfy these two requirements. The most commonly used is called the *squared exponential*, or Gaussian kernel, and is given by,

$$\Sigma_0(x, x') = \alpha \exp\left(-\sum_{i=1}^{d} \beta_i (x_i - x_i')^2 \right). \tag{3.3}$$

This kernel is parameterized by $d + 1$ parameters: α, and $\beta_1, \ldots, \beta_d$.

The parameter $\alpha > 0$ controls how much overall variability there is in the function f. We observe that under the prior, the variance of $f(x)$ is $\mathrm{Var}(f(x)) = \mathrm{Cov}(f(x), f(x)) = \alpha$. Thus, when α is large, we are encoding in our prior distribution that $f(x)$ is likely to take a larger range of values.

The parameters $\beta_i > 0$ controls how quickly the function f varies with x. For example, consider the relationship between some point x and another point $x' = x + [1, 0, \ldots, 0]$. When β_1 is small (close to 0), the covariance between $f(x)$ and $f(x')$ is $\alpha \exp(-\beta_1) \approx \alpha$, giving a correlation between $f(x)$ and $f(x')$ of nearly 1. This reflects a belief that $f(x)$ and $f(x')$ are likely to be very similar, and that learning the value of $f(x)$ will also teach us a great deal about $f(x')$. In contrast, when β_1 is large, the covariance between $f(x)$ and $f(x')$ is nearly 0, given a correlation between $f(x)$ and $f(x')$ that is also nearly 0, reflecting a belief that $f(x)$ and $f(x')$ are unrelated to each other, and learning something about $f(x)$ will teach us little about (x').

Going beyond the squared exponential kernel

There are several other possibilities for the covariance kernel beyond the squared exponential kernel, which encode different assumptions about the underlying behavior of the function f. One particularly useful generalization of the squared exponential covariance kernel is the Matérn covariance kernel, which allows more flexibility in modeling the smoothness of f.

Before describing this kernel, let $r = \sqrt{\sum_i \left(\frac{x_i - x'_i}{\beta_i}\right)^2}$ be the Euclidean distance between x and x', but where we have altered the length scale in each dimension by some strictly positive parameter β_i. Then, the squared exponential covariance kernel can be written as, $\Sigma_0(x, x') = \alpha \exp\left(-r^2\right)$.

With this notation, the Matérn covariance kernel is,

$$\Sigma_0(x, x') = \alpha \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} r\right)^{\nu} K_\nu \left(\sqrt{2\nu} r\right),$$

where K_ν is the modified Bessel function. If we take the limit as $\nu \to \infty$, we obtain the squared exponential kernel ([14], Sect. 4.2 p. 85).

The Matérn covariance kernel is useful because it allows modeling the smoothness of f in a more flexible way, as compared with the squared exponential kernel. Under the squared exponential covariance kernel, the function f is infinitely mean-square differentiable,[1] which may not be an appropriate assumption in many applications. In contrast, under the Matérn covariance kernel, f is k-times mean-square differentiable if and only if $\nu > k$. Thus, we can model a function that is twice differentiable but no more by choosing $\nu = 5/2$, and a function that is once differentiable but no more by choosing $\nu = 3/2$.

While the squared exponential and Matérn covariance kernels allow modeling a wide range of behaviors, and together represent a toolkit that will handle a wide variety of applications, there are other covariance kernels. For a thorough discussion of these, see Chap. 4 of [14].

Both the Matérn and squared exponential covariance kernel require choosing parameters. While it certainly is possible for one to choose the parameters α and β_i (and ν in the case of Matérn) based on one's intuition about f, and what kinds of variability f is likely to have in a particular application, it is more common to choose these parameters (especially α and β_i) adaptively, so as to best fit previously observed points. We discuss this more below in Sect. 3.3.6. First, however, we discuss the choice of the mean function.

[1] Being "mean-square differentiable" at x in the direction given by the unit vector e_i means that the limit $\lim_{\delta \to 0}(f(x + \delta e_i) - f(x))/\delta$ exists in mean square. Being "k-times mean-square differentiable" is defined analogously.

3.3.2 Choice of Mean Function

We now discuss choosing the mean function $\mu_0(\cdot)$. Perhaps the most common choice is to simply set the mean function equal to a constant, μ. This constant must be estimated, along with parameters of the covariance kernel such as α and β_i, and is discussed in Sect. 3.3.6.

Beyond this simple choice, if one believes that there will be trends in f that can be described in a parametric way, then it is useful to include trend terms into the mean function. This is accomplished by choosing

$$\mu_0(x) = \mu + \sum_{j=1}^{J} \gamma_j \Psi_j(x),$$

where $\Psi_j(\cdot)$ are known functions, and $\gamma_j \in \mathbb{R}$, along with $\mu \in \mathbb{R}$, are parameters that must be estimated.

A common choice for the Ψ_j, if one chooses to include them, are polynomials in x up to some small order. For example, if $d = 2$, so x is two-dimensional, then one might include all polynomials up to second order, $\Psi_1(x) = x_1$, $\Psi_2(x) = x_2$, $\Psi_3(x) = (x_1)^2$, $\Psi_4(x) = (x_2)^2$, $\Psi_5(x) = x_1 x_2$, setting $J = 5$. One recovers the constant mean function by setting $J = 0$.

3.3.3 Inference

Given the prior distribution (3.2) on $f(x_1), \ldots, f(x_n), f(x^*)$, and given (noise-free) observations of $f(x_1), \ldots, f(x_n)$, the critical step in Gaussian process regression is calculating the posterior distribution on $f(x^*)$. We rely on the following general result about conditional probabilities and multivariate normal distributions. Its proof, which may be found in the Derivations and Proofs section, relies on Bayes rule and algebraic manipulation of the probability density of the multivariate normal distribution.

Proposition 1 *Let θ be a k-dimensional multivariate normal random column vector, with mean vector μ and covariance matrix Σ. Let $k_1 \geq 1, k_2 \geq 1$ be two integers summing to k. Decompose θ, μ and Σ as*

$$\theta = \begin{bmatrix} \theta_{[1]} \\ \theta_{[2]} \end{bmatrix}, \qquad \mu = \begin{bmatrix} \mu_{[1]} \\ \mu_{[2]} \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} \Sigma_{[1,1]} & \Sigma_{[1,2]} \\ \Sigma_{[2,1]} & \Sigma_{[2,2]} \end{bmatrix},$$

so that $\theta_{[i]}$ and $\mu_{[i]}$ are k_i-column vectors, and $\Sigma_{[i,j]}$ is a $k_i \times k_j$ matrix, for each $i, j = 1, 2$.

If $\Sigma_{1,1}$ and $\Sigma_{2,2}$ are invertible, then, for any $u \in \mathbb{R}^{k_1}$, the conditional distribution of $\theta_{[2]}$ given that $\theta_{[1]} = u$ is multivariate normal with mean

$$\mu_{[2]} + \Sigma_{[2,1]}\Sigma_{[1,1]}^{-1}(u - \mu_{[1]})$$

and covariance matrix

$$\Sigma_{[2,2]} - \Sigma_{[2,1]}\Sigma_{[1,1]}^{-1}\Sigma_{[1,2]}.$$

We use this proposition to calculate the posterior distribution on $f(x^*)$, given $f(x_1), \ldots, f(x_n)$.

Before doing so, however, we first introduce some additional notation. We let $y_{1:n}$ indicate the column vector $[y_1, \ldots, y_n]^T$, and we let $x_{1:n}$ indicate the sequence of vectors $(x_1, \ldots, x_n)$. We let $f(x_{1:n}) = [f(x_1), \ldots, f(x_n)]^T$, and similarly for other functions of x, such as $\mu_0(\cdot)$. We introduce similar additional notation for functions that take pairs of points x, x', so that $\Sigma(x_{1:n}, x_{1:n})$ is the matrix $\begin{bmatrix} \Sigma_0(x_1,x_1) & \cdots & \Sigma_0(x_1,x_n) \\ \vdots & \ddots & \vdots \\ \Sigma_0(x_n,x_1) & \cdots & \Sigma_0(x_n,x_n) \end{bmatrix}$, $\Sigma_0(x^*, x_{1:n})$ is the row vector $[\Sigma_0(x^*, x_1), \ldots, \Sigma_0(x^*, x_n)]$, and $\Sigma_0(x_{1:n}, x^*)$ is the column vector $[\Sigma_0(x_1, x^*), \ldots, \Sigma_0(x_n, x^*)]^T$.

This notation allows us to rewrite (3.2) as

$$\begin{bmatrix} y_{1:n} \\ f(x^*) \end{bmatrix} = \text{Normal}\left(\begin{bmatrix} \mu_0(x_{1:n}) \\ \mu_0(x^*) \end{bmatrix}, \begin{bmatrix} \Sigma_0(x_{1:n}, x_{1:n}) & \Sigma_0(x_{1:n}, x^*) \\ \Sigma_0(x^*, x_{1:n}) & \Sigma_0(x^*, x^*) \end{bmatrix}\right). \tag{3.4}$$

We now examine this expression in the context of Proposition 1. We set $\theta_{[1]} = f(x_{1:n})$, $\theta_{[2]} = f(x^*)$, $\mu_{[1]} = \mu_0(x_{1:n})$, $\mu_{[2]} = \mu_0(x^*)$, $\Sigma_{[1,1]} = \Sigma_0(x_{1:n}, x_{1:n})$, $\Sigma_{[1,2]} = \Sigma_0(x_{1:n}, x^*)$, $\Sigma_{[2,1]} = \Sigma_0(x^*, x_{1:n})$, and $\Sigma_{[2,2]} = \Sigma_0(x^*, x^*)$.

Then, applying Proposition 1, we see that the posterior distribution on $f(x^*)$ given observations $y_i = f(x_i), i = 1, \ldots, n$ is normal, with a mean $\mu_n(x^*)$ and variance $\sigma_n^2(x^*)$ given by,

$$\mu_n(x^*) = \mu_0(x^*) + \Sigma_0(x^*, x_{1:n})\Sigma_0(x_{1:n}, x_{1:n})^{-1}(f(x_{1:n}) - \mu_0(x_{1:n})), \tag{3.5}$$

$$\sigma_n^2(x^*) = \Sigma_0(x^*, x^*) - \Sigma_0(x^*, x_{1:n})\Sigma_0(x_{1:n}, x_{1:n})^{-1}\Sigma_0(x_{1:n}, x^*). \tag{3.6}$$

The invertibility of $\Sigma_0(x_{1:n}, x_{1:n})$ (and also $\Sigma_0(x^*, x^*)$) required by Proposition 1 depends on the covariance kernel and its parameters (typically called hyperparameters), but this invertibility typically holds as long as these hyperparameters satisfy mild non-degeneracy conditions, and the $x_{1:n}$ are distinct, i.e., that we have not measured the same point more than once. For example, under the squared exponential covariance kernel, invertibility holds as long as $\alpha > 0$ and the $x_{1:n}$ are distinct. If we have measured a point multiple times, then we can safely drop all but one of the measurements, here where observations are noise-free. Below, we treat the case where observations are noisy, and in this case including multiple measurements of the same point is perfectly reasonable and does not cause issues.

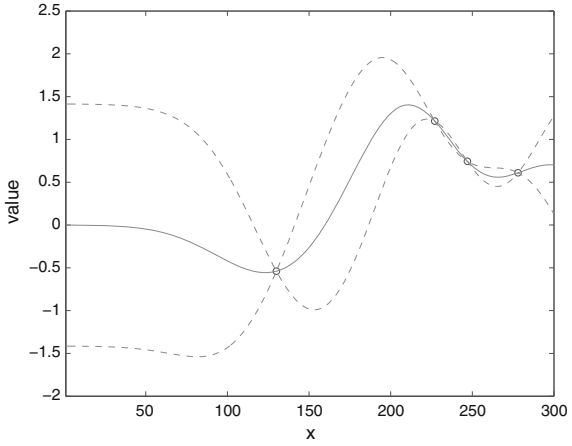

Fig. 3.1 Illustration of Gaussian process regression with noise-free evaluations. The *circles* show previously evaluated points, $(x_i, f(x_i))$. The *solid line* shows the posterior mean, $\mu_n(x)$, as a function of x, which is an estimate $f(x)$, and the *dashed lines* show a Bayesian credible interval for each $f(x)$, calculated as $\mu_n(x) \pm 1.96\sigma_n(x)$. Although this example shows f taking a scalar input, Gaussian process regression can be used for functions with vector inputs

Figure 3.1 shows the output from Gaussian process regression. In the figure, circles show points $(x_i, f(x_i))$, the solid line shows $\mu_n(x^*)$ as a function of x^*, and the dashed lines are positioned at $\mu_n(x^*) \pm 1.96\sigma_n(x^*)$, forming a 95 % Bayesian credible interval for $f(x^*)$, i.e., an interval in which $f(x^*)$ lies with posterior probability 95 %. (A credible interval is the Bayesian version of a frequentist confidence interval.) Because observations are noise-free, the posterior mean $\mu_n(x^*)$ interpolates the observations $f(x^*)$.

3.3.4 Inference with Just One Observation

The expressions (3.5) and (3.6) are complex, and perhaps initially difficult to assimilate. To give more intuition about them, and also to support some additional analysis below in Sect. 3.4, it is useful to consider the simplest case, when we have just a single measurement, $n = 1$.

In this case, all matrices in (3.5) and (3.6) are scalars, $\Sigma_0(x^*, x_1) = \Sigma_0(x_1, x^*)$, and the expressions (3.5) and (3.6) can be rewritten as,

$$\mu_1(x^*) = \mu_0(x^*) + \frac{\Sigma_0(x^*, x_1)}{\Sigma_0(x_1, x_1)}(f(x_1) - \mu_0(x_1)), \tag{3.7}$$

$$\sigma_1^2(x^*) = \Sigma_0(x^*, x^*) - \frac{\Sigma_0(x^*, x_1)^2}{\Sigma_0(x_1, x_1)}. \tag{3.8}$$

Intuition about the expression for the posterior mean

We first examine (3.7). We see that the posterior mean of $f(x^*)$, $\mu_1(x^*)$, which we can think of as our estimate of $f(x^*)$ after observing $f(x_1)$, is obtained by taking our original estimate of $f(x^*)$, $\mu_0(x^*)$, and adding to it a correction term. This correction term is itself the product of two quantities: the error $f(x_1) - \mu_0(x_1)$ in our original estimate of $f(x_1)$, and the quantity $\frac{\Sigma_0(x^*,x_1)}{\Sigma_0(x_1,x_1)}$. Typically, $\Sigma_0(x^*, x_1)$ will be positive, and hence also $\frac{\Sigma_0(x^*,x_1)}{\Sigma_0(x_1,x_1)}$. (Recall, $\Sigma_0(x_1, x_1)$ is a variance, so is never negative.) Thus, if $f(x_1)$ is bigger than expected, $f(x_1) - \mu_0(x_1)$ will be positive, and our posterior mean $\mu_1(x^*)$ will be larger than our prior mean $\mu_0(x^*)$. In contrast, if $f(x_1)$ is smaller than expected, $f(x_1) - \mu_0(x_1)$ will be negative, and our posterior mean $\mu_1(x^*)$ will be smaller than our prior mean $\mu_0(x^*)$.

We can examine the quantity $\frac{\Sigma_0(x^*,x_1)}{\Sigma_0(x_1,x_1)}$ to understand the effect of the position of x^* relative to x_1 on the magnitude of the correction to the posterior mean. Notice that x^* only enters this expression through the numerator. If x^* is close to x_1, then $\Sigma_0(x^*, x_1)$ will be large under the squared exponential and most other covariance kernels, and positive values for $f(x_1) - \mu_0(x_1)$ will also cause a strong positive change in $\mu_1(x^*)$ relative to $\mu_0(x^*)$. If x^* is far from x_1, then $\Sigma_0(x^*, x_1)$ will be close to 0, and $f(x_1) - \mu_0(x_1)$ will have little effect on $\mu_1(x^*)$.

Intuition about the expression for the posterior variance

Now we examine (3.8). We see that the variance of our belief on $f(x^*)$ under the posterior, $\sigma_1^2(x^*)$, is smaller than its value under the prior, $\Sigma_0(x^*, x^*)$. Moreover, when x^* is close to x_1, $\Sigma_0(x^*, x_1)$ will be large, and the reduction in variance from prior to posterior will also be large.

Conversely, when x^* is far from x_1, $\Sigma_0(x^*, x_1)$ will be close to 0, and the variance under the posterior will be similar to its value under the prior.

As a final remark, we can also rewrite the expression (3.8) in terms of the squared correlation under the prior, $\mathrm{Corr}(f(x^*), f(x_1))^2 = \Sigma_0(x^*, x_1)^2/(\Sigma_0(x^*, x^*)\Sigma_0(x_1, x_1)) \in [0, 1]$, as

$$\sigma_1^2(x^*) = \Sigma_0(x^*, x^*)\left(1 - \mathrm{Corr}(f(x^*), f(x_1))^2\right).$$

We thus see that the reduction in variance of the posterior distribution depends on the squared correlation under the prior, with larger squared correlation implying a larger reduction.

3.3.5 Inference with Noisy Observations

The previous section assumed that $f(x^*)$ can be observed exactly, without any error. When $f(x^*)$ is the outcome of a physical experiment, however, our observations are obscured by noise. Indeed, if we were to synthesize and test the same material design x^* multiple times, we might observe different results.

To model this situation, Gaussian process regression can be extended to allow observations of the form,

$$y(x_i) = f(x_i) + \varepsilon_i,$$

where we assume that the ε_i are normally distributed with mean 0 and constant variance, λ^2, with independence across i. In general, the variance λ^2 is unknown, but we treat it as a known parameter of our model, and then estimate it along with all the other parameters of our model, as discussed below in Sect. 3.3.6.

These assumptions of constant variance (called *homoscedasticity*) and independence make the analysis significantly easier, although they are often violated in practice. Experimental conditions that tend to violate these assumptions are discussed below, as are versions of GP regression that can be used when they are violated.

Analysis of independent homoscedastic noise

To perform inference under independent homoscedastic noise, and calculate a posterior distribution on the value of the function $f(x_*)$ at a given point x_*, our first step is to write down the joint distribution of our observations $y_1, \ldots, y_n$ and the quantity we wish to predict, $f(x_*)$, under the prior. That is, we write down the distribution of the vector $[y_1, \ldots, y_n, f(x_*)]$.

We first observe that $[y_1, \ldots, y_n, f(x_*)]$ is the sum of $[f(x_1), \ldots, f(x_n), f(x_*)]$ and another vector, $[\varepsilon_1, \ldots, \varepsilon_n, 0]$. The first vector has a multivariate normal distribution given by (3.4). The second vector is independent of the first and is also multivariate normal, with a mean vector that is identically 0, and a covariance matrix $\mathrm{diag}(\lambda^2, \ldots, \lambda^2, 0)$. The sum of two independent multivariate normal random vectors is itself multivariate normal, with a mean vector and covariance matrix given, respectively, by the sums of the mean vectors and covariance matrices of the summands. This gives the distribution of $[y_1, \ldots, y_n, f(x_*)]$ as

$$\begin{bmatrix} y_{1:n} \\ f(x^*) \end{bmatrix} \sim \text{Normal}\left(\begin{bmatrix} \mu_0(x_{1:n}) \\ \mu_0(x^*) \end{bmatrix}, \begin{bmatrix} \Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n & \Sigma_0(x_{1:n}, x^*) \\ \Sigma_0(x^*, x_{1:n}) & \Sigma_0(x^*, x^*) \end{bmatrix} \right), \quad (3.9)$$

where I_n is the n-dimensional identity matrix.

As we did in Sect. 3.3.3, we can use Proposition 1 with the above expression to compute the posterior on $f(x^*)$ given $f(x_{1:n})$. We obtain,

$$\mu_n(x^*) = \mu_0(x^*) + \Sigma_0(x^*, x_{1:n}) \left[\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n \right]^{-1} (y_{1:n} - \mu_0(x_{1:n})) \quad (3.10)$$

$$\sigma_n^2(x^*) = \Sigma_0(x^*, x^*) - \Sigma_0(x^*, x_{1:n}) \left[\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n \right]^{-1} \Sigma_0(x_{1:n}, x^*). \quad (3.11)$$

If we set $\lambda^2 = 0$, so there is no noise, then we recover (3.5) and (3.6).

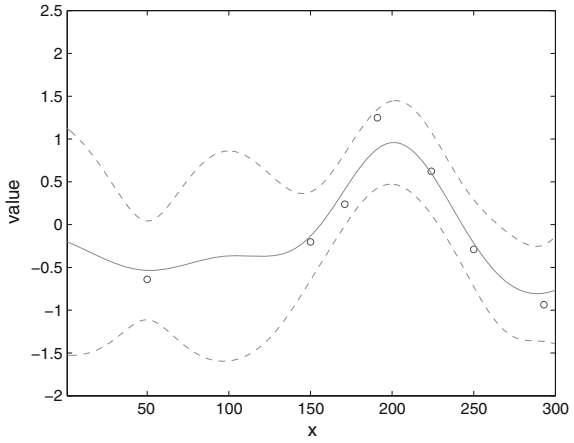

Fig. 3.2 Illustration of Gaussian process regression with noisy evaluations. As in Fig. 3.1, the *circles* show previously evaluated points, (x_i, y_i), where y_i is $f(x_i)$ perturbed by constant-variance independent noise. The *solid line* shows the posterior mean, $\mu_n(x)$, as a function of x, which is an estimate of the underlying function f, and the *dashed lines* show a Bayesian credible interval for f, calculated as $\mu_n(x) \pm 1.96\sigma_n(x)$

Figure 3.2 shows an example of a posterior distribution calculated with Gaussian process regression with noisy observations. Notice that the posterior mean no longer interpolates the observations, and the credible interval has a strictly positive width at points where we have measured. Noise prevents us from observing function values exactly, and so we remain uncertain about the function value at points we have measured.

Going beyond homoscedastic independent noise

Constant variance is violated if the experimental noise differs across materials designs, which occurs most frequently when noise arises during the synthesis of the material itself, rather than during the evaluation of a material that has already been created. Some work has been done to extend Gaussian process regression to flexibly model heteroscedastic noise (i.e., noise whose variance changes) [18–21]. The main idea in much of this work is to use a second Gaussian process to model the changing variance across the input domain. Much of this work assumes that the noise is independent and Gaussian, though [21] considers non-Gaussian noise.

Independence is most typically violated, in the context of physical experiments, when the synthesis and evaluation of multiple materials designs is done together, and the variation in some shared component simultaneously influences these designs, e.g., through variation in the temperature while the designs are annealing together, or through variation in the quality of some constituent used in synthesis. We are aware of relatively little work modeling dependent noise in the context of Gaussian process regression and Bayesian optimization, with one exception being [22].

3.3.6 Parameter Estimation

The mean and covariance functions contain several parameters. For example, if we use the squared exponential kernel, a constant mean function, and observations have independent homoscedastic noise, then we must choose or estimate the parameters $\mu, \alpha, \beta_1, \ldots, \beta_d, \lambda$. These parameters are typically called *hyperparameters* because they are parameters of the prior distribution. (λ^2 is actually a parameter of the likelihood function, but it is convenient to treat it together with the parameters of the prior.) While one may simply choose these hyperparameters directly, based on intuition about the problem, a more common approach is to choose them adaptively, based on data.

To accomplish this, we write down an expression for the probability of the observed data $y_{1:n}$ in terms of the hyperparameters, marginalizing over the uncertainty on $f(x_{1:n})$. Then, we optimize this expression over the hyperparameters to find settings that make the observed data as likely as possible. This approach to setting hyperparameters is often called *empirical Bayes*, and it can be seen as an approximation to full Bayesian inference.

We detail this approach for the squared exponential kernel with a constant mean function. Estimation for other kernels and mean functions is similar. Using the probability distribution of $y_{1:n}$ from (3.9), and neglecting constants, the natural logarithm of this probability, $\log p(y_{1:n} \mid x_{1:n})$ (called the "log marginal likelihood"), can be calculated as

$$-\frac{1}{2}(y_{1:n}-\mu)^T\left(\Sigma_0(x_{1:n},x_{1:n})+\lambda^2 I_n\right)^{-1}(y_{1:n}-\mu)-\frac{1}{2}\log|\Sigma_0(x_{1:n},x_{1:n})+\lambda^2 I_n|,$$

where $|\cdot|$ applied to a matrix indicates the determinant.

To find the hyperparameters that maximize this log marginal likelihood (the neglected constant does not affect the location of the maximizer), we will take partial derivatives with respect to each hyperparameter. We will then use them to find maximizers of μ and $\sigma^2 := \alpha + \lambda^2$ analytically, and then use gradient-based optimization to maximize the other hyperparameters.

Taking a partial derivative with respect to μ, setting it to zero, and solving for μ, we get that the value of μ that maximizes the marginal likelihood is

$$\hat{\mu} = \frac{\sum_{i=1}^{n}\left((\Sigma_0(x_{1:n},x_{1:n})+\lambda^2 I_n)^{-1}y_{1:n}\right)_i}{\sum_{i,j=1}^{n}(\Sigma_0(x_{1:n},x_{1:n})+\lambda^2 I_n)^{-1}_{ij}}.$$

Define R as the matrix with components

$$R_{ij} = \begin{cases} 1 & i = j, \\ g\exp\left(-\sum_{i=1}^{d}\beta_i(x_i - x_j)^2\right) & i \neq j, \end{cases}$$

where $g = \frac{\alpha}{\sigma^2}$. Then $\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n = \sigma^2 R$ and $\hat{\mu}$ can be written in terms of R as $\hat{\mu} = \frac{\sum_{i=1}^n \left(R^{-1} y_{1:n}\right)_i}{\sum_{i,j=1}^n R_{ij}^{-1}}$. The log marginal likelihood (still neglecting constants) becomes

$$\log p(y_{1:n} \mid x_{1:n}) \sim -\frac{1}{2}(y_{1:n} - \hat{\mu})^T (\sigma^2 R)^{-1} (y_{1:n} - \hat{\mu}) - \frac{1}{2} \log |\sigma^2 R|.$$

Taking the partial derivative with respect to σ^2, and noting that $\hat{\mu}$ does not depend on σ^2, we solve for σ^2 and obtain

$$\widehat{\sigma^2} = \frac{1}{n}(y_{1:n} - \hat{\mu}) R^{-1} (y_{1:n} - \hat{\mu}).$$

Substituting this estimate, the log marginal likelihood becomes

$$\log p(y_{1:n} \mid x_{1:n}) \sim -\log \left(\frac{1}{n} |R|^{\frac{1}{n}} (y_{1:n} - \hat{\mu})^T R^{-1} (y_{1:n} - \hat{\mu}) \right). \tag{3.12}$$

The expression (3.12) cannot in general be optimized analytically. Instead, one typically optimizes it numerically using a first- or second-order optimization algorithm, such as Newton's method or gradient descent, obtaining estimates for $\beta_1, \ldots, \beta_d$ and g. These estimates are in turn substituted to provide an estimate of R, from which estimates $\hat{\mu}$ and $\widehat{\sigma^2}$ may be computed. Finally, using $\widehat{\sigma^2}$ and the estimated value of g, we may estimate α and λ.

3.3.7 Diagnostics

When using Gaussian process regression, or any other machine learning technique, it is advisable to check the quality of the predictions, and to assess whether the assumptions made by the method are met. One way to do this is illustrated by Fig. 3.3, which comes from a simulation of blood flow near the heart, based on [23], for which we get exact (not noisy) observations of $f(x)$.

This plot is created with a technique called leave-one-out cross validation. In this technique, we iterate through the datapoints $x_{1:n}$, $y_{1:n}$, and for each $i \in \{1, \ldots, n\}$, we train a Gaussian process regression model on all of the data *except* x_i, y_i, and then use it, together with x_i, to predict what the value y_i should be. We obtain from this a posterior mean (the prediction), call it $\mu_{-i}(x_i)$, and also a posterior standard deviation, call it $\sigma_{-i}(x_i)$. When calculating these estimates, it is best to separately re-estimate the hyperparameters each time, leaving out the data (x_i, y_i). We then calculate a 95 % credible interval $\mu_{-i}(x_i) \pm 2\sigma_{-i}(x_i)$, and create Fig. 3.3 by plotting "Predicted" versus "Actual", where the "Actual" coordinate (on the x-axis) is y_i, and the "Predicted" value (on the y-axis) is pictured as an error bar centered at $\mu_{-i}(x_i)$ with half-width $2\sigma_{-i}(x_i)$.

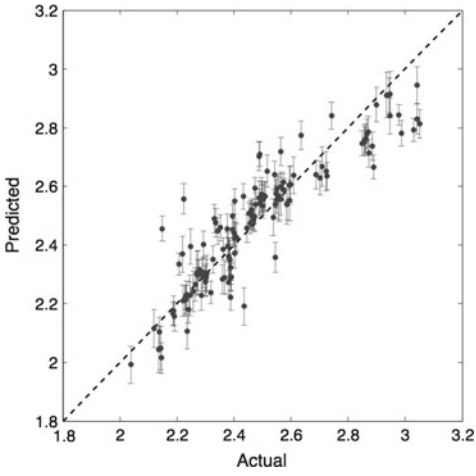

Fig. 3.3 Diagnostic plot for Gaussian process regression, created with leave-one-out cross validation. For each point in our dataset, we hold that point (x_i, y_i) out, train on the remaining points, calculate a 95 % credible interval for y_i, and plot this confidence interval as an error bar whose x-coordinate is the actual value y_i. If Gaussian process regression is working well, 95 % of the *error bars* will intersect the diagonal line Predicted = Actual

If the uncertainty estimates outputted by Gaussian process regression are behaving as anticipated, then approximately 95 % of the credible intervals will intersect the diagonal line Predicted = Actual. Moreover, if Gaussian process regression's predictive accuracy is high, then the credible intervals will be short, and their centers will be close to this same line Predicted=Actual.

This idea may be extended to noisy function evaluations, under the assumption of independent homoscedastic noise. To handle the fact that the same point may be sampled multiple times, let $m(x)$ be the number of times that a point $x \in \{x_1, \ldots, x_n\}$ was sampled, and let $\overline{y}(x)$ be the average of the observed values at this point. Moreover, by holding out all $m(x)$ samples of x and training Gaussian process regression, we would obtain a normal posterior distribution on $f(x_i)$ that has mean $\mu_{-i}(x_i)$ and standard deviation $\sigma_{-i}(x_i)$.

Since $\overline{y}(x_i)$ is then the sum of $f(x_i)$ and some normally distributed noise with mean 0 and variance $\lambda^2/m(x_i)$, the resulting distribution of $\overline{y}(x_i)$ is normal with mean $\mu_{-i}(x_i)$ and standard deviation $\sqrt{\sigma^2_{-i}(x_i) + \lambda^2/m(x_i)}$.

From this, a 95 % credible interval for $\overline{y}(x_i)$ is then $\mu_{-i}(x_i) \pm 2\sqrt{\sigma^2_{-i}(x_i) + \lambda^2/m(x_i)}$. We would plot Predicted versus Observed by putting this credible interval along the y-axis at x-coordinate $\overline{y}(x_i)$. If Gaussian process regression is working well, then approximately 95 % of these credible intervals will intersect the line Predicted = Observed.

For Gaussian process regression to best support Bayesian optimization, it is typically most important to have good uncertainty estimates, and relatively less important to have high predictive accuracy. This is because Bayesian optimization uses Gaussian process regression as a guide for deciding where to sample, and so if Gaussian process regression reports that there is a great deal of uncertainty at a particular location and thus low predictive accuracy, Bayesian optimization can choose to sample at this location to improve accuracy. Thus, Bayesian optimization has

a recourse for dealing with low predictive accuracy, as long as the uncertainty is accurately reported. In contrast, if Gaussian process regression estimates poor performance at a location that actually has near-optimal performance, and also provides an inappropriately low error estimate, then Bayesian optimization may not sample there within a reasonable timeframe, and thus may never correct the error.

If either the uncertainty is incorrectly estimated, or the predictive accuracy is unsatisfactorily low, then the most common "fixes" employed are to adopt a different covariance kernel, or to transform the objective function f. If the objective function is known to be non-negative, then the transformations $\log(f)$ and $\sqrt{f}$ are convenient for optimization because they are both strictly increasing, and so do not change the set of maximizers (or minimizers). If f is not non-negative, but is bounded below by some other known quantity a, then one may first shift f upward by a.

3.3.8 Predicting at More Than One Point

Below, to support the development of the knowledge-gradient method in Sects. 3.4.2 and 3.6, it will be useful to predict the value of f at multiple points, $x_1^*, \ldots, x_k^*$, with noise. To do so, we could certainly apply (3.10) and (3.11) separately for each $x_1^*, \ldots, x_k^*$, and this would provide us with both an estimate (the posterior mean) and a measure of the size of the error in this estimate (the posterior variance) associated with each $f(x_i^*)$. It would not, however, quantify the relationship between the errors at several different locations. For this, we must perform the estimation jointly.

As we did in Sect. 3.3.5, we begin with our prior on $[y_{1:n}, f(x_{1:k}^*)]$, which is,

$$\begin{bmatrix} y_{1:n} \\ f(x_{1:k}^*) \end{bmatrix} \sim \text{Normal}\left(\begin{bmatrix} \mu_0(x_{1:n}) \\ \mu_0(x_{1:k}^*) \end{bmatrix}, \begin{bmatrix} \Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n & \Sigma_0(x_{1:n}, x_{1:k}^*) \\ \Sigma_0(x_{1:k}^*, x_{1:n}) & \Sigma_0(x_{1:k}^*, x_{1:k}^*) \end{bmatrix} \right),$$

We then use Proposition 1 to compute the posterior on $f(x_{1:k}^*)$ given $f(x_{1:n})$, which is multivariate normal with mean vector $\mu_n(x_{1:k}^*)$ and covariance matrix $\Sigma_n(x_{1:k}^*, x_{1:k}^*)$ given by,

$$\mu_n(x_{1:k}^*) = \mu_0(x_{1:k}^*) + \Sigma_0(x_{1:k}^*, x_{1:n}) \left[\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n \right]^{-1} (y_{1:n} - \mu_0(x_{1:n})), \tag{3.13}$$

$$\Sigma_n(x_{1:k}^*, x_{1:k}^*) = \Sigma_0(x_{1:k}^*, x_{1:k}^*) - \Sigma_0(x_{1:k}^*, x_{1:n}) \left[\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n \right]^{-1} \Sigma_0(x_{1:n}, x_{1:k}^*). \tag{3.14}$$

We see that setting $k = 1$ provides the expressions (3.10) and (3.11) from Sect. 3.3.5.

3.3.9 Avoiding Matrix Inversion

The expressions (3.10) and (3.11) for the posterior mean and variance in the noisy case, and also (3.7) and (3.8) in the noise-free case, include a matrix inversion term. Calculating this matrix inversion is slow and can be hard to accomplish accurately in practice, due to the finite precision of floating point implementations. Accuracy is especially an issue when Σ has terms that are close to 0, which arises when data points are close together.

In practice, rather than calculating a matrix inverse directly, it is typically faster and more accurate to use a mathematically equivalent algorithm, which performs a Cholesky decomposition and then solves a linear system. This algorithm is described below, and is adapted from Algorithm 2.1 in Sect. 2.3 of [14]. This algorithm also computes the log marginal likelihood required for estimating hyperparameters in Sect. 3.3.6.

Algorithm 1 Implementation using Cholesky decomposition

Require: $x_{1:n}$ (inputs), $y_{1:n}$ (responses), $\Sigma_0(x, x')$ (covariance function), λ^2 (variance of noise), x^* (test input).

$L = \text{Cholesky}\left(\Sigma_0(x_{1:n}, x_{1:n}) + \lambda^2 I_n\right)$

$\delta = L^T \backslash (L \backslash (y_{1:n} - \mu_0(x_{1:n})))$

$\mu_n(x^*) = \mu_0(x^*) + \Sigma_0(x^*, x_{1:n})\delta$

$v = L \backslash \Sigma_0(x_{1:n}, x^*)$

$\sigma_n^2(x^*) = \Sigma_0(x^*, x^*) - v^T v$

$\log p(y_{1:n} \mid x_{1:n}) = -\frac{1}{2}\left(y_{1:n} - \mu_0(x_{1:n})\right)^T \alpha - \Sigma_i \log L_{ii} - \frac{n}{2}\log 2\pi$

return $\mu_n(x^*)$ (mean), $\sigma_n^2(x^*)$ (variance), $\log p(y_{1:n} \mid x_{1:n})$ (log marginal likelihood).

3.4 Choosing Where to Sample

Being able to infer the value of the objective function $f(x)$ at unevaluated points based on past data $x_{1:n}$,$y_{1:n}$ is only one part of finding good designs. The other part is using this ability to make good decisions about where to direct future sampling.

Bayesian optimization methods addresses this by using a measure of the value of the information that would be gained by sampling at a point. Bayesian optimization methods then choose the point to sample next for which this value is largest. A number of different ways of measuring the value of information have been proposed. Here, we describe two in detail, expected improvement [2, 4], and the knowledge gradient [24, 25], and then survey a broader collection of design criteria.

3.4.1 Expected Improvement

Expected improvement, as it was first proposed, considered only the case where measurements are free from noise. In this setting, suppose we have taken n measurements at locations $x_{1:n}$ and observed $y_{1:n}$. Then

$$f_n^* = \max_{i=1,\ldots,n} f(x_i)$$

is the best value observed so far. Suppose we are considering evaluating f at a new point x. After this evaluation, the best value observed will be

$$f_{n+1}^* = \max(f(x), f_n^*),$$

and the difference between these values, which is the *improvement* due to sampling, is

$$f_{n+1}^* - f_n^* = \max(f(x) - f_n^*, 0) = (f(x) - f_n^*)^+,$$

where $a^+ = \max(a, 0)$ indicates the positive part function.

Ideally, we would choose x to make this improvement as large as possible. Before actually evaluating $f(x)$, however, we do not know what this improvement will be, so we cannot implement this strategy. However, we do have a probability distribution on $f(x)$, from Gaussian process regression. The *expected improvement*, indicated $\mathrm{EI}(x)$, is obtained by taking the expectation of this improvement with respect to the posterior distribution on $f(x)$ given $x_{1:n}$, $y_{1:n}$.

$$\mathrm{EI}(x) = E_n[(f(x) - f_n^*)^+], \tag{3.15}$$

where $E_n[\,\cdot\,] = E[\,\cdot\,|x_{1:n}, y_{1:n}]$ indicates the expectation with respect to the posterior distribution.

The expectation in (3.15) can be computed more explicitly, in terms of the normal cumulative distribution function (cdf) $\Phi(\cdot)$, and the normal probability density function (pdf) $\varphi(\cdot)$. Recalling from Sect. 3.3.3 that $f(x) \sim \mathrm{Normal}(\mu_n(x), \sigma_n^2(x))$, where $\mu_n(x)$ and $\sigma_n^2(x)$ are given by (3.5) and (3.6), and integrating with respect to the normal distribution (a derivation may be found in the Derivations and Proofs section), we obtain,

$$\mathrm{EI}(x) = (\mu_n(x) - f_n^*)\Phi\left(\frac{\mu_n(x) - f_n^*}{\sigma_n(x)}\right) + \sigma_n(x)\varphi\left(\frac{\mu_n(x) - f_n^*}{\sigma_n(x)}\right). \tag{3.16}$$

Figure 3.4 plots this expected improvement for a problem with a one-dimensional input space. We can see from this plot that the expected improvement is largest at locations where both the posterior mean $\mu_n(x)$ is large, and also the posterior standard deviation $\sigma_n(x)$ is large. This is reasonable because those points that are most likely to provide large gains are those points that have a high predicted value, but that also

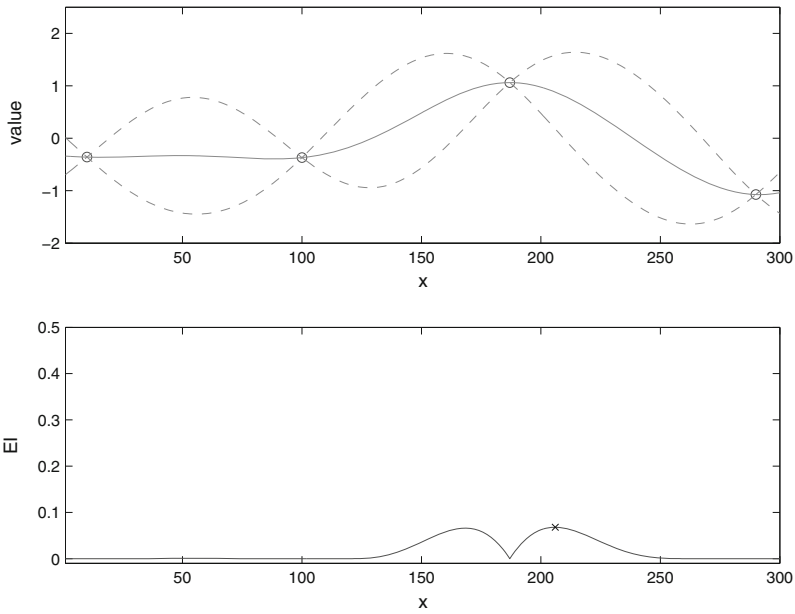

Fig. 3.4 *Upper panel* shows the posterior distribution in a problem with no noise and a one-dimensional input space, where the *circles* are previously measured points, the *solid line* is the posterior mean $\mu_n(x)$, and the *dashed lines* are at $\mu_n(x) \pm 2\sigma_n(x)$. *Lower panel* shows the expected improvement $\mathrm{EI}(x)$ computed from this posterior distribution. An "x" is marked at the point with the largest expected improvement, which is where we would evaluate next

have significant uncertainty. Indeed, at points where we have already observed, and thus have no uncertainty, the expected improvement is 0. This is consistent with the idea that, in a problem without noise, there is no value to repeating an evaluation that has already been performed.

This idea of favoring points that, on the one hand, have a large predicted value, but, on the other hand, have a significant amount of uncertainty, is called the *exploration versus exploitation tradeoff*, and appears in areas beyond Bayesian optimization, especially in reinforcement learning [26, 27] and multi-armed bandit problems [28, 29]. In these problems, we are taking actions repeatedly over time whose payoffs are uncertain, and wish to simultaneously get good immediate rewards, while learning the reward distributions for all actions to allow us to get better rewards in the future. We emphasize, however, that the correct balance between exploration and exploitation is different in Bayesian optimization as compared with multi-armed bandits, and should more favor exploration: in optimization, the advantage of measuring where the predicted value is high is that these areas tend to give more useful information about where the optimum lies; in contrast, in problems where we must "learn while doing" like multi-armed bandits, evaluating an action with high predicted reward is good primarily because it tends to give a high immediate reward.

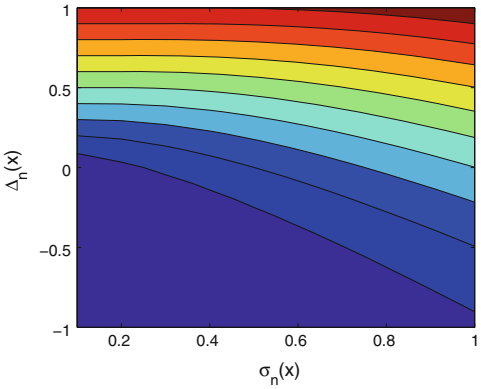

Fig. 3.5 Contour plot of the expected improvement, as a function of the difference in means $\Delta_n(x) := \mu_n(x) - f_n^*$ and the posterior standard deviation $\sigma_n(x)$. The expected improvement is larger when the difference in means is larger, and when the standard deviation is larger

We can also see the exploration versus exploitation tradeoff implicit in the expected improvement function in the contour plot, Fig. 3.5. This plot shows the contours of $\mathrm{EI}(x)$ as a function of the posterior mean, expressed as a difference from the previous best, $\Delta_n(x) := \mu_n(x) - f_n^*$, and the posterior standard deviation $\sigma_n(x)$.

Given the expression (3.16), Bayesian optimization algorithms based on expected improvement, such as the Efficient Global Optimization (EGO) algorithm proposed by [4], and the earlier algorithms of Mockus (see, e.g., the monograph [2]), then recommend sampling at the point with the largest expected improvement. That is,

$$x_{n+1} \in \operatorname*{argmax}_x \mathrm{EI}(x). \tag{3.17}$$

Finding the point with largest expected improvement is itself a global optimization problem, like the original problem that we wished to solve (3.1). Unlike (3.1), however, $\mathrm{EI}(x)$ can be computed quickly, and its first and second derivatives can also be computed quickly. Thus, we can expect to be able to solve (3.1) relatively well using an off-the-shelf optimization method for continuous global optimization. A common approach is to use a local solver for continuous optimization, such as gradient ascent, in a multistart framework, where we start the local solver from many starting points chosen at random, and then select the best local solution discovered. In Sect. 3.5 we describe several codes that implement expected improvement methods, and each makes its own choice about how to solve (3.17).

The algorithm given by (3.17) is *optimal* under three assumptions: (1) that we will take only a single sample; (2) there is no noise in our samples; and (3) that the x we will report as our final solution (i.e., the one that we will *implement*) must be among those previously sampled.

In practice, assumption (1) is violated, as Bayesian optimization methods like (3.17) are applied iteratively, and is made simply because it simplifies the analysis. Being able to handle violations of assumption (1) in a more principled way is of great interest to researchers working on Bayesian optimization methodology, and some partial progress in that direction is discussed in Sect. 3.4.3. Assumption

(2) is also often violated in a broad class of applications, especially those involving physical experiments or stochastic simulations. In the next section, we present an algorithm, the knowledge-gradient algorithm [24, 25], that relaxes this assumption (2), and also allows relaxing assumption (3) if this is desired.

3.4.2 Knowledge Gradient

When we have noise in our samples, the derivation of expected improvement meets with difficulty. In particular, if we have noise, then $f_n^* = \max_{i=1,\ldots,n} f(x_i)$ is not precisely known, preventing us from using the expression (3.16).

One may simply take a quantity like $\max_{i=1,\ldots,n} y_i$ that is similar in spirit to $f_n^* = \max_{i=1,\ldots,n} f(x_i)$, and replace f_n^* in (3.16) with this quantity, but the resulting algorithm is no longer justified by an optimality analysis. Indeed, for problems with a great deal of noise, $\max_{i=1,\ldots,n} y_i$ tends to be significantly larger than the true underlying value of the best point previously sampled, and so the resulting algorithm may be led to make a poor tradeoff between exploration and exploitation, and exhibit poor performance in such situations.

Instead, the knowledge-gradient algorithm [24, 25] takes a more principled approach, and starts where the derivation of expected improvement began, but fully accounts for the introduction of noise (assumption 2 in Sect. 3.4.1), and the possibility that we wish to search over a class of solutions broader than just those that have been previously evaluated when recommending the final solution (assumption 3 in Sect. 3.4.1).

We first introduce a set A_n, which is the set of points from which we would choose the final solution, if we were asked to recommend a final solution at time n, based on $x_{1:n}$, $y_{1:n}$. For tractability, we suppose A_n is finite. For example, if A is finite, as it often is in discrete optimization via simulation problems, we could take $A_n = A$, allowing the whole space of feasible solutions. This choice was considered in [24]. Alternatively, one could take $A_n = \{x_1, \ldots, x_n\}$, stating that one is willing to consider only those points that have been previously evaluated. This choice is consistent with the expected improvement algorithm. Indeed, we will see that when one makes this choice, and measurements are free from noise, then the knowledge-gradient algorithm is identical to the expected improvement algorithm. Thus, the knowledge-gradient algorithm generalizes the expected improvement algorithm.

If we were to stop sampling at time n, then the expected value of a point $x \in A_n$ based on the information available would be $E_n[f(x)] = \mu_n(x)$. In the special case when evaluations are free from noise, this is equal to $f(x)$, but when there is noise, these two quantities may differ. If we needed to report a final solution, we would then choose the point in A_n for which this quantity is the largest, i.e., we would choose $\text{argmax}_{x \in A_n} \mu_n(x)$. Moreover, the expected value of this solution would be

$$\mu_n^* = \max_{x \in A_n} \mu_n(x).$$

If evaluations are free from noise and $A_n = \{x_{1:n}\}$, then μ_n^* is equal to f_n^*, but in general these quantities may differ.

If we take one additional sample, then the expected value of the solution we would report based on this additional information is

$$\mu_{n+1}^* = \max_{x \in A_{n+1}} \mu_{n+1}(x),$$

where as before, A_{n+1} is some finite set of points we would be willing to consider when choosing a final solution. Observe in this expression that $\mu_{n+1}(x)$ is not necessarily the same as $\mu_n(x)$, even for points $x \in \{x_{1:n}\}$ that we had previously evaluated, but that $\mu_{n+1}(x)$ can be computed from the history of observations $x_{1:n+1}$, $y_{1:n+1}$.

The improvement in our expected solution value is then the difference between these two quantities, $\mu_{n+1}^* - \mu_n^*$. This improvement is random at time n, even fixing x_{n+1}, through its dependence on y_{n+1}, but we can take its expectation. The resulting quantity is called the *knowledge-gradient (KG) factor*, and is written,

$$\mathrm{KG}_n(x) = E_n\left[\mu_{n+1}^* - \mu_n^* \mid x_{n+1} = x\right]. \tag{3.18}$$

Calculating this expectation is more involved than calculating the expected improvement, but nevertheless can also be done analytically in terms of the normal pdf and normal cdf. This is described in more detail in the Derivations and Proofs section.

The knowledge-gradient algorithm is then the one that chooses the point to sample next that maximizes the KG factor,

$$\operatorname*{argmax}_x \mathrm{KG}_n(x).$$

The KG factor for a one-dimensional optimization problem with noise is pictured in Fig. 3.6. We see a similar tradeoff between exploration and exploitation, where the KG factor favors measuring points with a large $\mu_n(x)$ and a large $\sigma_n(x)$. We also see local minima of the KG factor at points where we previously evaluated, just as with the expected improvement, but because there is noise in our samples, the value at these points is not 0—indeed, when there is noise, it may be useful to sample repeatedly at a point.

Choice of A_n and A_{n+1}

Recall that the KG factor depends on the choice of the sets A_n and A_{n+1}, through the dependence of μ_n^* and μ_{n+1}^* on these sets. Typically, if we choose these sets to contain more elements, then we allow μ_n^* and μ_{n+1}^* to range over a larger portion of the space, and we allow the KG factor calculation to more accurately approximate the value that would result if we allowed ourself to implement the best option. However, as we increase the size of these sets, computing the KG factor is slower, making implementation of the KG method more computationally intensive.

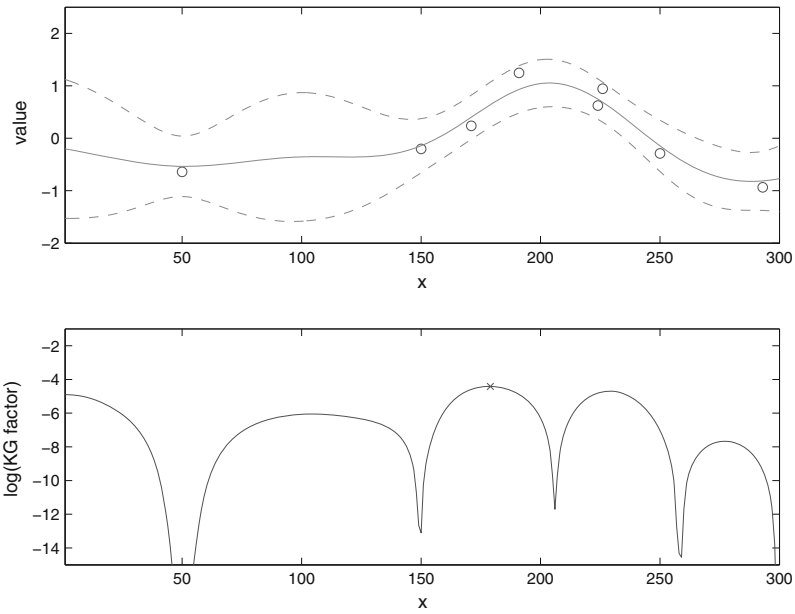

Fig. 3.6 *Upper panel* shows the posterior distribution in a problem with independent normal homoscedastic noise and a one-dimensional input space, where the *circles* are previously measured points, the *solid line* is the posterior mean $\mu_n(x)$, and the *dashed lines* are at $\mu_n(x) \pm 2\sigma_n(x)$. *Lower panel* shows the natural logarithm of the knowledge-gradient factor $\mathrm{KG}(x)$ computed from this posterior distribution, where $A_n = A_{n+1}$ are the discrete grid $\{1, \ldots, 300\}$. An "x" is marked at the point with the largest KG factor, which is where the KG algorithm would evaluate next

For applications with a finite A, [24] proposed setting $A_{n+1} = A_n = A$, which was seen to require fewer function evaluations to find points with large f, in comparison with expected improvement on noise-free problems, and in comparison with another Bayesian optimization method, sequential kriging optimization (SKO) [30] on noisy problems. However, the computation and memory required grows rapidly with the size of A, and is typically not feasible when A contains more than 10,000 points.

For large-scale applications, [25] proposed setting $A_{n+1} = A_n = \{x_{1:n+1}\}$ in (3.18), and called the resulting quantity the *approximate knowledge gradient (AKG)*, observing that this choice maintains computational tractability as A grows, but also offers good performance. This algorithm is implemented in the DiceKriging package [31].

Finally, in noise-free problems (but not in problems with noise), setting $A_{n+1} = \{x_{1:n+1}\}$ and $A_n = \{x_{1:n}\}$ recovers expected improvement.

3.4.3 Going Beyond One-Step Analyses, and Other Methods

Both expected improvement and the knowledge-gradient method are designed to be optimal, in the special case where we will take just one more function evaluation and then choose a final solution. They are not, however, known to be optimal for the more general case in which we will take multiple measurements, which is the way they are used in practice.

The optimal algorithm for this more general setting is understood to be the solution to a partially observable Markov decision process, but actually computing the optimal solution using this understanding is intractable using current methods [32]. Some work has been done toward the goal of developing such an optimal algorithm [33], but computing the optimal algorithm remains out of reach. Optimal strategies have been computed for other closely related problems in optimization of expensive noisy functions, including stochastic root-finding [34], multiple comparisons with a standard [35], and small instances of discrete noisy optimization with normally distributed noise (also called "ranking and selection") [36].

Expected improvement and the knowledge gradient are both special cases of the more general concept of value of information, or expected value of sample information (EVSI) [37], as they calculate the expected reward of a final implementation decision as a function of the posterior distribution resulting from some information, subtract from this the expected reward that would result from not having the information, and then take the expectation of this difference with respect to the information itself.

Many other Bayesian optimization methods have been proposed. A few of these methods optimize the value of information, but are calculated using different assumptions than those used to derive expected improvement or value of information. A larger number of these methods optimize quantities that do not correspond to a value of information, but are derived using analyses that are similar in spirit. These include methods that optimize the probability of improvement [1, 38, 39], the entropy of the posterior distribution on the location of the maximum [40], and other composite measures involving the mean and the standard deviation of the posterior [30].

Other Bayesian optimization methods are designed for problem settings that do not match the assumptions made in this tutorial. These include [41–43], which consider multiple objectives; [6, 44–46], which consider multiple simultaneous function evaluations; [47–49], which consider objective functions that can be evaluated with multiple fidelities and costs; [50], which considers Bernoulli outcomes, rather than normally distributed ones; [51], which considers expensive-to-evaluate inequality constraints; and [52], which considers optimization over the space of small molecules.

3.5 Software

There are a number of excellent software packages, both freely available and commercial, that implement the methods described in this chapter, and other similar methods.

- Metrics Optimization Engine (MOE), an open-source code in C++ and Python, developed by the authors and engineers at Yelp. http://yelp.github.io/MOE/,
- Spearmint, an open-source code in Python, implementing algorithms described in [6]. https://github.com/JasperSnoek/spearmint
- DiceKriging and DiceOptim, an open-source R package that implements expected improvement, the approximate knowledge-gradient method, and a variety of algorithms for parallel evaluations. An overview is provided in [31]. http://cran.r-project.org/web/packages/DiceOptim/index.html,
- TOMLAB, a commercial package for MATLAB. http://tomopt.com/tomlab/
- matlabKG, an open-source research code that implements the discrete knowledge-gradient method for small-scale problems. http://people.orie.cornell.edu/pfrazier/src.html

A list of software packages focused on Gaussian process regression (but not Bayesian optimization) may be found at http://www.gaussianprocess.org/.

3.6 Conclusion

We have presented Bayesian optimization, including Gaussian process regression, the expected improvement method, and the knowledge-gradient method. In making this presentation, we wish to emphasize that this approach to materials design acknowledges the inherent uncertainty in statistical prediction and seeks to guide experimentation in a way that is robust to this uncertainty. It is inherently iterative, and does not seek to circumvent the fundamental trial-and-error process.

This is in contrast with another approach to informatics in materials design, which holds the hope that predictive methods can short-circuit the iterative loop entirely. In this alternative view of the world, one hopes to create extremely accurate prediction techniques, either through physically-motivated *ab initio* calculations, or using data-driven machine learning approaches, that are so accurate that one can rely on the predictions alone rather than on physical experiments. If this can be achieved, then we can search over materials designs *in silico*, find those designs that are predicted to perform best, and test those designs alone in physical experiments.

For this approach to be successful, one must have extremely accurate predictions, which limits its applicability to settings where this is possible. We argue that, in contrast, predictive techniques can be extremely powerful even if they are not perfectly accurate, as long as they are used in a way that acknowledges inaccuracy, builds in robustness, and reduces this inaccuracy through an iterative dialog with physical

reality mediated by physical experiments. Moreover, we argue that mathematical techniques like Bayesian optimization, Bayesian experimental design, and optimal learning provide us the mathematical framework for accomplishing this goal in a principled manner, and for using our power to predict as effectively as possible.

Acknowledgments Peter I. Frazier was supported by AFOSR FA9550-12-1-0200, AFOSR FA9550-15-1-0038, NSF CAREER CMMI-1254298, NSF IIS-1247696, and the ACSF's AVF. Jialei Wang was supported by AFOSR FA9550-12-1-0200.

Derivations and Proofs

This section contains derivations and proofs of equations and theoretical results found in the main text.

Proof of Proposition 1

Proof Using Bayes' rule, the conditional probability density of $\theta_{[2]}$ at a point $u_{[2]}$ given that $\theta_{[1]} = u_{[1]}$ is

$$p(\theta_{[2]} = u_{[2]} \mid \theta_{[1]} = u_{[1]}) = \frac{p(\theta_{[1]} = u_{[1]}, \theta_{[2]} = u_{[2]})}{p(\theta_{[1]} = u_{[1]})} \propto p(\theta_{[1]} = u_{[1]}, \theta_{[2]} = u_{[2]})$$
$$\propto \exp\left(-\frac{1}{2}\begin{bmatrix} u_{[1]} - \mu_{[1]} \\ u_{[2]} - \mu_{[2]} \end{bmatrix}^T \begin{bmatrix} \Sigma_{[1,1]} & \Sigma_{[1,2]} \\ \Sigma_{[2,1]} & \Sigma_{[2,2]} \end{bmatrix}^{-1} \begin{bmatrix} u_{[1]} - \mu_{[1]} \\ u_{[2]} - \mu_{[2]} \end{bmatrix}\right). \quad (3.19)$$

To deal with the inverse matrix in this expression, we use the following identity for inverting a block matrix: the inverse of the block matrix $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$, where both A and D are invertible square matrices, is

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} (A - BD^{-1}C)^{-1} & -(A - BD^{-1}C)^{-1}BD^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix}. \quad (3.20)$$

Applying (3.20) to (3.19), and using a bit of algebraic manipulation to get rid of constants, we have

$$p(\theta_{[2]} = u_{[2]} \mid \theta_{[1]} = u_{[1]}) \propto \exp\left(-\frac{1}{2}(u_{[2]} - \mu^{\text{new}})^T(\Sigma^{\text{new}})^{-1}(u_{[2]} - \mu^{\text{new}})\right), \quad (3.21)$$

where $\mu^{\text{new}} = \mu_{[2]} - \Sigma_{[2,1]}\Sigma_{[1,1]}^{-1}(u_{[1]} - \mu_{[1]})$ and $\Sigma^{\text{new}} = \Sigma_{[2,2]} - \Sigma_{[2,1]}\Sigma_{[1,1]}^{-1}\Sigma_{[1,2]}$.

We see that (3.21) is simply the unnormalized probability density function of a normal distribution. Thus the conditional distribution of $\theta_{[2]}$ given $\theta_{[1]} = u_{[1]}$ is multivariate normal, with mean μ^{new} and covariance matrix Σ^{new}.

Derivation of Equation (3.16)

Since $f(x) \sim \text{Normal}(\mu_n(x), \sigma_n^2(x))$, the probability density of $f(x)$ is $p(f(x) = z) = \frac{1}{\sqrt{2\pi}} \exp\left((z - \mu_n(x))^2/2\sigma_n(x)^2\right)$. We use this to calculate $\text{EI}(x)$:

$$
\begin{aligned}
\text{EI}(x) &= E_n[(f(x) - f_n^*)^+] \\
&= \int_{f_n^*}^{\infty} (z - f_n^*) \frac{1}{\sqrt{2\pi}\sigma_n(x)} e^{\frac{-(z-\mu_n(x))^2}{2\sigma_n^2(x)}} dz \\
&= \int_{f_n^*}^{\infty} z \frac{1}{\sqrt{2\pi}\sigma_n(x)} e^{\frac{-(z-\mu_n(x))^2}{2\sigma_n^2(x)}} dz - f_n^* \left(1 - \Phi\left(\frac{f_n^* - \mu_n(x)}{\sigma_n(x)}\right)\right) \\
&= \int_{f_n^*}^{\infty} (\mu_n(x) + (z - \mu_n(x))) \frac{1}{\sqrt{2\pi}\sigma_n(x)} e^{\frac{-(z-\mu_n(x))^2}{2\sigma_n^2(x)}} dz - f_n^* \left(1 - \Phi\left(\frac{f_n^* - \mu_n(x)}{\sigma_n(x)}\right)\right) \\
&= \int_{f_n^*}^{\infty} (z - \mu_n(x)) \frac{1}{\sqrt{2\pi}\sigma_n(x)} e^{\frac{-(z-\mu_n(x))^2}{2\sigma_n^2(x)}} dz + (\mu_n(x) - f_n^*) \left(1 - \Phi\left(\frac{f_n^* - \mu_n(x)}{\sigma_n(x)}\right)\right) \\
&= \sigma_n(x) \frac{1}{\sqrt{2\pi}} e^{\frac{-(f_n^* - \mu_n(x))^2}{2\sigma_n(x)^2}} + (\mu_n(x) - f_n^*) \left(1 - \Phi\left(\frac{f_n^* - \mu_n(x)}{\sigma_n(x)}\right)\right) \\
&= (\mu_n(x) - f_n^*) \left(1 - \Phi\left(\frac{f_n^* - \mu_n(x)}{\sigma_n(x)}\right)\right) + \sigma_n(x) \varphi\left(\frac{f_n^* - \mu_n(x)}{\sigma_n(x)}\right) \\
&= (\mu_n(x) - f_n^*) \Phi\left(\frac{\mu_n(x) - f_n^*}{\sigma_n(x)}\right) + \sigma_n(x) \varphi\left(\frac{\mu_n(x) - f_n^*}{\sigma_n(x)}\right).
\end{aligned}
$$

Calculation of the KG factor

The KG factor (3.18) is calculated by first considering how the quantity $\mu_{n+1}^* - \mu_n^*$ depends on the information that we have at time n, and the additional datapoint that we will obtain, y_{n+1}.

First observe that $\mu_{n+1}^* - \mu_n^*$ is a deterministic function of the vector $[\mu_{n+1}(x) : x \in A_{n+1}]$ and other quantities that are known at time n. Then, by applying the analysis in Sect. 3.3.5, but letting the posterior given $x_{1:n}, y_{1:n}$ play the role of the prior, we obtain the following version of (3.10), which applies to any given x,

$$
\mu_{n+1}(x) = \mu_n(x) + \frac{\Sigma_n(x, x_{n+1})}{\Sigma_n(x_{n+1}, x_{n+1}) + \lambda^2} (y_{n+1} - \mu_n(x_{n+1})). \tag{3.22}
$$

In this expression, $\mu_n(\cdot)$ and $\Sigma_n(\cdot,\cdot)$ are given by (3.13) and (3.14).

We see from this expression that $\mu_{n+1}(x)$ is a linear function of y_{n+1}, with an intercept and a slope that can be computed based on what we know after the nth measurement.

We will calculate the distribution of y_{n+1}, given what we have observed at time n. First, $f(x_{n+1})|x_{1:n}, y_{1:n} \sim \text{Normal}\left(\mu_n(x_{n+1}), \Sigma_n(x_{n+1}, x_{n+1})\right)$. Since $y_{n+1} = f(x_{n+1}) + \varepsilon_{n+1}$, where ε_{n+1} is independent with distribution $\varepsilon_{n+1} \sim \text{Normal}(0, \lambda^2)$, we have

$$y_{n+1}|x_{1:n}, y_{1:n} \sim \text{Normal}\left(\mu_n(x_{n+1}), \Sigma_n(x_{n+1}, x_{n+1}) + \lambda^2\right).$$

Plugging the distribution of y_{n+1} into (3.22) and doing some algebra, we have

$$\mu_{n+1}(x)|x_{1:n}, y_{1:n} \sim \text{Normal}\left(\mu_n(x), \widetilde{\sigma}^2(x, x_{n+1})\right),$$

where $\widetilde{\sigma}(x, x_{n+1}) = \frac{\Sigma_n(x, x_{n+1})}{\sqrt{\Sigma_n(x_{n+1}, x_{n+1}) + \lambda^2}}$. Moreover, we can write $\mu_{n+1}(x)$ as

$$\mu_{n+1}(x) = \mu_n(x) + \widetilde{\sigma}(x, x_{n+1})Z,$$

where $Z = (y_{n+1} - \mu_n(x_{n+1}))/\sqrt{\Sigma_n(x_{n+1}, x_{n+1}) + \lambda^2}$ is a standard normal random variable, given $x_{1:n}$ and $y_{1:n}$.

Now (3.18) becomes

$$\text{KG}_n(x) = E_n\left[\max_{x' \in A_{n+1}} \mu_n(x') + \widetilde{\sigma}(x', x_{n+1})Z \mid x_{n+1} = x\right] - \mu_n^*.$$

Thus, computing the KG factor comes down to being able to compute the expectation of the maximum of a collection of linear functions of a scalar normal random variable. Algorithm 2 of [24], with software provided as part of the matlabKG library [53], computes the quantity

$$h(a, b) = \mathbb{E}\left[\max_{i=1,\ldots,|a|} (a_i + b_i Z)\right] - \max_{i=1,\ldots,|a|} a_i$$

for arbitrary equal-length vectors a and b. Using this ability, and letting $\mu_n(A_{n+1})$ be the vector $[\mu_n(x') : x' \in A_{n+1}]$ and $\widetilde{\sigma}(A_{n+1}, x)$ be the vector $[\widetilde{\sigma}(x', x) : x' \in A_{n+1}]$, we can write the KG factor as

$$\text{KG}_n(x) = h(\mu_n(A_{n+1}), \widetilde{\sigma}(A_{n+1}, x)) + \left[\max(\mu_n(A_{n+1})) - \mu_n^*\right].$$

If $A_{n+1} = A_n$, as it is in the versions of the knowledge-gradient method proposed in [24, 25], then the last term $\max(\mu_n(A_{n+1})) - \mu_n^*$ is equal to 0 and vanishes.

References

1. H.J. Kushner, A new method of locating the maximum of an arbitrary multi- peak curve in the presence of noise. J. Basic Eng. **86**, 97–106 (1964)
2. J. Mockus, *Bayesian Approach to Global Optimization: Theory and Applications* (Kluwer Academic, Dordrecht, 1989)
3. J. Mockus, V. Tiesis, A. Zilinskas, The application of Bayesian methods for seeking the extremum, in *Towards Global Optimisation*, ed. by L.C.W. Dixon, G.P. Szego, vol. 2 (Elsevier Science Ltd., North Holland, Amsterdam, 1978), pp. 117–129
4. D.R. Jones, M. Schonlau, W.J. Welch, Efficient Global Optimization of Expensive Black-Box Functions. J. Global Optim. **13**(4), 455–492 (1998)
5. A. Booker, J. Dennis, P. Frank, D. Serafini, V. Torczon, M.W. Trosset, Optimization using surrogate objectives on a helicopter test example. Prog. Syst. Control Theor. **24**, 49–58 (1998)
6. J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms. in *Advances in Neural Information Processing Systems*, pp. 2951–2959 (2012)
7. E. Brochu, M. Cora, N. de Freitas, *A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning*. Technical Report TR-2009-023, Department of Computer Science, University of British Columbia, November 2009
8. A. Forrester, A. Sobester, A. Keane, *Engineering Design Via Surrogate Modelling: A Practical Guide* (Wiley, West Sussex, UK, 2008)
9. T.J. Santner, B.W. Willians, W. Notz, *The Design and Analysis of Computer Experiments* (Springer, New York, 2003)
10. M.J. Sasena, *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. Ph.D. thesis, University of Michigan (2002)
11. D.G. Kbiob, A statistical approach to some basic mine valuation problems on the witwatersrand. J. Chem. Metall. Min. Soc. S. Afr. (1951)
12. G. Matheron, *The theory of regionalized variables and its applications*, vol 5. École national supérieure des mines (1971)
13. N. Cressie, The origins of kriging. Math. Geol. **22**(3), 239–252 (1990)
14. C.E. Rasmussen, C.K.I. Williams, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, MA, 2006)
15. C.E. Rasmussen (2011), http://www.gaussianprocess.org/code, Accessed 15 July 2015
16. A.B. Gelman, J.B. Carlin, H.S. Stern, D.B. Rubin, *Bayesian Data Analysis* (CRC Press, Boca Raton, FL, second edition, 2004)
17. J.O. Berger, *Statistical decision theory and Bayesian analysis* (Springer-Verlag, New York, second edition) (1985)
18. B. Ankenman, B.L. Nelson, J. Staum, Stochastic kriging for simulation metamodeling. Oper. Res. **58**(2), 371–382 (2010)
19. P.W. Goldberg, C.K.I. Williams, C.M. Bishop, Regression with input-dependent noise: a gaussian process treatment. *Advances in neural information processing systems*, p. 493–499 (1998)
20. K. Kersting, C. Plagemann, P. Pfaff, W. Burgard, Most likely heteroscedastic Gaussian process regression. In *Proceedings of the 24th international conference on Machine learning*, ACM, pp. 393–400 (2007)
21. C. Wang, *Gaussian Process Regression with Heteroscedastic Residuals and Fast MCMC Methods*. Ph.D. thesis, University of Toronto (2014)
22. P.I. Frazier, J. Xie, S.E. Chick, Value of information methods for pairwise sampling with correlations, in *Proceedings of the 2011 Winter Simulation Conference*, ed. by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, M. Fu (Institute of Electrical and Electronics Engineers Inc, Piscataway, New Jersey, 2011), pp. 3979–3991
23. S. Sankaran, A.L. Marsden, The impact of uncertainty on shape optimization of idealized bypass graft models in unsteady flow. *Physics of Fluids (1994-present)*, 22(12):121–902 (2010)

24. P.I. Frazier, W.B. Powell, S. Dayanik, The knowledge gradient policy for correlated normal beliefs. INFORMS J. Comput. **21**(4), 599–613 (2009)
25. W. Scott, P.I. Frazier, W.B. Powell, The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. SIAM J. Optim. **21**(3), 996–1026 (2011)
26. L.P. Kaelbling, *Learning in Embedded Systems* (MIT Press, Cambridge, MA, 1993)
27. R.S. Sutton, A.G. Barto, *Reinforcement Learning* (The MIT Press, Cambridge, Massachusetts, 1998)
28. J. Gittins, K. Glazebrook, R. Weber. *Multi-armed Bandit Allocation Indices*. Wiley, 2nd edition (2011)
29. A. Mahajan, D. Teneketzis, Multi-armed bandit problems. In D. Cochran A. O. Hero III, D. A. Castanon, K. Kastella, (Ed.). *Foundations and Applications of Sensor Management*. Springer-Verlag (2007)
30. D. Huang, T.T. Allen, W.I. Notz, N. Zeng, Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. J. Global Optim. **34**(3), 441–466 (2006)
31. O. Roustant, D. Ginsbourger, Y. Deville, Dicekriging, diceoptim: two R packages for the analysis of computer experiments by kriging-based metamodelling and optimization. J. Stat. Softw. **51**(1), p. 54 (2012)
32. P.I. Frazier, *Learning with Dynamic Programming*. John Wiley and Sons (2011)
33. D. Ginsbourger, R. Riche, Towards gaussian process-based optimization with finite time horizon. *mODa 9–Advances in Model-Oriented Design and Analysis*, p. 89–96 (2010)
34. R. Waeber, P.I. Frazier, S.G. Henderson, Bisection search with noisy responses. SIAM J. Control Optim. **51**(3), 2261–2279 (2013)
35. J. Xie, P.I. Frazier, Sequential bayes-optimal policies for multiple comparisons with a known standard. Oper. Res. **61**(5), 1174–1189 (2013)
36. P.I. Frazier, Tutorial: Optimization via simulation with bayesian statistics and dynamic programming, in *Proceedings of the 2012 Winter Simulation Conference Proceedings*, ed. by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, A.M. Uhrmacher (Institute of Electrical and Electronics Engineers Inc., Piscataway, New Jersey, 2012), pp. 79–94
37. R.A. Howard, Information Value Theory. Syst. Sci. Cybern. IEEE Trans. **2**(1), 22–26 (1966)
38. C.D. Perttunen, A computational geometric approach to feasible region division inconstrained global optimization. in *Proceedings of 1991 IEEE International Conference on Systems, Man, and Cybernetics, 1991.'Decision Aiding for Complex Systems*, pp. 585–590 (1991)
39. B.E. Stuckman, A global search method for optimizing nonlinear systems. Syst. Man Cybern. IEEE Trans. **18**(6), 965–977 (1988)
40. J. Villemonteix, E. Vazquez, E. Walter, An informational approach to the global optimization of expensive-to-evaluate functions. J. Global Optim. **44**(4), 509–534 (2009)
41. D.C.T. Bautista, *A Sequential Design for Approximating the Pareto Front using the Expected Pareto Improvement Function*. Ph.D. thesis, The Ohio State University (2009)
42. P.I. Frazier, A.M. Kazachkov, Guessing preferences: a new approach to multi-attribute ranking and selection, in *Proceedings of the 2011 Winter Simulation Conference*, ed. by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, M. Fu (Institute of Electrical and Electronics Engineers Inc, Piscataway, New Jersey, 2011), pp. 4324–4336
43. J. Knowles, ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. Evol. Comput. IEEE Trans. **10**(1), 50–66 (2006)
44. S.C. Clark, J. Wang, E. Liu, P.I. Frazier, Parallel global optimization using an improved multipoints expected improvement criterion (working paper, 2014)
45. D. Ginsbourger, R. Le Riche, L. Carraro, A multi-points criterion for deterministic parallel global optimization based on kriging. In *International Conference on Nonconvex Programming, NCP07*, Rouen, France, December 2007
46. D. Ginsbourger, R. Le Riche, and L. Carraro, Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, Springer, vol. 2, p. 131–162 (2010)

47. A.I.J. Forrester, A. Sóbester, A.J. Keane, Multi-fidelity optimization via surrogate modelling. Proc. R. Soc. A: Math. Phys. Eng. Sci. **463**(2088), 3251–3269 (2007)
48. P.I. Frazier, W.B. Powell, H.P. Simão, Simulation model calibration with correlated knowledge-gradients, in *Proceedings of the 2009 Winter Simulation Conference Proceedings*, ed. by M.D. Rossetti, R.R. Hill, B. Johansson, A. Dunkin, R.G. Ingalls (Institute of Electrical and Electronics Engineers Inc, Piscataway, New Jersey, 2009), pp. 339–351
49. D. Huang, T.T. Allen, W.I. Notz, R.A. Miller, Sequential kriging optimization using multiple-fidelity evaluations. Struct. Multi. Optim. **32**(5), 369–382 (2006)
50. J. Bect, D. Ginsbourger, L. Li, V. Picheny, E. Vazquez, Sequential design of computer experiments for the estimation of a probability of failure. Stat. Comput. **22**(3), 773–793 (2012)
51. J.R. Gardner, M.J. Kusner, Z. Xu, K. Weinberger, J.P. Cunningham, Bayesian optimization with inequality constraints. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 937–945 (2014)
52. D.M. Negoescu, P.I. Frazier, W.B. Powell, The knowledge gradient algorithm for sequencing experiments in drug discovery. INFORMS J. Comput. 23(1) (2011)
53. P.I. Frazier (2009–2010), http://people.orie.cornell.edu/pfrazier/src.html