Worlds Collide through Gaussian Processes: Statistics, Geoscience and Mathematical Programming

Ryan B. Christianson

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Statistics

Robert B. Gramacy, Chair
Leanna L. House
Ryan M. Pollyea
Jennifer H. Van Mullekom

April 28, 2023
Blacksburg, Virginia

Keywords: Kriging, Mining, Bayesian Optimization, Robust, Adversary

Worlds Collide through Gaussian Processes: Statistics, Geoscience and Mathematical Programming

Ryan B. Christianson

(ABSTRACT)

Gaussian process (GP) regression is the canonical method for nonlinear spatial modeling among the statistics and machine learning communities. Geostatisticians use a subtly different technique known as kriging. I shall highlight key similarities and differences between GPs and kriging through the use of large scale gold mining data. Most importantly GPs are largely hands-off, automatically learning from the data whereas kriging requires an expert human in the loop to guide analysis. To emphasize this, I show an imputation method for left censored values frequently seen in mining data. Oftentimes geologists ignore censored values due to the difficulty of imputing with kriging, but GPs execute imputation with relative ease leading to better estimates of the gold surface. My hope is that this research can serve as a springboard to encourage the mining community to consider using GPs over kriging for diverse utility after GP model fitting. Another common use of GPs that would be inefficient for kriging is Bayesian Optimization (BO). Traditionally BO is designed to find a global optima by sequentially sampling from a function of interest using an acquisition function. When two or more local or global optima of the function of interest have similar objective values, it often makes some sense to target the more "robust" solution with a wider domain of attraction. However, traditional BO weighs these solutions the same, favoring whichever has a slightly better objective value. By combining the idea of expected improvement (EI) from the BO community with mathematical programming's concept of an adversary, I introduce a novel algorithm to target robust solutions called robust expected improvement (REI). The adversary penalizes "peaked" areas of the objective function making those values appear less desirable. REI performs acquisitions using EI on the adversarial space yielding data sets focused on the robust solution that exhibit EI's already proven excellent balance of exploration and exploitation.

Worlds Collide through Gaussian Processes: Statistics, Geoscience and Mathematical Programming

Ryan B. Christianson

(GENERAL AUDIENCE ABSTRACT)

Since its origins in the 1940's, spatial statistics modeling has adapted to fit different communities. The geostatistics community developed with an emphasis on modeling mining operations and has further evolved to cover a slew of different applications largely focused on two or three physical dimensions. The computer experiments community developed later when these physical experiments started moving into the virtual realm with advances in computer technology. While birthed from the same foundation, computer experimenters often look at ten or sometimes even higher dimension problems. Due to these differences among others, each community tailored their methods to best fit their common problems. My research compares the modern instantiations of the differing methodology on two sets of real gold mining data. Ultimately, I prefer the computer experiments methods for their ease of adaptation to downstream tasks at no cost to model performance. A statistical model is almost never a standalone development; it is created with a specific goal in mind. The first case I show of this is "imputation" of mining data. Mining data often have a detection threshold such that any observation with very small mineral concentrations are recorded at the threshold. Frequently, geostatisticians simply throw out these observations because they cause problems in modeling. Statisticians try to use the information that there is a low concentration combined with the rest of the fully observed data to derive a best guess at the concentration of thresholded locations. Under the geostatistics framework, this is cumbersome, but the computer experiments community consider imputation an easy extension. Another common model task is creating an experiment to best learn a surface. The surface may be a gold deposit on Earth or an unknown virtual function or anything measurable really. To do this, computer experimenters often use "active learning" by sampling one point at a time, using that point to generate a better informed model which suggests a new point to sample, repeating until a satisfactory number of points are sampled. Geostatisticians often prefer "one-shot" experiments by deciding all samples prior to collecting any. Thus the geostatistics framework is not appropriate for active learning. Active learning tries to

find the "best" location of the surface with either the maximum or minimum response. I adapt this problem to redefine best to find a "robust" location where the response does not change much even if the location is not perfectly specified. As an example, consider setting operating conditions for a factory. If locations produce a similar amount of product, but one needs an exact pressure setting or else it blows up the factory, the other is certainly preferred. To design experiments to find robust locations, I borrow ideas from the mathematical programming community to develop a novel method for robust active learning.

Acknowledgments

Firstly, I would like to thank my advisor, Bobby Gramacy, for his patience and guidance throughout my time at Virginia Tech (VT). It is safe to say that may experience would have been completely different without you. Each of you on my committee also had a great impact on my time at VT. Leanna House, thank you for teaching my first VT statistics course and working through the stress of the qualifier exams. Ryan Pollyea, thank you for collaborating on my first primary author paper submission and giving the geoscience perspective on my statistical work. Jennifer Van Mullekom, thank you so much for mentoring me throughout in statistical collaboration. The statistical applications and innovations group was one of the primary reasons I came to VT and I love that part of the curriculum. I hope I can become even a fraction as good at statistical collaboration as you. A big thank you to the whole of the VT statistics department, but in particular, thank you to Scotland Leman, Sierra Merkes and Erica Porter for helping through the tougher times of the PhD process; you all somehow always knew the right thing to say to get back on track.

Thank you to my family for supporting me along the way. Of course to my parents, Fred and Jennifer Christianson, thank you always showing me the value of education. To my brother, Michael Christianson, thank you for showing me that this beast of a PhD process can be conquered and providing guidance throughout. Thank you to the extended Christianson and Walter family for all the love and support.

Finally, thank you to the Schwartz family and extensions. I greatly appreciate all of your interest in the process and your support. Of course in particular, I have to thank Marissa Schwartz. This roller coaster ride would have had many more downs without you. Thank you for always being there and listening to my at times incomprehensible statistical ramblings. I am so glad that I get to traverse this Markov process called life with you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The field of statistics has often evolved relative to synergies with other fields. Take for example the origins of epidemiology being a hybrid between medical science and statistics from John Snow and the cholera outbreak of 1854 [69]. Since then, the field has grown in many ways to the founding of the statistics department at Virginia Tech being essentially an offshoot of the agriculture and economics departments in the 1940s [3]. Moving into the modern era, the exact identity of statistics is not set. There are still many fields that are seamingly synonomous, yet differ subtly depending on who you ask, such as: statistics, data science, machine learning, artificial intelligence, etc. Furthermore, statistics applied to fields often further differentiate themselves such as biostatistics, econometrics, geostatistics, etc. For better or worse, statistics has ingrained itself into collaboration between other fields. Personally, I embrace this and promote the umbrella term of "statistical engineering" [2] being the application to any field with the science of "statistics." To quote a legendary statistician, John Tukey, "the best thing about being a statistician is that you get to play in everyone's backyard." I wholeheartedly agree and enjoy putting my research on the boundaries between fields, capitalizing on blending concepts from various communities to create

novel ideas. Throughout this dissertation, I hope to close the gaps between some of the before mentioned fields by introducing novel methodology, taking the best from all worlds, in particular targeting statistics through the computer experiments community, machine learning, mathematical programming and geostatistics, extending into computational geology.

The main unifier through this dissertion is the use of Gaussian process (GP) regression or kriging. Both GPs and kriging originate from the mining analytics of Danie Krige [57] and Henri de Wijs [27] and the subsequent work of Matheron [66]. Similar ideas were developed independently around the same time to aid the early analysis of computer simulation experiments, like those conducted in the study of nuclear weapons and energy, however (unclassified) publications did not appear until later (e.g., [80]). The spatial statistics community was responsible for much of the subsequent advances in kriging methodology, particularly spanning research in the 70s, 80s and early 90s (e.g., [23]), and software for kriging in use commercially (e.g., `LeapFrog`, `Vulcan`, `Surfer`, etc.) and academically (e.g., `GSLIB`) in mining today. More recently, the surrogate modeling of computer experiments [38] and machine learning [77], have pushed the boundaries of fidelity and computational tractability as modeling ambition and scale of data collection continue to grow. These literatures have converged around the nomenclature of GP regression as a generative framework for the kinds of data and procedures involved in kriging, but with a more cohesive and flexible approach to inference, approximation and automation based upon the likelihood, which is the foundation to modern statistcal learning.

With innovations from both sides, the modern kriging community is somewhat detached from the GP community. Anecdotally, I have met computational geologists who have never heard of GPs, but are master krigers and similarly many statisticians who are experts at GPs have little to no knowledge of kriging. However, these methods are largely similar and were born from the same idea! These are the exact hard lines that I hope to erase, enriching

both communities by showing the advantages and drawbacks of the other. Throughout this dissertion, I look at GPs through the lens of the computer experiments community mostly because that was how I learned it first. However, I include frequent references to the geostatistics kriging when helpful. My initial focus in Chapter 3 will be on comparisons between GPs and kriging using mining data, but that will extend into considering a common extension to the GP framework to active learning in Chapter 4 that would be cumbersome to say the least using the kriging framework.

This dissertation is a compilation of two main projects with a central unifier of GPs. Chapter 3 covers the first which has already been somewhat introduced as the comparison between the computer experiments community and the geostatistics community with an emphasis on mining data, requiring customized features to handle the "quirks" of the data. Many more specifics to come. The second project in Chapter 4 continues with the GP framework and you will hopefully understand why the kriging framework is infeasible after Chapter 3. The central idea lies in combining methods from the computer experiments community with mathematical programming concepts to introduce a novel algorithm for experimental design. For now, I will leave it at that as it will not be discussed until Chapter 4 which will have a more comprehensive introduction of important topics with it. Those main projects will be Chapters 3 and 4, but before those, Chapter 2 gives a review of the foundational components used in subsequent chapters. Finally, Chapter 5 proposes further extensions to Chapters 3 and 4 and provides some ideas of how to combine their ideas together.

Chapter 2

Review

I begin by introducing fundamental concepts necessary to understand my contributions: Gaussian process (GP) regression, kriging and Bayesian optimaziton (BO). This chapter also contains foreshadowing of example functions utilized through the remaining chapters.

2.1 Gaussian Process Regression

Suppose one wishes to model a function $f : \mathbb{R}^d \to \mathbb{R}$ with a limited number of noisy evaluations $y_i = f(x_i) + \epsilon_i$, for $i = 1, \ldots, N$. Let X_N be an $N \times d$ matrix formed with d-dimensional $x_i^\top$ in each of its rows. Similarly combine scalar outputs y_i into an N-vector Y_N. Throughout this document, I privilege an input–output (x, y) notational scheme in keeping with the vast majority of machine learning and statistical literature on regression, nonparametric and nonlinear or otherwise. In many geospatial contexts [e.g., 5], where f might be an environmental or geological process, it is common to use s_i for (spatial) input sites and $z_i = Z(s_i)$, among many alternatives, for the response. I think this unproductively biases thinking towards $d = 2$-dimensional point-referenced (latitude and longitude) data, whereas these methods

can be applied much more widely than that. Machine learning [e.g., 77] and computer surrogate modeling [e.g., 38] applications are typically in higher input dimension, and one of my goals is to introduce this way of thinking into the mining literature.

A common nonparametric model for such data is a Gaussian process (GP), which assumes that outputs Y_N follow a multivariate normal (MVN) distribution. In GP spatial modeling contexts, but also in machine learning and beyond, inputs X_N are primarily involved in the specification of the MVN covariance $\Sigma_N \equiv \Sigma(X_N, X_N)$ with a form for $\Sigma(\cdot, \cdot)$ that inverts Euclidean distances between its arguments. For example,

$$Y_N \sim \mathcal{N}_N(0, \Sigma_N), \quad \text{where} \quad \Sigma_N^{ij} \quad \text{follows} \quad S\left(\frac{1}{\text{Dist}(x_i, x_j)}\right) \quad \text{for some decreasing } S. \tag{2.1}$$

In Eq. (2.1) I am being deliberately imprecise about the form of Σ_N, a topic I shall detail shortly in Section 2.2. For now, simply suppose correlation in outputs decays as a function of distance in inputs: $\mathbb{C}\text{orr}(y_i, y_j) < \mathbb{C}\text{orr}(y_i, y_k)$ if x_i is "closer" to x_k than it is to x_j. I am also using a zero mean specification, so that all of the modeling "action" is in the covariance. Extensions abound.

Although a Bayesian interpretation is not essential in characterizing GP regression, Eq. (2.1) can be said to specify a prior over (noisy evaluations of) functions like f, abstracting as $Y_N \sim \text{GP}$. Choices for the mean (0) and variance (Σ) determine the modeling properties of f like its smoothness and wiggliness. I shall largely leave those properties to the references, except as relevant to particular choices for $\Sigma(\cdot, \cdot)$, again in Section 2.2. Then, if N' new locations $\mathcal{X}$ come along where one does not yet have observations, $Y(\mathcal{X})$, one can summarize the understanding for those in light of the (training) data one does have – a predictive distribution – through the lens of posterior conditioning: $Y(\mathcal{X}) \mid Y_N$. First, extend the GP

prior to cover $Y(\mathcal{X})$ jointly with Y_N:

$$\begin{bmatrix} Y_N \\ Y(\mathcal{X}) \end{bmatrix} \sim \mathcal{N}_{N+N'}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_N & \Sigma(X_N, \mathcal{X}) \\ \Sigma(\mathcal{X}, X_N) & \Sigma(\mathcal{X}, \mathcal{X}) \end{bmatrix} \right).$$

In so doing, one may leverage that $Y(\mathcal{X})$ values are more highly correlated with Y_N values whose X_N entries are close to $\mathcal{X}$ ones by applying standard MVN conditioning rules, such as those found in Kalpić and Hlupić [52], $Y(\mathcal{X}) \mid Y_N \sim \mathcal{N}_{N'}(\mu_N(\mathcal{X}), \Sigma_N(\mathcal{X}))$ where

$$\begin{aligned} \mu_N(\mathcal{X}) &= \Sigma(\mathcal{X}, X_N)\Sigma_N^{-1} Y_N \\ \Sigma_N(\mathcal{X}) &= \Sigma(\mathcal{X}, \mathcal{X}) - \Sigma(\mathcal{X}, X_N)\Sigma_N^{-1}\Sigma(X_N, \mathcal{X}). \end{aligned} \tag{2.2}$$

The diagonal of the $N' \times N'$ matrix $\Sigma_N(\mathcal{X})$ provides pointwise predictive variances which may be denoted as $\sigma_N^2(\mathcal{X})$. Note that $\Sigma_N(\mathcal{X})$ and $\Sigma(\mathcal{X}, \mathcal{X})$ are $N' \times N'$ matrices; the N subscript serves as a reminder of conditioning on Y_N. Observe that $\mu_N(\mathcal{X})$ is a (high dimensional) linear projection of those Y_N values, where the "weights" involved are inversely proportional to the distance between their X_N values and those of $\mathcal{X}$. Such conditioning identities apply for any MVN, based on a GP prior or otherwise. The special thing about the regression context is the (inverse) distance-based dynamics manifest as $\mathcal{O}(N)$ weights in each row of $\Sigma(\mathcal{X}, X_N)$, and $\mathcal{O}(N^2)$ in Σ_N, involved in the projection, rather than the usual $\mathcal{O}(d)$ or $\mathcal{O}(d^2)$ weights in, say, an ordinary linear regression. That higher-dimensional linear projection, $\mu_N(\mathcal{X})$, has properties that transcend the Bayesian interpretation. For example, under certain conditions it is a best linear unbiased predictor (BLUP). Although much of modern statistical and machine learning literature now understands Eq. (2.2) in a wider, primarily Bayesian GP context, they are identical to the so-called *kriging equations* [65], which have been instrumental in geospatial and mining analysis of point-referenced

measurements for more than half a century.

To illustrate use of Eq. (2.2) in practice, consider $f(x) = 2 + 2\sin(4\pi x)$, observed at $N = 20$ x_i-values distributed uniformly in $[0, 1]$ as $y_i = f(x_i) + \varepsilon_i$, where $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, 0.1)$. While GP regression is usually applied in higher dimension, such as 2d and beyond, the 1d setting is convenient for visualization. These twenty (x_i, y_i) pairs comprise the "training data" (Y_N, X_N). Now, suppose one had a dense testing grid of N' predictive locations $\mathcal{X}$ covering $[0, 1]$. Applying Eq. (2.2) would produce an N' vector of predictive means $\mu_N(\mathcal{X})$ and an $N' \times N'$ matrix of predictive covariances $\Sigma_N(\mathcal{X})$ summarizing the regression of y onto x. Variances $\sigma_N^2(\mathcal{X})$ along the diagonal of $\Sigma_N(\mathcal{X})$ could be used to build error-bars describing a predictive interval (PI) as roughly $\mu_N \pm 2\sigma_N$ for 95% coverage.

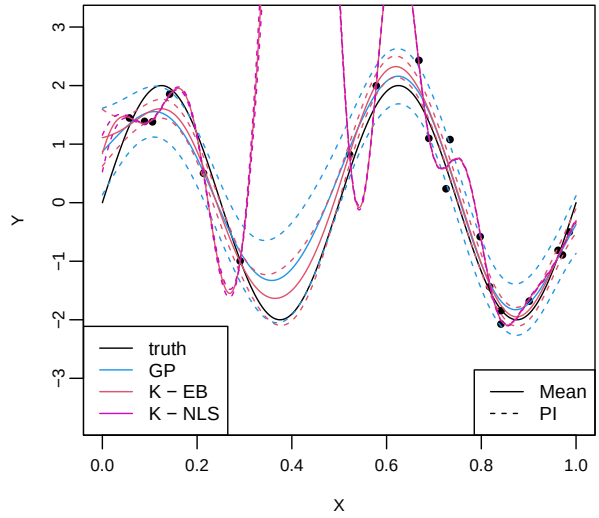


Figure 2.1: A 1d example function in black with dots being the observed locations. The GP prediction is in blue with dashed lines being the 90% predictive interval (PI). Kriging predictions are shown in red and magenta with EB and NLS referring to different variography techniques.

These quantities are shown for one example of such data in Figure 2.1. Noisy data evaluations (solid dots) dance around the true unknown function f (black line); the prediction(s) μ_N and PIs in (blue/red lines, dashed respectively) in two variations (labeled "GP" and "Kriging")

accurately distill the essence of the input-output relationship. Of course, all of this is modulo a fortuitous choice for $\Sigma(\cdot,\cdot)$ which I have yet to detail. Its specification, inference for unknown quantities and interpretation, comprise of the most important difference between modern approaches using GP regression in statistics and machine learning, and what is known historically as kriging. The following discussion is designed to shed light on the differences between red and blue in the figure.

2.2 Modeling

The discussion above hinges on a choice of $\Sigma(\cdot,\cdot)$, or S in Eq. (2.1). S was merely used as a notational device to delay discussion until this moment; I shall not use S going forward, however I will relate $\Sigma(\cdot,\cdot)$ to a function of inverse distances with similar, canonical machine learning notation. That Eq. (2.1) formulation is attractive because it abstracts all modeling details down to this "one" choice. "One" is in quotes because $\mathcal{O}(N^2)$ quantities, one for each pair of N data elements (x_i, y_i), are actually constrained by the covariance structure. This vast number of potentially tunable quantities, more even than N, is why one refers to GPs as nonparametric. But of course, it's neither practical nor valid to allow oneself such unbridled freedoms. For example, one must choose $\Sigma(\cdot,\cdot)$ so that Σ_N is finite and positive definite for use as an MVN covariance.

An inverse-distance-based covariance is conventional as an intuitive spatial modeling device. However, this is not a requirement and may not be ideal in all situations, e.g., when modeling periodic effects. One may wish to allow flexibility in how distances are measured, in what coordinates and with what decay in inversion, and to control how such choices relate signal to noise. Such considerations lead to frameworks for choices of $\Sigma(\cdot,\cdot)$ whose tunable quantities, sometimes called *hyperparameters* to acknowledge a nonparametric modeling apparatus, can

be leaned from data.

For example, if the range of the responses Y_N is unknown *a priori* one might wish to design $\Sigma(\cdot,\cdot)$ to include a scale hyperparameter, say τ^2. If Y_N is noisy and/or contains measurement error, one may wish to encode it as part signal and part noise. Sometimes this is governed by a so-called *nugget* hyperparameter, which I shall denote as g. I caution that the role of GP nugget is inspired by, but is subtly different from, a parameter of the same name in the geostatistics/kriging literature. More in Section 2.4. One may wish to control the smoothness and rate of decay of correlation of the signal in terms of (inverse) distance, and thereby the smoothness and other properties of the underlying response surface. Control of these phenomena may be accomplished through selection of a so-called *kernel* function $k_\theta(x_i, x_j) : \mathbb{R}^d \to [0, 1]$ whose hyperparameter θ can be used to describe the rate of radial decay from $k_\theta(x_i, x_j) = 1$ for $x_i = x_j$ down to zero as x_j moves away from x_i in an isotropic modeling context. Kernels k may also re-scale and/or rotate the space for anisotropic modeling in a way that preserves positive definiteness.

One way to put these elements together is

$$\Sigma(x_i, x_j) = \tau^2(k_\theta(x_i, x_j) + g\delta_{ij}) \quad \text{so that} \quad \Sigma_N = \tau^2(K_N + g\mathbb{I}_N). \tag{2.3}$$

Above, δ_{ij} is the Kronecker delta function returning 1 when the index $i = j$, i.e., when the same training data element appears in both arguments, and zero otherwise, and $K_N \equiv k_\theta(X_N, X_N)$ applying k elementwise as $K_N^{ij} = k_\theta(x_i, x_j)$. Observe that the diagonal of Σ_N is $\tau^2(1 + g)$ and all off-diagonal entries are less than or equal to τ^2, and strictly less than for all $x_i \neq x_j$. This discontinuity between diagonal and off-diagonal, as long as $g > 0$, leads to smoothing of the predictive surface when following Eq. (2.2). Otherwise, when $g = 0$ the surface interpolates. Again, this is a little different than the typical geostatistics formulation

as explained later in Section 2.4.

Choices for distance-based kernels k preserving positive definiteness and targeting certain other properties abound. See, e.g., [1] or [102]. The two that are in most common use in statistics and machine learning are the power exponential and the Matérn. These are provided below in an isotropic setting, i.e., with radial decay as a function of distance.

$$
\begin{aligned}
k_\theta^P(x_i, x_j) &= \exp\left\{-\frac{||x_i - x_j||^p}{\theta}\right\} \\
k_\theta^M(x_i, x_j) &= \frac{2^{1-\nu}}{\Gamma(\nu)}\left(||x_i - x_j||\sqrt{\frac{2\nu}{\theta}}\right)^\nu \mathcal{K}_\nu\left(||x_i - x_j||\sqrt{\frac{2\nu}{\theta}}\right)
\end{aligned}
\tag{2.4}
$$

In both cases above the hyperparameter θ appears in the denominator, scaling (squared or square root) Euclidean distances between x_i and x_j, and is thus sometimes called the *characteristic lengthscale*. Both have additional hyperparmeters, p in the case of the power exponential and ν for the Matérn, which must be positive and which my notation does not include in θ. These control the smoothness of the kernel, and thus the smoothness of the resulting response surface. When $p = 2$ the power exponential produces infinitely smooth (mean-square differentiable) realizations, and sometimes this special case is called the Gaussian kernel,[1] which I denote as k_θ^G. When $p \neq 2$, the surface is nowhere differentiable. While sometimes such pathological non-smoothness is a reasonable assumption – and in spite of this the predictive surfaces (2.2) often look smooth – there are better mechanisms for relaxing unreasonable (infinite) smoothness.

Many of the choices in my references above offer higher fidelity control over smoothness (beyond none and infinite). Of those, the Matérn has percolated into the canonical position thanks largely to persuasive technical arguments from Stein [90]. The parameter ν controls

[1] It is worth noting that the name *Gaussian* here refers to the expression resembling the density of a Gaussian distribution. But it has nothing to do with making a Gaussian assumption, or its use in GP. Such kernels are used in a variety of other contexts.

this aspect, with higher values leading to greater smoothness, yielding surfaces which are $\lceil \nu \rceil - 1$ mean-square differentiable. Ultimately when $\nu \to \infty$ the Gaussian kernel is recovered as a special case. However, the modified Bessel function $\mathcal{K}_\nu$ can be difficult to work with computationally. Specific settings with $\nu \in \left\{\frac{3}{2}, \frac{5}{2}\right\}$ have algebraic closed forms (no Bessel functions) which yield degree one and two differentiability, with the latter being most often applied in practice because most interesting dynamics result from at least twice (but not infinitely) differentiable processes.

Both the (isotropic) power exponential/Gaussian and Matérn are *stationary* kernels because they are defined only in terms of displacement $x_i - x_j$, so the resulting response surface would have identical dynamics throughout the entire input space. Nonstationary modeling is also possible, but a lot more difficult in general. See, e.g., [81], and further discussion in Section 3.2.1. However, one can still capture nontrivial dynamics with stationary kernels, for example by deploying several of them simultaneously: sums, products, convolutions (and more) of valid kernels for GP regression (i.e., are positive definite) are also valid. For details, see e.g., [77, , Section 4.2.4] or [38, , Section 5.3.3]. One of the most common applications of this result is to extend to axis-aligned anisotropy by taking a product of kernels applied univariately in each coordinate direction: $k_\theta(x_i, x_j) = \prod_{k=1}^d k_{\theta_k}(x_{ik}, x_{jk})$, abusing the notation somewhat. This can be done with any kernel. Notice here I am introducing a d-dimensional lengthscale parameter $\theta = (\theta_1, \ldots, \theta_d)$, controlling the rate of spatial correlation differentially in each coordinate direction. For the Gaussian kernel, the result is identical to

$$k_\theta^G(x_i, x_j) = \exp\left\{-\sum_{k=1}^d \frac{(x_{ik} - x_{jk})^2}{\theta_k}\right\}, \tag{2.5}$$

which is sometimes called the *separable* Gaussian kernel, or in machine learning the ARD Gaussian kernel. ARD stands for *automatic relevance determination* [60], borrowing terminology from early neural networks literature. The idea is that the data can inform on longer

lengthscales (less relevant) or shorter ones (more relevant) for each input variable separately. ARD/separable Matérn kernels are also common, but their expression(s) are less tidy so I do not include it here. For more discussion, consult Rasmussen and Williams [77] or Gramacy [38].

There is a one-to-one relationship between vectorized lengthscale in the ARD kernel formulation and with re-scaling inputs X_N, say as a pre-processing step. Rather than scaling each input differentially, one can extend this idea to rotations and projections to accommodate less rigid anisotropy either as preprocessing or as a hyperparameterized kernel formulation. For an example of a pre-processing approach see Wycoff et al. [105] and references therein; for learning rotations and projects *within* a kernel see Gramacy and Lian [35]. Although these represent interesting, modern, high-powered approaches to GP modeling, I find that they are overkill for the mining applications that motivated this work. I illustrate how separable/ARD modeling is sufficient to improve upon the state-of-the-art in modeling borehole-extracted ore deposits.

2.3 Inference

The models put forth above have tunable quantities, or hyperparameters, that could be set by hand but would ideally be learned from data. I restrict the focus to those introduced for $\Sigma(\cdot, \cdot)$ (2.3), particularly $\phi \equiv (\tau^2, g, \theta)$ with the latter usually vectorized in an ARD setting, but there could potentially be others or additional quantities which must be estimated from data. There are many criteria and algorithms across several literatures devoted to such "fitting" enterprises. Yet there is a remarkable confluence in modern statistical and machine learning practice when it comes to the near universality of likelihood-based methods when distributional assumptions are being made (like the MVN in Eq. (2.1)). The reason is that

no additional criteria need be introduced to commence with learning. One may choose to impose additional assumptions, like priors on aspects of ϕ for a Bayesian approach (which is still likelihood-based), or not – simply maximize the likelihood. This is the approach I shall advocate here.

Eq. (2.1) depicts how observations/outputs (like Y_N), are distributed in relation to inputs (like X_N) and parameters or other structure (like $\Sigma(\cdot,\cdot)$ via hyperhaprameters ϕ and kernels k). The *likelihood* simply re-frames the density of that distribution, which assigns positive real values to Y_N as a function of parameters (or hyperparameters ϕ, say), the other way around: providing positive reals for ϕ given Y_N. Once in that context, it makes sense to seek out the parameterization that makes the observed Y_N most likely, i.e., that maximizes the likelihood. There are two benefits to this approach. One is that it reduces a statistical inference question to an optimization one without introducing auxiliary criteria. The other is that the solution to this optimization, the so-called maximum likelihood estimator (MLE), has special properties that can be used to quantify uncertainty. For a review of likelihood-based inference, see Casella and Berger [17].

By inspecting the MVN density [52], using a mean of zero and covariance Σ_N, one may obtain the following expressions for the likelihood and its logarithm.

$$L(\phi; Y_N) = (2\pi)^{-\frac{N}{2}} |\Sigma_N|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} Y_N^\top \Sigma_N^{-1} Y_N\right\} \tag{2.6}$$
$$\ell(\phi; Y_N) = \log(L(\phi; Y_N)) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_N| - \frac{1}{2} Y_N^\top \Sigma_N^{-1} Y_N$$

Recall from Eq. (2.3) that the parameters ϕ are embedded in $\Sigma_N = \tau^2(K_N + g\mathbb{I}_N)$ via K_N which is built through pairwise application of $k_\theta(\cdot,\cdot)$. Technically, this is a *marginal likelihood* since it tacitly defines both prior (which is integratedout) and observational variability. But this distinction is not important for my discussion. The curious reader is referred to Section

5.3.2 of Gramacy [38]. The log likelihood helps with maximization, easing differentiation but preserving critical points.

Consider τ^2 first. Observe that

$$\ell(\phi; Y_N) = c_1 - \frac{N}{2}\log\tau^2 + \frac{N}{2\tau^2} Y_N^\top \Sigma_N^{-1} Y_N$$

where $c_1 = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log|K_N + g\mathbb{I}_N|$ is constant with respect to τ^2. Differentiating with respect to τ^2, setting to zero and solving, is straightforward.

$$0 \stackrel{\text{set}}{=} \frac{d\ell(\phi; Y_N)}{d\tau^2} = -\frac{N}{2\tau^2} + \frac{1}{2(\tau^2)^2} Y_N^\top (K_N + g\mathbb{I}_N)^{-1} Y_N^\top \quad \rightarrow \quad \hat{\tau}^2 = \frac{Y_N^\top (K_N + g\mathbb{I}_N)^{-1} Y_N^\top}{N} \tag{2.7}$$

If I then plug $\hat{\tau}^2$ back into $\ell(\phi; Y_N)$ I get what is known as a concentrated, or profile log likelihood:

$$\ell(\theta, g; Y_N) = \ell(\phi; Y_N)\big|_{\tau^2 = \hat{\tau}^2} = c_2 - \frac{N}{2}\log Y_N (K_N + g\mathbb{I}_N)^{-1} - \frac{1}{2}\log|K_N + g\mathbb{I}_N|$$

where c_2 is not a function of θ or g. Differentiation here is more challenging because the parameters are buried within matrix inverses and determinants. But it is still doable. See, e.g., Eq. (5.9) in Gramacy [38]. I have

$$\begin{aligned} \frac{d\ell(\theta, g; Y_N)}{dg} &= \frac{N}{2}\frac{Y_N^\top ((K_N + g\mathbb{I}_N)^{-1})^2 Y_N}{Y_N^\top (K_N + g\mathbb{I}_N)^{-1} Y_N} - \frac{1}{2}\mathrm{tr}((K_N + g\mathbb{I}_N)^{-1}) \\ \frac{d\ell(\theta, g; Y_N)}{d\theta} &= \frac{N}{2}\frac{Y_N^\top K_N^{-1} K_N' (K_N + g\mathbb{I}_N)^{-1} Y_N}{Y_N^\top (K_N + g\mathbb{I}_N)^{-1} Y_N} - \frac{1}{2}\mathrm{tr}((K_N + g\mathbb{I}_N)^{-1} K_N'), \end{aligned} \tag{2.8}$$

where $K_N' = \frac{dK_N}{d\theta}$. The former is actually a special case of the latter, taking instead θ as g, since $\frac{d(K_N + g\mathbb{I}_N)}{dg} = \mathbb{I}_N$. When θ are lengthscales in k_θ, K_N' is formed of $(K_N')^{ij} = dk_\theta(x_i, x_j)/d\theta$, i.e., with component-wise derivatives of the kernel, say Gaussian or Matérn,

with respect to θ. When θ is vectorized, one may instead calculate $\partial \ell(\theta, g; Y_N)/\partial \theta_k$, forming a gradient for $k = 1, \ldots, d$.

The next step is to set these derivatives to zero, either simultaneously or separately, and solve. However, no algebraic solution is known. Library-based numerical optimization, such as the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) [16] provided in R's `optim` function, can be used to find MLEs $\hat{g}$ and $\hat{\theta}$. Convergence is usually both robust and fast, even in high dimension d, when gradients (2.8) are provided since the underlying $\ell(\theta, g; Y_N)$ surface is almost always convex.

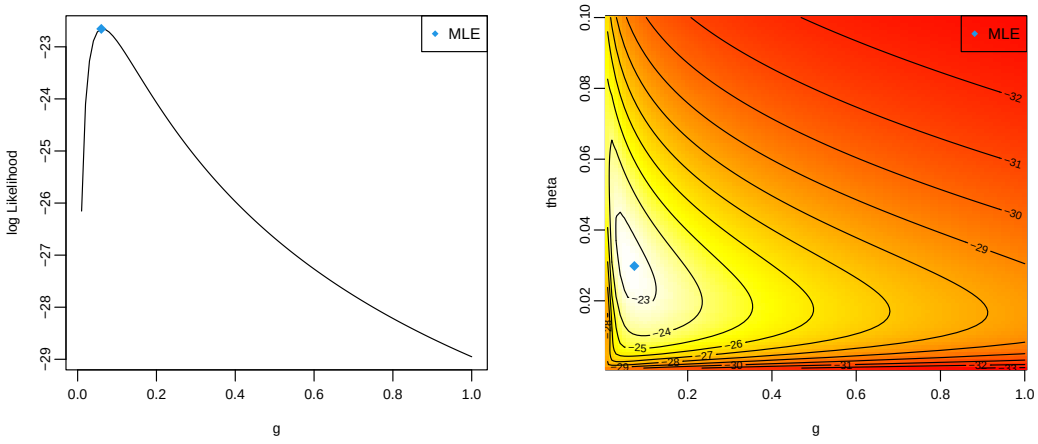


Figure 2.2: Log likelihood surfaces for the function in Figure 2.1. Left: Surface for g, holding τ^2 and θ constant at their MLE values. Right: Surface for g and θ, holding τ^2 at its MLE value.

To illustrate, Figure 2.2 shows $\ell(\theta, g; Y_N)$ using data associated with Figure 2.1. The left panel varies g conditional on a setting of θ, whereas the right panel varies both. In both cases, $\hat{\tau}^2$ has been concentrated out. Note that raw (un-logged) likelihood surfaces are even more peaked than these figures present. Regardless, it is easy to eyeball the MLE. R's `optim` is able to find $\hat{g}$ and $\hat{\theta}$, shown as blue diamonds in both panels in seventy-nine iterations. Looking back at Figure 2.1, the blue curves correspond to predictive surfaces uses these MLE

hyperparamters. The red ones are subject of the following discussion.

2.4 Kriging and Variography

The main difference between classical kriging and the GP presentation above regards inference for unknown quantities and, in the case of the latter, a more up-front and highly-leveraged distributional assumption (Gaussian) for the response. Both use Eq. (2.2) to form predictions and quantify uncertainty. In mining geostatistics, these are known as the "kriging equations," even when other aspects historically associated with kriging are not faithfully replicated. Classical kriging focuses on lower input dimension – particularly $d \in \{2, 3\}$ in spatial contexts – and as such prefers isotropic modeling after a suitable transformation of spatial inputs. *Variography* is used to select the kernel and its hyperparameters, rather than the likelihood. This has advantages and disadvantages. Many of the advantages are related to the historically larger training data sets encountered in spatial problems, although that gap is narrowing in machine learning and computer experiments. More on this in Section 3.2, wherein further distinctions arise. The main disadvantage is that input pre-processing and variogram inspection are inherently hands-on, human driven enterprises, albeit ones enhanced by computational tools. Other differences are more superficial, like naming, symbol choice and applications of hyperparameters within variography.

The *semivariogram*, or half the *variogram*, denoted as $\gamma(h)$, is the variance of two output y-values that are distance h apart in the input x-space:

$$\gamma(h) = \frac{1}{2}\mathbb{V}\mathrm{ar}(Y(x+h) - Y(x)). \tag{2.9}$$

Implicit in this definition is an assumption of intrinsic stationarity, implying that $\mathbb{E}[Y(x+$

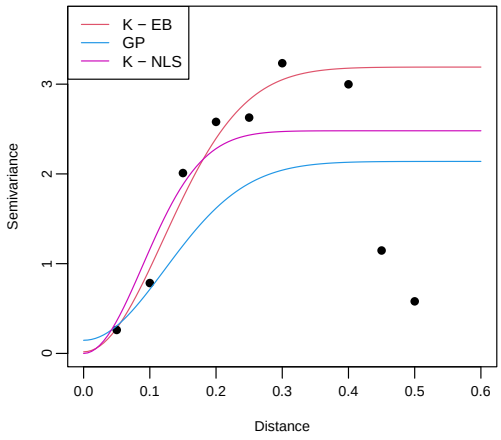
$h) - Y(x)] = 0$, or that the covariance between two y values depends not on position but on relative distance notated by the displacement h between them. If h is calculated using Euclidean distance, intrinsic stationarity implies isotropy. When this is a limitation to effective spatial modeling, one may prescale or rotate the coordinate system. This is often based on expert-judgment of the prevailing variabilities within the input domain, like the direction of an ore body within the geologic topology. As mentioned earlier, a modern GP approach would deploy separable lengthscales (2.5), or more flexibly parameterized rotations and scales that are learned jointly with other unknowns.

The semivariogram is a theoretical/population construct that would be hard to specify *a priori* even with expert knowledge, but simple to observe empirically given data. One estimate of an *empirical semivariogram* could be obtained by binning the data by distance and calculating sample covariances within those bins. Let $N(h_k) = \{(x_i, x_j) : ||x_i - x_j|| \in I_k\}$ where $I_1 = [0, h_1], I_2 = (h_1, h_2], \ldots, I_k = (h_{k-1}, h_k]$ denote a neighborhood structure striated by bands of distance $0, h_1, \ldots, h_k$. Then estimate

$$\hat{\gamma}(h) = \frac{1}{2|N(h)|} \sum_{(x_i,x_j)\in N(h)} (y_i - y_j)^2. \tag{2.10}$$

As defined continuously for any h, $\hat{\gamma}(h)$ is a step function. However it is customarily visualized discretely as a scatter plot with $(h_i + h_{i+1})/2$ as the x-axis coordinate. The left panel of Figure 2.3 shows these as dots for the 1d problem introduced in Figure 2.1 using a bin size $(h_{i+1} - h_i)$ of 0.05.

One can then match these empirical observations of spatial covariance with a parameterized form for the population semivariogram. Here, similar constructs are used to model spatial dependence as the kernels introduced earlier (2.4). Let $\gamma(0) = 0$ and for $h > 0$, power



	GP	K-EB	K-NLS
τ^2 or σ^2	1.99	3.17	2.48
$\tau^2 g$ or τ_k^2	0.145	0.018	0.00010
θ or R	0.17^2	0.17	0.13

Figure 2.3: A Gaussian kernel/semivariogram fit to the function in Figure 2.1. "EB" denotes a semivariogram fit by hand, whereas "NLS" uses non-linear least squares; "GP" derives the semivariogram from an MLE hyperparameterization. The table compares hyperparameter estimates.

exponential and Matérn model semivariograms are often written as

$$\begin{aligned}\gamma_\theta^P(h) &= \tau_k^2 + \sigma^2 \exp\left\{-\left(\frac{h}{R}\right)^p\right\} \\ \gamma_\theta^M(h) &= \tau_k^2 + \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(h\sqrt{\frac{2\nu}{R}}\right)^\nu \mathcal{K}_\nu\left(h\sqrt{\frac{2\nu}{R}}\right).\end{aligned} \tag{2.11}$$

Semivariogram parameters are known as *nugget* (τ_k^2),[2] *partial sill* (σ^2), and *range* (R).

Taking $\gamma(0) = 0$ is a contentious choice outside of the geospatial modeling literature. It implies that there is no intrinsic variance in measurements. In part this is because such measurements are inherently unrepeatable in certain contexts; you cannot dig a borehole in the same place twice. But if you could, it stands to reason that you would get different measurements for such *replicates* for all sorts of reasons, e.g., even without operator error the drill bit might interact with the surface and ore body differently the second time. Primordial process producing the ore body are subject to uncertainties that are best characterized as

[2]A subscript k is not standard; I added it to distinguish with the GP scale τ^2.

random variables even if the process is not inherently stochastic. Thus, it is acknowledged that there will be small-scale variability between *nearby* observations that are best described by noise. This noise, at all distances $h = \epsilon > 0$, is what is parameterized by the nugget τ_k^2. The distinction with the GP nugget g, which characterizes the noise as $\tau^2 g$ at $h = 0$, is thus subtle. Choosing $\gamma(0) = 0$ has an impact on the kriging equations (2.2), leading to discontinuities at the training data locations, where an otherwise smooth predictive surface would be pocked with spikes of "interpolation." I do not show these in the red curve by deliberately omitting X_N values from the predictive grid $\mathcal{X}$ for Figure 2.1 for aesthetic reasons.

Kriging versus GP distinctions between the other two kernel parameters are more superficial. The partial sill σ^2 controls the maximum covariance as $h \to \infty$. This has a 1:1 correspondence with the τ^2 hyperparameter from earlier. Sometimes the *sill* parameter, $\tau_k^2 + \sigma^2$, is preferred by geostatisticians instead. The range R controls the distance between maximum and minimum covariance, and plays an identical role as the square root lengthscale: $R = \sqrt{\theta}$. It is not uncommon to instead specify a decay parameter $\phi = 1/R$, and such inversions are common in the GP literature as well.

Each setting of these parameters could be used to overlay a curve onto the left panel of Figure 2.3. For example, using the MLE hyparameters from the earlier GP analysis yields the curve in blue. Alternatively, one could automate a search for the "best fitting" variogram parameterization with a generalized/nonlinear (possibly weighted) least-squares (NLS) criterion [22]. This corresponds to the magenta curve. Observe that neither of these result in a terrific fit to the semivariogram "data." An outlying pair of dots near $h = 0.5$ drags these variograms down, sacrificing fit for smaller pairwise distances. One reason these are outlying may be that there are many fewer long-distance pairs in the data than short distance ones. A common remedy would be to downweight or altogether ignore these when fitting

the variogram parameters, focusing only on the short distance readings. I refer to one such fit as the "eyeball" (or EB) variogram in the figure, although in practice NLS may similarly be deployed. This can lead to more accurate predictions out-of-sample, as I demonstrate in Chapter 3. However it has the downside of introducing non-statistical (e.g., NLS) and non-metric (determination of outlying semivariogram estimates) criteria which diminishes reproducibility and automation, and incurs the expense of human expert intervention. This enterprise is also sensitive to other choices such as bin size $h_{i+1} - h_i$ and a choice of maximum distance to calculate the empirical variogram. I chose $h_{\max} = 0.5$ for Figure 2.3, but could have gone out to $h_{\max} = 1$, producing a much "noisier" empirical semivariogram.

Figure 2.3 details hyperparameter estimates for each of the three techniques. Lengthscale and range settings exhibit high agreement. For scale/partial sill, NLS and GP MLE values are "drawn down" by the noisier higher distance bins relative to the EB alternative which ignored those values. The nugget is where things start to substantially diverge: $\tau^2 g \gg \tau_k^2$ means the GP-MLE estimates more noise/less signal than the kriging alternatives (EW and NLS). Notice that EB and NLS nuggets are on different orders of magnitude. The tiny NLS τ_k^2 may be attributed to a lack of small distance pairs, and consequently the optimizer converged at the boundary of the search space for that parameter: 10^{-4}, meaning very high signal/low noise. Although this seems innocuous when it comes to the corresponding semivariograms on the left in the figure, the implications out-of-sample are severe. This is foreshadowed in Figure 2.1, but I will delay further discussion of the comparisons between GPs and kriging until Chapter 3 as it starts moving away from foundational review.

2.5 Bayesian optimization

Globally optimizing a black-box function f, finding

$$x^* = \underset{x \in [0,1]^d}{\operatorname{argmin}} \; f(x) \tag{2.12}$$

is a common problem in recommender systems [95], hyperparameter tuning of deep learners [89], and inventory management [45]. In Chapter 4, I plan to target a robot pushing simulator problem [51]. In such settings, f is an expensive to calculate computer simulation, so one needs to carefully design an experiment to effectively learn the function [80] and its properties, like local or global minima. Optimization via modeling and design has a rich history in statistics [for nice review see 14, 70]. Its modern instantiation is known as Bayesian optimization [BO; 68].

In BO, one fits a flexible, non-linear and nonparametric response-surface model to a limited campaign of example runs, obtaining a so-called *surrogate* $\hat{f}$ [38]. Based on that fit – and in particular its predictive equations describing uncertainty for $f(x)$ at new locations x – one then devises a criteria, a so-called *acquisition function* [86], targeting desirable qualities, such as $x^\star$ minimizing f. One must choose an appropriate surrogate family for f, paired with a fitting scheme for $\hat{f}$, choose an appropriate acquisition criteria, and method for solving for that acquisition. It is also important that $\hat{f}$ be quickly updated with new acquisitions, to make the most of each new run without excessive overhead in modeling/solving. BO is an example of *active learning* (AL) where one attempts to sustain a virtuous cycle of learning and data collection. Many good solutions exist in this context, and I shall not provide an in-depth review here.

Perhaps the most common surrogate for BO, and generally for the design and analysis of computer simulation experiments, is the Gaussian Process as described in Section 2.1.

Perhaps the most popular acquisition function is *expected improvement* (EI) which aims to select the point which has the greatest potential to improve on the current best minimum [48]. EI suggests locations with either high variance or low mean, or both; balancing exploration and exploitation. Greater detail is provided in Section 4.1.2. EI is highly effective and has nice theoretical properties.

The idea is to proceed sequentially, $n = n_0, \ldots, N$, and in each iteration n make a *greedy* selection of the next, $n + 1^{\text{st}}$ run, x_{n+1}, based on solving an acquisition function tied to a surrogate fit to data D_n. The initial n_0-sized design could be space-filling, like with a Latin hypercube sample (LHS) or maximin design [28, Chapter 17] or, as some have argued [106], purely at random. GPs have emerged as the canonical surrogate model. Although there are many acquisition functions in the literature tailored to BO via GPs, expected improvement (EI) is perhaps the most popular.

EI may be described as follows. Let $f_n^{\min} = \min(y_1, \ldots, y_n)$ denote the best solution found so far, after the first n acquisitions (n_0 of which are space-filling/random). Sometimes $f_n^{\min}$ is called the "best observed value" (BOV), a quantity often tracked in empirical benchmarking exercises evaluating global optimizers. Then, define the *improvement* at input location x as $I(x) = \max\{0, f_n^{\min} - Y(x)\}$. $I(x)$ is a random variable, inheriting its distribution from $Y(x) \mid D_n$. If $Y(x)$ is Gaussian, as it is under $\text{GP}_{\hat{\theta}}(D_n)$ via Eq. (2.2), then the expectation of $I(x)$ over the distribution of $Y(x)$ given D_n has a closed form:

$$\text{EI}_n(x) = \mathbb{E}\{I(x)|D_n\} = (f_n^{\min} - \mu_n(x))\Phi\left(\frac{f_n^{\min} - \mu_n(x)}{\sigma_n(x)}\right) + \sigma_n(x)\phi\left(\frac{f_n^{\min} - \mu_n(x)}{\sigma_n(x)}\right), \quad (2.13)$$

where Φ and ϕ are the standard Gaussian CDF and PDF, respectively. Notice how EI is a combination of mean $\mu_n(x)$ and uncertainty $\sigma_n(x)$, or colloquially as a linear combination of "exploitation and exploration". The first term of the sum is high when the GP thinks $f(x)$

is much lower than $f_{\min}$, while the second term is high when the GP has high uncertainty at x. Thus by maximizing EI, one gets an organic balance of exploration of new regions with high variance and exploitation of regions already exhibiting low mean.

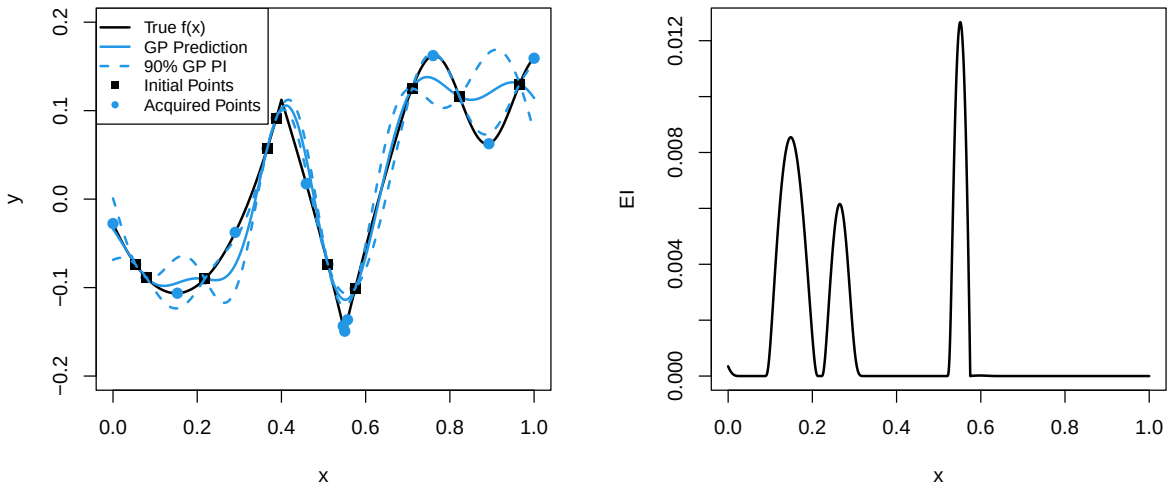


Figure 2.4: Left: EI-based (EGO) acquisition with $n_0 = 10$ initial points as black open circles and 10 acquisitions as blue open circles using EGO on the 1D example function from Eq. (4.7) with initial GP fit in blue. Right: EI using the initial design at different locations of x.

The *right* panel of Figure 2.4 shows an EI surface implied by the GP surrogate (blue) shown in the *left* panel. This fit is derived from $n = 10$ training data points (black open circles) sampled from $f(x)$ later defined in Figure 4.7. I will discuss the blue open circles momentarily. Observe how EI is high in the two local troughs of minima, near $x = 0.15$ and $x = 0.55$, but away from the training data. The total volume (area under the curve) of EI is larger around the left, shallower trough, but the maximal EI location is in the right, spiky trough. This (x near 0.55) is where EI recommends choosing the next, $n + 1^{\text{st}}$ acquisition.

Operationalizing that process, i.e., numerically solving for the next acquisition involves its own, "inner" optimization: $x_{n+1} = \text{argmax}_{x \in [0,1]^d} \text{EI}_n(x)$. This can be tricky because EI is multi-modal. Observe that while f only has two local minima, EI as shown for $n = 10$ in Figure 2.4, has three local maxima. So in some sense the inner optimization is harder

than the "outer" one. As n grows, the number of local EI maxima can grow. A numerical optimizer such as BFGS [16] is easily stuck, necessitating a multi-start scheme [15]. Another common approach is to use a discrete space-filling set of candidates, replacing a continuous search for $x \in [0,1]^d$ into a discrete one for $x \in \mathcal{X}_{\text{cand}}$. Hybrid and smart-candidate schemes are also popular [41, 84]. In general, one can afford a comprehensive effort toward solving the "inner" optimization because the objective, derived from $\text{GP}_{\hat{\theta}}(D_n)$, is cheap – especially compared to the black-box $f(\cdot)$. Repeated application of EI toward selecting x_{n+1} is known as the *efficient global optimization* algorithm [EGO, 48].

The blue open circles in Figure 2.4 indicate how nine further acquisitions (ten total) play out, i.e., $10 = n_0, \ldots, n, \ldots, N = 20$ in EGO. Notice that the resulting training data set D_N, combining both black and blue points concentrates acquisitions in the "tip" of the spiky, right trough. The rest of the space, including the shallower left trough, is more uniformly (and sparsely) sampled. My goal in Chapter 4 will be to concentrate more acquisition effort in the shallower trough.

Chapter 3

Modern Kriging for Geologic Data

Accurate characterization of mining site geology is essential to determine the size and economics of a potential resource of critical minerals, as well as the safety and efficacy of excavation, extractive metallurgy, and environmental remediation. During "deposit definition" and "resource estimation" the deposit is usually drilled on a grid, and the distance between holes is determined using simple methods that satisfy international reporting standards. This drilling and the subsequent analysis of the drilled material are expensive, inefficient, and considerable uncertainty remains even when the spacing is deemed sufficient by reporting standards. This geologic uncertainty creates risk for the operation and the humans involved. For example, financial, safety, and environmental repercussions might include smaller ore reserves than originally estimated, unanticipated mineralogical characteristics that impede ore recovery or generate acid mine drainage, or geotechnical instabilities that endanger workers. In order to provide the best characteristics of mining sites, I compare modern spatial interpolation and smoothing algorithms to find which methods are accurate, provide good uncertainty quantification and are fast to run.

Common geoscience applications for spatial smoothing and interpolation include ore-grade

estimation and reservoir characterization/simulation; however, software tools utilized for such applications lag the state-of-the-art (as outlined above) by a decade or more. For example, obtaining fits requires expert human interaction with the software library and intuition in order to entertain alternatives of spanning anisotropies, neighborhood sizes of ordinary kriging in the face of large training data sets, and appropriate semivariogram forms modeling the decay of spatial correlation. Recent advances from the statistics and data analytics communities automate many of these time-consuming tasks, while offering substantial improvements in computational efficiency and run-time and use of contemporary computing architectures such as multi-core workstations and clusters.

In this chapter, I compare two recent R libraries for Gaussian processes regression, against kriging software in current use on commercial platforms using real gold ore data. One library, called `GPvecchia`, uses the scaled Vecchia approximation [56], which is a modern implementation of established methods that approximate likelihoods in high dimensional conditional distributions [97]. Another, called `laGP` [37], is the evolution of a surrogate modeling/machine learning take on ordinary kriging, where neighborhoods are determined dynamically based on the nearby training data's contribution to reduction in variance, rather than simply being distance-based. Both packages are fully automated in the sense that all tunable parameters are inferred via likelihood-based criteria offloaded to robust optimization libraries. No settings are left to user discretion. So not only are they easy to use, but they are also hard to misuse.

In order to make this comparison, I leverage two real data borehole-based mining examples with data records on gold and other minerals, over spatial and depth coordinates, sized in the hundreds of thousands. Figure 3.1 shows a subset of the data described in more detail in Section 3.3 as a 2d projection; note the long stretches of observations all in a line. These data exhibit many typical yet challenging features such as abrupt changes in dynamics, left

censoring of small values due to the sensitivity of the measurement instrument, and large measurement gaps in space. Using these data, I devise a cross-validation-based out-of-sample exercise which is careful to respect the borehole nature of data collection. The outcome of that exercise is evidence that modern GP-based methods are both more accurate, more hands-off, more economical (in terms of computing resources), and offer better uncertainty quantification than their kriging-based analogues. They also enable extensions which would be difficult to entertain without a fully probabilistic generative framework. As a showcase, I entertain a multiple imputation scheme to handle left censoring that involves only a few lines of code around library-based GP fitting and prediction subroutines.

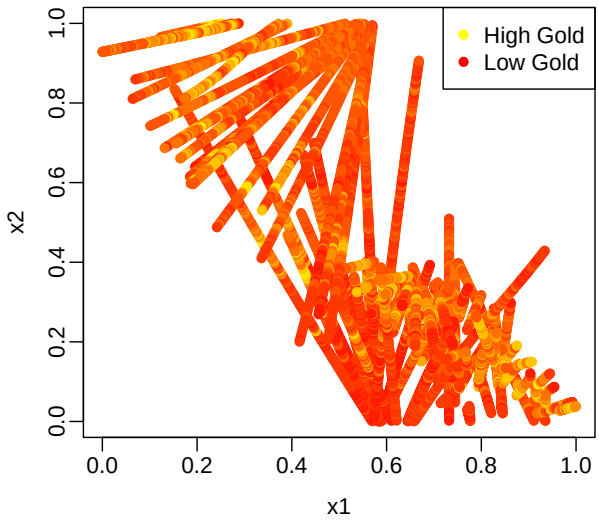


Figure 3.1: An example of borehole data as a 2d projection.

This chapter will continue as such. First, I further explore the differences of GPs and kriging using our toy problems introduced from Chapetr 2. Then I cover modern state-of-the-art approaches for adapting GPs and kriging to big data. Next, I demonstrate how the modern methods compare to each other on two real drill hole data sets. Finally, I finish the chapter off with a discussion about further possible extensions.

3.1 A comparison of GP regression and kriging

In Figure 2.1 the EB kriging fit (solid-red) is visually similar to the GP-MLE fit (solid-blue) except perhaps near $x = 0.4$. This is noteworthy in light of the disparate parameterization and semivariograms in Figure 2.3, and in particular the human intervention required to ignore outlying values in favor of short distances. Qualitatively, the red curve may be more accurate compared to the truth (black), but closer inspection reveals a more pernicious concern despite apparent higher accuracy: poor uncertainty quantification. The red 90% PI (error-bars) cover only about half of the training data locations, suggesting that nominal coverage has not been achieved. In contrast, the blue (GP-MLE) error-bars cover many more of the data points.

There are many ways to be more precise about out-of-sample prediction accuracy. Consider a testing set comprised of a grid of $\mathcal{X}$-values of size N' with true values $y(\mathcal{X})$. Given predictions from a kriging/GP fit (2.2), notated generically as $\mu(\mathcal{X})$, the *proper scoring rule* [33] is root mean square error (RMSE).

$$\text{RMSE}(y(\mathcal{X}), \mu(\mathcal{X})) = \sqrt{\frac{1}{N'} \sum_{i=1}^{N'} (y_i(x_i) - \mu_i(x_i))^2} \tag{3.1}$$

Lower RMSE is better. The first row of the left panel of Table 3.1 shows that the mean predictions for kriging via EB are indeed more accurate than via MLE.

If one also has covariances $\Sigma(\mathcal{X})$, or just variances $\sigma^2(\mathcal{X}) = \text{diag}(\Sigma(\mathcal{X}))$, associated with predictions, the proper scoring rule is more complex as accuracy must be normalized by

1d	GP	K-EB	K-NLS
RMSE	0.14	0.084	23.11
score_f	869	-4161	-1255140

Meuse	GP	K-EB	K-NLS
RMSE	0.103	0.109	0.108
score_f	16.79	15.97	16.50

Table 3.1: Left: RMSE and score for GP and kriging inference for the 1d toy problem on a grid of size 1000 uniformly spanning 0-1. Right: RMSE and score for GP and kriging inference for the Meuse river data on a set of 15 hold out points.

uncertainty:

$$\text{score}_\text{f}(Y(\mathcal{X}), \mu(\mathcal{X}), \Sigma(\mathcal{X})) = -\log(|\Sigma(\mathcal{X})|) - (Y(\mathcal{X}) - \mu(\mathcal{X}))^\top \Sigma(\mathcal{X})^{-1}(Y(\mathcal{X}) - \mu(\mathcal{X}))$$

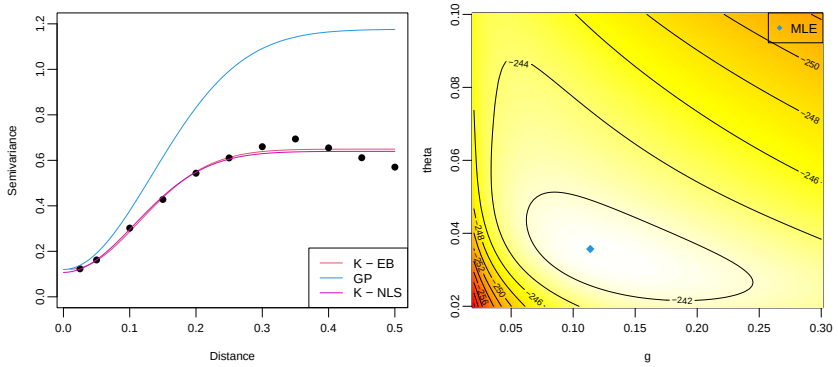
$$\text{score}_\text{p}(Y(\mathcal{X}), \mu(\mathcal{X}), \sigma^2(\mathcal{X})) = -\sum_{i=1}^{N'} \log(\sigma_i^2) - \sum_{i=1}^{N'} \left(\frac{(Y_i(x_i) - \mu_i(x_i))^2}{\sigma_i^2} \right). \tag{3.2}$$

In these equations an upper-case $Y(\mathcal{X})$ is used to convey that a comparison is being made to noisy sample of Y-values following the data-generating mechanism, Observe that score based on full covariance (score_f) is the out-of-sample analog of the log likelihood (2.6), i.e., within an additive and multiplicative constant of criteria deplored in-sample for GP hyperparameter inference. Consequently, this criteria is sometimes called the *predictive log likelihood* [32]. The "pointwise" analog (score_p) offers an approximation for the situation where full covariances cannot be derived, as sometimes happens with big testing sets $\mathcal{X}$. For now, I have full $\Sigma(\mathcal{X})$ for all three predictors so Table 3.1 provides score_f-values. Notice that by score, the GP-MLE hyperparameterization is better, confirming my intuition that the PIs for K-EB are too narrow.

A more realistic example

Consider the Meuse river data, which often serves as a tutorial-level example of kriging [74]. These data consist of measurements of concentrations of metals in the topsoil of the Meuse

river in the Netherlands. My analysis here focuses on 155 zinc measurements in parts per million (ppm) taken at 2d spatial locations recorded as Rijksdriehoeks Coordinates (National Triangulation Coordinates of the Neatherlands) distributed at varying uniformity along the river. To prepare the data for analysis, I follow the machine learning/surrogate modeling practice of coding these inputs as X_N in the unit cube $[0,1]^2$ and model the centered (i.e., empirically mean adjusted) natural logarithm of zinc concentration as the response. This automatic pre-processing step helps initialize the numerical solvers for hyperparameters at default settings, and makes a Gaussian assumption on the response more compatible.



	GP	K-EB	K-NLS
τ^2 or σ^2	1.06	0.53	0.53
$\tau^2 g$ or τ_k^2	0.12	0.12	0.11
θ or R	0.19^2	0.15	0.15

Figure 3.2: Left: Variogram for kriging by eye in red, kriging by NLS in purple and GP in blue. Middle: GP log MLE surface with optimum in blue. Right: Table comparing hyperparameters of GP vs kriging inference.

Figure 3.2 shows the variogram and MLE-based analysis in views similar to Figures 2.2 and 2.3 for these data. Observe that the empirical semivariogram is much better behaved, compared to the first example, and that consequently EB and NLS hyperparameter estimates and variogram fits are quite similar. The main difference between the GP and these kriging alternatives is that the estimated scale hyperparameter τ^2 is much larger. With similar nuggets and lengthscales, the rate of increase in the variogram is about the same, but reaching a higher level. Since the estimated scale does not impact the predictive mean $\mu(\mathcal{X})$, see Eq. (2.2), it is perhaps not surprising that the predictive surfaces for GP-MLE and K-EB are so similar in Figure 3.3. A surface for K-NLS is not shown because of the stark similarity

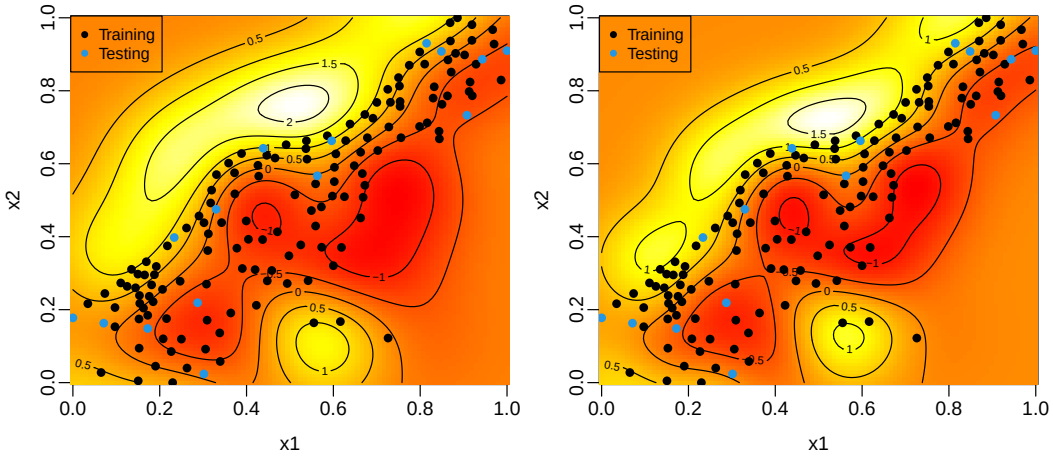


Figure 3.3: GP-MLE (left) and K-EB (right) predictions of log zinc.

with K-EB. These figures used a $\mathcal{X}$ comprised of a 100×100 grid in the coded input space.

Unfortunately, no one knows the real response(s) on this predictive grid, so an out-of-sample analysis must be more limited than the first example. As a pre-cursor to the cross-validation scheme introduced in Section 3.3, I consider a simple random 90:10 train–test partition of the original 155 training data records. This results in $N = 140$ values (X_N, Y_N) that can be used for training the three models, and $N' = 15$ values $(\mathcal{X}, Y(\mathcal{X}))$ quarantined away for testing to see which works best out-of-sample by RMSE and score (again score_f from Eq.(3.1)). Observe that in this case there is no distinction between "true" and random y-values as I only have the testing outputs from the data. In Figure 3.3, the random partition is indicated by coloring the 15 testing dots blue as opposed to black. Table 3.1 records these values. Notice that while GPs edge out both kriging methods, the results are comparable for all three. Ideally I would repeat this enterprise multiple times, with novel 90:10 splits. For the MLE-GP this is easy, but for the kriging alternative it can be labor intensive if eyeballs are involved. I will revisit this exercise with greater rigor in Section 3.3.

An important takeaway from this analysis, and the previous one, is that labor-intensive expert intervention does not always pay off. Although surely it is possible to tweak things

just so to beat the machine, there are also real risk associated with such endeavors. The automated GP approach performs admirably despite leveraging additional assumptions, such as Gaussianity, which are not always a good match to the data generating mechanism. Yet there are other advantages to kriging approaches when dealing with data at modern scale, which is the next topic.

3.2 Large Scale Kriging and Gaussian Processes

Likelihood-based GP inference for hyperparameters (2.6–2.8) and prediction (2.2) requires matrix decomposition for the inverse and determinant of the covariance structure. Most kernels, e.g., Gaussian and Matérn, produce dense K_N (and thus Σ_N), which require $\mathcal{O}(N^3)$ decomposition, say via Cholesky (which furnishes both inverse and determinant). This is prohibitive for N larger than a few thousand. For example, decomposing a single $40,000 \times 40,000$ matrix on a workstation using 8 cores and specialized linear algebra libraries (Intel MKL) takes about 10 minutes. Numerical optimization of hyperparameters might require hundreds of such decompositions in search of the MLE via BFGS. With cubic scaling for larger N, computation time quickly explodes to hours or days. $\mathcal{O}(N^2)$ storage of the $N \times N$ matrix can also become problematic even on the most powerful workstations. Kriging-based inference for hyperparameters via variography bypasses the need to work with an $N \times N$ covariance matrix by binning the data. However, Eq. (2.2) still requires a dense $N \times N$ inverse to furnish predictions, which is still cubic in computational order.

In the modern age, datasets can easily push into the multimillions and the methods described in Chapter 2 quickly become infeasible. For that reason, there are increasingly many approaches that seek a thrifty approximation to GP/kriging models. Heaton et al. [44] give a thorough comparison of about a dozen different recently introduced spatial meth-

ods equipped to handle large data. Here I focus on three representative approaches as a means of spanning myriad alternatives in a mining context: Ordinary kriging [OK; 66] is the standard method in mining/geostats which makes local (approximate) prediction after full-data variogram-based hyperparameter estimation; Local Approximate Gaussian Processes [LAGP; 39] can be seen as a likelihood-based contemporary analog of OK developed in the surrogate modeling community, making it a natural comparator to OK; finally, the scaled Vecchia approximation (SVecchia) [56] uses a global approximation to estimate the full covariance structure based on similar locality principles as LAGP/OK. Details for each of these follow in subsections below.

Lastly as a baseline, I consider subset GPs trained via a randomly selected, computationally feasible, $m \ll N$-sized subset of the data points and use them to form an approximation to the full model. Specifically, I build $(X_m, Y_m) \subset (X_N, Y_N)$, using the likelihood for hyperparameter inference via Eqs.(2.6–2.8) using (X_m, Y_m) and prediction similarly following (2.2). I consider m ranging from 1,000 to 8,000; I show later in Figure 3.7 that $m >$ 8,000 is very slow and not competitive with the other methods in terms of accuracy out-of-sample.

3.2.1 Transductive Modeling

Perhaps the most common solution to big-data matrix issues when predicting via kriging is to deploy what is known as *ordinary kriging* [OK; 66]. OK involves using full-data variography to learn kernel hyperparameters, and then a *local* application of that learned kernel through predictive equations (2.2) conditioned only on a small subset of the data nearby the predictive location(s) of interest. Let $x \in \mathcal{X}$ denote the coordinates of one such location, and $X_m(x) \subseteq X_N$ denote the m "closest" (e.g., via Euclidean distance) members of X_N to x, and let $Y_m(x)$ be the m-associated output values. These are sometimes called the *m-nearest neighbors*

(NN) to x in X_N. Then simply apply Eq. (2.2) with $(X_m(x), Y_m(x))$ rather than (X_N, Y_N). When N is so large that the requisite $N \times N$ matrix decompositions are intractable, choosing $m \ll N$ like $m = 50$ can represent a thrifty-yet-accurate alternative acknowledging that the discarded points $X_N \setminus X_m(x)$ have vanishingly small impact on the predictive equations especially when kernels involve exponential decay.

If a multitude of $x \in \mathcal{X}$ are of interest, these may be processed in serial or, as is increasingly common with modern computing architectures, in parallel on multiple cores of a workstation and/or nodes of a supercomputer. Vast predictive grids $\mathcal{X}$ can be processed efficiently in this manner. There are several variations on this theme, many involving how the "neighborhood" $X_m(x)$ and its size m are defined. For example, one may work with a radius r instead, implicitly defining m depending on the local nature of design locations X_N nearby x. Suitable r from a modeling perspective may be selected by the estimated range R of the semivariogram. However, this does not guarantee a suitably-sized m for all x. One may end up with too small of a neighborhood to make computationally stable calculations/reliable low-variance predictions, or too large of one to be carried out efficiently from a computational perspective. Consequently, there are many hybrids that are often deployed in this space [98, Chapters 11-13].

The idea of tailoring a statistical calculation to a predictive task, using different data and possibly different calculations depending on the predictive location x of interest, is now known as *transductive learning*, a term coined in the machine learning literature [96]. The transductive moniker is meant to contrast with the more typical *inductive learning* setup where one trains first and predicts second. Under transductive learning, the training happens bespoke to each $x \in \mathcal{X}$, and usually on-demand/in real time. Examples span the gamut of statistical modeling enterprises, often offering both speed and accuracy gains over the inductive analog. Reviewing these would be a distraction here. Instead, I note that OK is an example

of transductive learning ahead of its time, albeit a somewhat limited one. Hyperparameter learning with OK is inductive, whereas posterior predictive conditioning is transductive. It is this latter stage where the nonparametric flexibility really comes from, although one might wonder whether things could improve by enhancing the degree transductively, as it were.

A prime example of transductive GP modeling in the machine learning and surrogate modeling literature is the so-called *local approximate Gaussian process* [LAGP; 39]. LAGP is similar to OK, using $X_m(x) \subset X_N$ and $Y_m(x)$ analogously for prediction, but it is different in that it extends the notion of locality to hyperparameter inference via the (local) likelihood. That is, the entire process conditions only on $(X_m(x), Y_m(x))$ for inference (2.6–2.8) and prediction (2.2). Everything is hands-off, offloading inference to numerical optimization. When the response surface is nonstationary, e.g., benefiting by longer lengthscales for some x and shorter for others, LAGP offers enhanced reactivity compared to single, global setting of hyperparameters. Any variation tailored to the full-GP is easy to port to the local setting because LAGP is just many small GPs. Hybrids are possible too. One such example alluded to earlier involves pre-scaling or rotating/projecting [e.g., 93, 105] to handle non-axis-aligned anisotropy. Such tasks are best performed globally first, perhaps on data subsets, before local conditioning on neighborhood data. Extensions abound. However, the biggest difference between LAGP and OK is not about pre-processing or about hyperparameter inference details or potential for hybridization. It is about how the neighborhood is defined.

Given fixed m, usually chosen via computational considerations (a common default is $m = 50$), it has been known for sometime that the m-NNs in X_N to x, whether via Euclidean distance or otherwise, do not comprise of an optimal conditioning set $(X_m(x), Y_m(x))$ under any reasonable criteria [e.g., 91]. Example criteria include (Fisher) information about unknown hyperparameters or, as is usually more relevant when predictive accuracy is concerned, predictive uncertainty (a.k.a., mean-squared prediction error). Note that this, in turn, means

that the OK predictor is also sub-optimal as a transductive learner. However searching for the best conditioning set $(X_m(x), Y_m(x))$, again under almost any criteria, represents a computationally daunting task because there are $\binom{N}{m}$ alternatives to explore.

Here, another machine learning idea comes in handy: *active learning* (AL). AL is a branch of reinforcement learning/optimal control, or maybe viewed as a modernization of statistical sequential design of experiments. In the AL literature, one can often show that a one-step-at-a-time, *greedy* selection of training data is nearly as good as an exhaustive optimization of some criteria by demonstrating that the criteria has a *submodularity* property [101]. For example, it can be shown that repeatedly acquiring training data (x_{n+1}, y_{n+1}) such that x_i maximizes the predictive variance $x_{n+1} = \text{argmax}_x \ \sigma_n^2(x)$ of a GP (2.2) or neural network model training only on $\{(x_i, y_i)\}_{i=1}^n$ obtained previously (e.g., via similar greedy optimization), well-approximates a so-called maximum entropy design, i.e., maximizing Shannon information about unknown hyperparameters (GP lengthscales) for the entire selection $i = 1, \ldots, N$, say. This idea is due to MacKay [62] for neural networks, and dubbed ALM by Seo et al. [85] in extension to GPs.

Intuitively, selecting points which have maximum variance will result in a space-filling design because variance is higher away from the training data. Also intuitively, spreading points out will increase accuracy and reduce uncertainty throughout the input space. But this is coincidental, that maximizing entropy reduces variance. If what you want is reduced predictive variance everywhere, or at a particular location (for choosing an LAGP neighborhood), then it might be better to choose a criteria that squarely targets reduced variance in the region of interest. A common choice is integrated mean-squared predictive error [IMSPE; 80]:

$$\text{IMSPE}(x_{n+1}) = \int_{\mathcal{X}} \sigma_{n+1}^2(x) \, dx \approx \sum_{x \in X_{\text{ref}}} \sigma_{n+1}^2(x) = \text{ALC}(x_{n+1}, X_{\text{ref}}). \tag{3.3}$$

Above, the integral is usually taken over the entire input space, but $\mathcal{X}$ could be any set. This could be interpreted as a criteria for the entire design $X_{n+1} = [X_n; x_{n+1}]^\top$, or simply to select the next input x_{n+1} in an active learning context: $x_{n+1} = \text{argmin}_{x \in \mathcal{X}} \text{IMSPE}(x)$. Due to submodulatirty, both (approximately) optimize the IMSPE critera over all X_{n+1}. The first application of this idea in machine learning was for neural networks [21], and again Seo et al. [85] extended it to GPs. Notice that ALC, as quoted above, approximates the integral as a sum over a discrete reference set X_{ref}. This is not necessary for GPs, because the integral is analytic when following Eq. (2.2), but it is for neural networks and more generally.

For LAGP, the goal is to get as accurate of a prediction at x as possible, which can be interpreted as a singleton $\{x\} = \mathcal{X} = X_{\text{ref}}$, effectively discarding the sum or integral. One can select a new $x_{n+1} = \text{argmin}_{x_{n+1}} \text{ALC}(x_{n+1}, x)$, and repeated applications will approximate a "local" optimal design for predicting at x. This would usually be applied for selecting new training data in an AL context, but for LAGP one already has a fixed training data set (X_N, Y_N) and so one desires a subsample instead: $x_{n+1} = \text{argmin}_{x_{n+1} \in X_N \setminus X_n} \text{ALC}(x_{n+1}, x)$, which is even easier than a continuous search potentially everywhere in the input space.

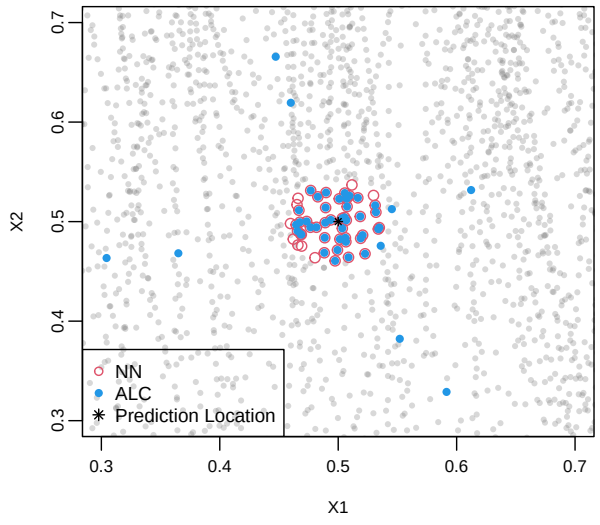


Figure 3.4: Using simulated borehole data, I show neighborhoods of size 50 for NN (analogous to ordinary kriging) and ALC when predicting at (0.5, 0.5).

In practice, early ALC acquisitions (small n) result in neighborhoods that are indistinguishable from NN. However, later acquisitions (larger n), after many NNs have been conditioned upon near x, the local ALC/IMSPE criteria prefers acquiring "satellite" points farther away for a wider perspective. Intuitively, information farther afield becomes more valuable as NNs accumulate near x: you want x_{n+1} to be both close to x but far from X_n. At the start the former dominates, but eventually the latter has higher weight in the criteria. The end result for $n = 50$, seen in Figure 3.4, involves ten or so satellite points, departing from an OK or NN subdesign set of the same size.

To a certain extent, LAGP has a "chicken or the egg" problem when dealing with anisotropy. Neighborhoods selected for x are based on Euclidean distance to determine hyperparamters, like $\hat{\theta}_k(x)$, but those values control notions of distance differently in each input coordinate through the kernel. OK experiences this problem too, albeit to a lesser extent when anisotropy is handled by the practitioner as a pre-processing step. Several remedies have been proposed. The original LAGP paper [39] suggested initializing with a default, isotropic θ_0 for all testing locations x, upon which neighborhoods are built (e.g., via ALC) and local, anisotropic local $\hat{\theta}(x)$ are learned through local MLEs separately for each x. This can then be repeated, with neighborhoods based on those $\hat{\theta}(x)$, until things stabilize. Subsequently, a simpler/better approach was promoted by Sun et al. [93] that is more akin to OK pre-processing, but still completely automated. Liu and Hung [61] showed that unbiased (global) lengthscales $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_d)$ can be estimated via MLEs from carefully constructed data subsets of large X_N without expense cubic in N. Once these have been learned, they can be used to pre-scale inputs X as $X_k/\sqrt{\theta_k}$ so that the implied MLE global lengthscale is $\hat{\theta}_0 = 1$ under squared-distance kernels like in Eq. (2.4). In this transformed space, Euclidean distance can be used to determine neighborhoods. This pre-scaled LAGP has become the default setup, outperforming other methods as it offers a kind of local–global hybrid. The

pre-scaling idea has since been extended to other input "warpings" by Wycoff et al. [105].

Taken as a predictive field over a densely gridded testing set of x-values, both OK and LAGP (NN or ALC) are discontinuous. Prediction at each point $x \in \mathcal{X}$ is processed independently, both in a statistical and computation sense. Therefore, two testing locations right next to each other may have substantively different predictions (in mean and/or variance) because different neighborhoods are used for conditioning. In some cases, this can be a good thing when the response surface exhibits regime/abrupt changes. In tamer, more stationary settings, a non-smooth prediction could be detrimental to statistically efficient and aesthetically pleasing analysis.

3.2.2 The Scaled Vecchia Approximation

The Vecchia GP approximation [97] borrows the neighborhood idea while providing a global model which results in smooth predictions. It relies on a particular probability identity relating joint distributions and their conditionals:

$$\begin{aligned} p(y) = p(y_1)p(y_2|y_1)\dots p(y_n|y_1,y_2,y_3,\dots y_{n-1}) &= \prod_{i=1}^{N} p(y_i|y_{k(i)}) \quad \text{where} \quad k(i) = \{j : j < i\} \\ &\approx \prod_{i=1}^{N} p(y_i|y_{c(i)}) \quad \text{where} \quad c(i) \subset k(i) \end{aligned} \tag{3.4}$$

The first line above (equality) is true for any re-indexing of the variables $y = y_1, \dots, y_n$, and for any y – not specifically for GPs. The approximation (second line) arises from dropping some of those conditioning variables. Let m denote the maximum size of those sets, i.e., so that $|c(i)| = \min(i-1, m)$, controlling the fidelity of the approximation – more severely for $m \ll i$. The quality of this approximation is determined by the indexing (i.e., the ordering

of the conditionals), size m, and which of the conditioning variables $k(i)$ are dropped in $c(i)$ when $m < i$.

Specifically for GPs, one may view the likelihood (2.6), in this context:

$$L(\phi; Y_N) \approx \prod_{i=1}^{N} L(\phi; y_i | y_{c(i)}) \tag{3.5}$$
$$= (2\pi)^{-\frac{N}{2}} \left(\prod_{i=1}^{N} \sigma_i^2 \right)^{-\frac{1}{2}} \exp\left\{ -\sum_{i=1}^{N} \frac{1}{2\sigma_i^2} (y_i - \Sigma(x_i, X_{c(i)}) \Sigma(X_{c(i)}, X_{c(i)})^{-1} y_{c(i)})^2 \right\}$$

where $\Sigma(\cdot, \cdot)$ is defined as in Section 2.1 and $\sigma_i^2 = \Sigma(x_i, x_i) - \Sigma(x_i, X_{c(i)}) \Sigma(X_{c(i)}, X_{c(i)})^{-1} \Sigma(X_{c(i)}, x_i)$ is the predictive variance at location i given the conditioning set, $c(i)$. Since distance in the input space, via $\Sigma(\cdot, \cdot)$ is fundamental to GP inference and prediction, one can think of $c(i)$ as defining a "neighborhood." In that context it makes sense (as it did for OK and LAGP) to include in the neighborhood those indices whose input values $X_{c(i)}$ are closer to x_i. But hold that thought for a moment while I focus on computational details.

Eq. (3.5) is similar to (2.6) except instead of performing one $N \times N$ matrix decomposition (for inverse and determinant) in $\mathcal{O}(N^3)$ time, the Vecchia approximation involves N smaller inversions of $m \times m$ matrices, requiring $\mathcal{O}(Nm^3)$ flops. If m is small, typically between 10 and 25 [26, 56], $\mathcal{O}(Nm^3)$ is quasilinear in N [54]. Further computational gains can be realized through sparse-matrix libraries and parallelization by re-writing the likelihood through a Cholesky decomposition of the precision matrix of Y_N, denoted as U:

$$L(\phi; Y_N) \approx (2\pi)^{-\frac{N}{2}} \left(\prod_{i=1}^{N} \sigma_i^2 \right)^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{N} (Y_N^\top U_i U_i^\top Y_N) \right\} \tag{3.6}$$
$$= (2\pi)^{-\frac{N}{2}} |UU^\top|^{\frac{1}{2}} \exp\left\{ -\frac{1}{2} (Y_N^\top UU^\top Y_N) \right\},$$

where U_i is a $1 \times N$ vector whose j^{th} entry is:

$$U_i^{(j)} = \begin{cases} \frac{1}{\sigma_i} & i = j \\ -\frac{1}{\sigma_i}\left(\Sigma(x_i, x_j)\left(\Sigma(X_{c(i)}, X_{c(i)})^{-1}\right)^{(j,j)}\right) & j \in c(i) \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

Observe that there are no $N \times N$ matrix decompositions involved, and that any inverses are implicit in the Cholesky factor $UU^\top$, which is sparse.

One may maximize the likelihood (3.6) to estimate hyperparameters [42]. Prediction follows the classical setup (2.2), forming $Y(\mathcal{X}) \mid Y_N$ by stacking training and testing responses:

$$\begin{bmatrix} Y_N \\ Y(\mathcal{X}) \end{bmatrix} \sim \mathcal{N}_{N+N'}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} UU^\top = \sum_{i=1}^N U_i U_i^\top & \sum_{i=1}^N U_i \sum_{i=1}^{N'} U_i'^\top \\ \sum_{i=1}^{N'} U_i' \sum_{i=1}^N U_i^\top & U'U'^\top = \sum_{i=1}^{N'} U_i' U_i'^\top \end{bmatrix}^{-1}\right).$$

Here I have introduced a new $N' \times N'$ matrix, U' following Eq. (3.7), via $\mathcal{X}$ rather than X_N. Then, the analog of Eq. (2.2) yields

$$\begin{aligned} \mu_N(\mathcal{X}) &= -(U'U'^\top)^{-1} \sum_{i=1}^{N'} U_i' \sum_{i=1}^{N} U_i^\top Y_N \\ \Sigma_N(\mathcal{X}) &= U'U'^\top. \end{aligned} \tag{3.8}$$

With global mean and variance predictions, Vecchia models do everything classical kriging/GP models can do which is potentially better than both OK and LAGP because Vecchia provides a full joint distribution. This is especially appealing because a single prediction, after training, requires just $\mathcal{O}(m^3)$ additional time [56], assuming cached values of U from MLE calculations. A total of $\mathcal{O}((N'+N)m^3)$ flops are required for inference and prediction, as opposed to $\mathcal{O}(N'^3 + N^3)$ for an ordinary GP.

All that remains is to determine the ordering of indices i in y_i and the composition of the neighborhood sets $c(i)$, since not all choices (when $m \ll n$) lead to equally good approximations (3.4–3.5). One option is to follow the LAGP playbook and attempt to optimize over these variables. However this has proved elusive in the literature because an exhaustive search over alternatives would be combinatorially cumbersome, and there is no obvious greedy approach that enjoys submodularity for active learning. Nevertheless there are rules of thumb that make sense intuitively. Many orderings work well [42, 53, 91], but there is a consensus in the literature [26, 92, 104] for random indexing. Likewise, those authors prefer NN conditioning sets $c(i)$ comprised of indices $j < i$ whose x_j-values are closest to x_i. This choice has been dubbed NNGP by Datta et al. [26], although it is important to note that NN are not being used in the same way as LAGP or OK, and thus not in the typical way that NN are used in general in machine learning.

Since distances are involved in NN calculations, the Vecchia approximation faces the same "chicken or the egg" problem as LAGP in the face of anisotropy. To help, Katzfuss et al. [56] describe a scheme similar to pre-scaling for LAGP which updates lengthscales $\hat{\theta}_k$ via Fisher scoring [73] (similar to MLE calculations), and re-scales inputs so that NNs can be recalculated, and repeats. [56] [56] call this "scaled Vecchia" (SVecchia), and argue that it works best with a maximin [47] indexing. I adopt SVecchia as the preferred variation on this theme, in part because it is neatly packaged in software [Section 3.3]. However, note that Wycoff et al. [105] have demonstrated that simpler pre-scaling schemes work equally well.

Figure 3.5 shows an illustration using simulated borehole data, providing conditioning sets of size $m = 10$ for two points, labeled with indices $i = 4$ (triangles) and $i = 115$ (circles) respectively. The left plot shows what the conditioning sets look like in the raw, unscaled space while the right is after scaling x_1 by $\frac{2}{3}$; both using the same maximin ordering for easy comparison. The scaling has no effect on the small indices, since point 4, for example,

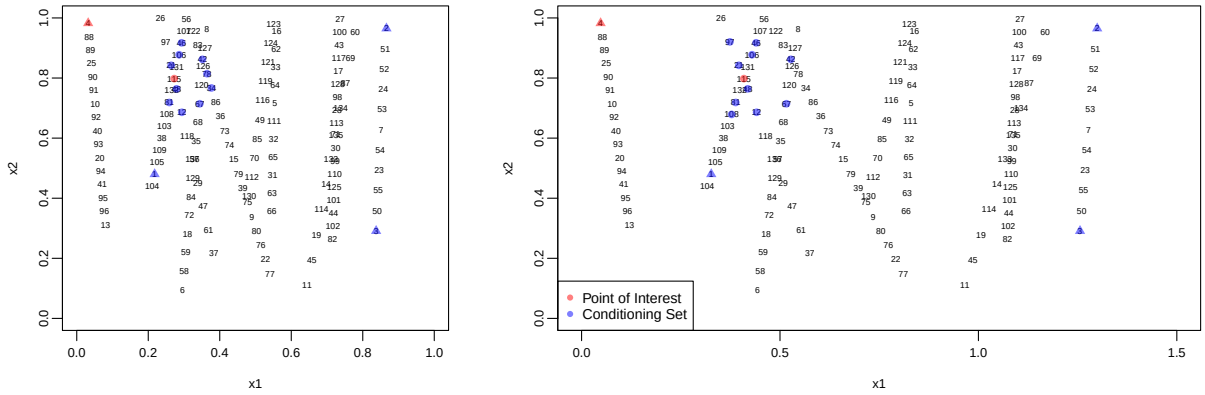


Figure 3.5: Conditioning sets for two inputs using $m = 10$: point 4 (triangles) and 115 (circles). Left shows the original space in coded inputs, and right shows x_1 pre-scaled by $1/\sqrt{\theta_1} = 2/3$.

can only condition on $k(4) = c(4) = \{1, 2, 3\}$. See that the lower indices in $c(4)$ (purple triangles) are spread out through space due to maximin ordering which Katzfuss et al. [56] argued was important. However, point 115 has $|k(115)| = 114$ points to choose from for it's neighborhood $c(115)$. Thus the prescaling has an impact on which points are selected for $c(115)$. Observe that the conditioning sets are different between the unscaled and scaled versions. For example, point 97 is closer than point 34 in the scaled plot, but farther in the unscaled plot. With high-indexed points spread via maximin, it is possible to ensure many conditioning points nearby, although not exclusively filled with NNs due to indexing requirements. In this way, SVecchia embodies the spirit of LAGP-style nearest and satellite points while producing a global model for smooth predictive surfaces.

3.3 Model evaluation on ore data

Here I shall expand on the out-of-sample analysis from Section 3.1 in order to illustrate how modern approximate GP and kriging alternatives [Section 3.2] compare on two, real and large-scale ore data sets. The layout is as follows. I first introduce the data, evaluation

metrics, and detail software in Section 3.3.1. Section 3.3.2 presents my validation apparatus, which offers a subtle twist on conventional methods to respect the borehole nature of data acquisition/measurement. Here I also provide results from my first sets of comparisons. Finally, Section 3.3.3 presents one of several potentially more nuanced analyses that, I believe, is only possible (with ease) in the GP setting, i.e., with full probabilistic modeling and hands-off automated calculation: coping with left-censoring prevalent in ore measurements.

3.3.1 Implementation details

The ore data involve three-dimensional inputs, representing positions measured in longitude, latitude and depth in standard units. Although these data record measurements (potential ore outputs) for multiple elements, the analysis here focuses on gold concentration in parts per million. I take the logarithm of this concentration as the response variable. The two data sets record gold concentrations at geographically disparate mining sites that are characterized by different ore forming processes. The first data set involves more than 150,000 measurements from approximately four-thousand boreholes; the second has $N \approx 500{,}000$ from 8,000 holes. The second data set also has a substantial number of left-censored values (i.e., thresholded measurements below the detection limits of the apparatus used to sample the core). For example, about 40% of the gold measurements in these data are recorded as 0.05. There are a smaller number of higher, identical limiting values as well. I shall detail how I handle this with two different treatments later in Sections 3.3.2 and 3.3.3. I am deliberately being vague about many aspects of the data in order to honor confidentiality agreements with mine operators.

I wish to draw an out-of-sample comparison between the methods in Section 3.2 on these data. In addition to RMSE and proper score (3.1–3.2), I also report time, considering

both compute (machine) time and practitioner (human) time. Machine time is measured precisely, in seconds, for execution on an eight-core hyperthreaded Intel Core i9-9900K CPU at 3.6GHz with 128GB RAM and Intel MKL linear algebra subroutines. Human time is more subjective/imprecise, and I shall have more to say about that in due course. It is worth remarking that none of the small-data/exact methods from Chapter 2 are applicable when $N \gg 10{,}000$, as I have here. Approximation is essential. One such approximation, i.e., beyond those from Section 3.2, is simply to (randomly) subset the data to a manageable size and apply exact inference on that subset. In the passages which follow I refer to this method as "subset GP," and entertain varying subset sizes.

For the approximate methods of Section 3.2, I leverage the following software libraries. The `laGP` package [37] for R [75] on CRAN accommodates subset GP, LAGP, and SLAGP. Its implementation is primarily in C, with `OpenMP` for symmetric multi-processing (SMP) parallelization. Only Gaussian kernels are supported by this software. I use defaults throughout, including ALC neighborhoods of size $m = 50$. An SVecchia implementation is provided by Katzfuss et al. [56],[1] which piggy-backs off of two R packages: `GpGp` [42] and `GPVecchia` [55]. Although primarily in R under-the-hood, `Rccp` [31] is used for key subroutines. These, and other calculations based on sparse matrix libraries [6] for efficient linear algebra, are also SMP parallelized, although to a lesser degree compared with `laGP`. Here I use the Matèrn kernel and other defaults throughout, including a conditioning set size of $m = 25$. Conventional pre-processing is used to code spatial inputs to the unit 3-cube as recommended by both `laGP` or `SVecchia` documentation.

OK is provided by `GSLIB` [29], which is in Fortran, and commonly assisted by a shell scripting interface. Whereas the other methods are plug-and-play modulo with few choices that are largely relegated to defaults, interacting with GSLIB is a human-intensive endeavor, involv-

[1] https://github.com/katzfuss-group/scaledVecchia

ing extensive pre-processing. For example, the analyst must first generate an experimental semivariogram (2.10) before fitting a positive-definite kernel function. When anisotropic spatial correlation is suspected, the analyst may also have to prepare a variogram map to identify the principal directions of spatial continuity, and then generate experimental semivariograms for two or three directions. This task is also complicated by numerous choices that are part of the analysis, e.g., angular tolerances for directional search windows and different models for spatial autocorrelation (exponential, spherical, Gaussian, etc.) When the variogram analysis is complete, the analyst may also have to complete a coordinate axis rotation to align the dataset with the directions of maximum and minimum spatial correlation, and also experiment with numerous choices for the OK algorithm itself, e.g., search window size and shape, minimum/maximum observations to be use for each estimate. When working with spatially isotropic data, pre-processing may require only an hour or so of additional work; however, this time expense increases dramatically when faced with spatially anisotropic data.

3.3.2 Validation exercise

For the warmup Meuse river validation exercise of Section 3.1, I randomly held out ten percent of the data for testing, training and building predictions on the rest. Repeating that randomization multiple times mitigates the so-called Monte Carlo (MC) error metrics like RMSE and score. Here I use $K = 10$-fold *cross validation* [CV; 43, Chapter 7] to average over train–test partitions while controlling MC error further by ensuring that each data element is used exactly once for testing, and complementarily exactly nine times for training. CV commences by first shuffling the data, and then evenly dividing it into a partition of K mutually-exclusive *folds*, then iterating over those folds $k = 1, \ldots, 10$, forming a testing set of the data in the k^{th} fold while taking the complement as the training set. In this way,

K metric evaluations (like RMSE, score or time) can be calculated and summarized for comparison. An attractive feature of CV is that each data element is used exactly once for testing, and exactly $K - 1$ times for training.

Early attempts at a CV evaluation of the methods in Section 3.2, after this fashion, revealed a shortcoming in the context of the borehole-driven ore data sets. Namely, the best predictors of a particular held-out testing element were almost always comprised entirely of members of the training data coming from the same borehole. With boreholes "holding" approximately thirty to sixty data elements each, depending on the hole and the data set, this meant that it was highly probable that an accurate prediction could be made trivially just by those nearby evaluations. This conveyed a substantial advantage to OK and LAGP. I determined that it would be more realistic, and more fair, to hold out entire boreholes for testing, rather than partitioning the data on individual data elements regardless of which borehole they were in. The idea is to simulate what might happen if I was to predict measurements for a new borehole that has not been drilled yet, i.e., to entertain how the spatial predictors behave as a surrogate.

Toward that end, I built a custom CV which randomly partitioned the data into K folds of roughly equally-sized boreholes instead. In this way, boreholes are "tested" all at once, without being able to lean on other data within the same borehole for training. Figure 3.6 shows one such training and testing partition via a single fold of this "borehole-preserving CV." Finally, it is worth remarking that all of the comparator methods use exactly the same CV folds.

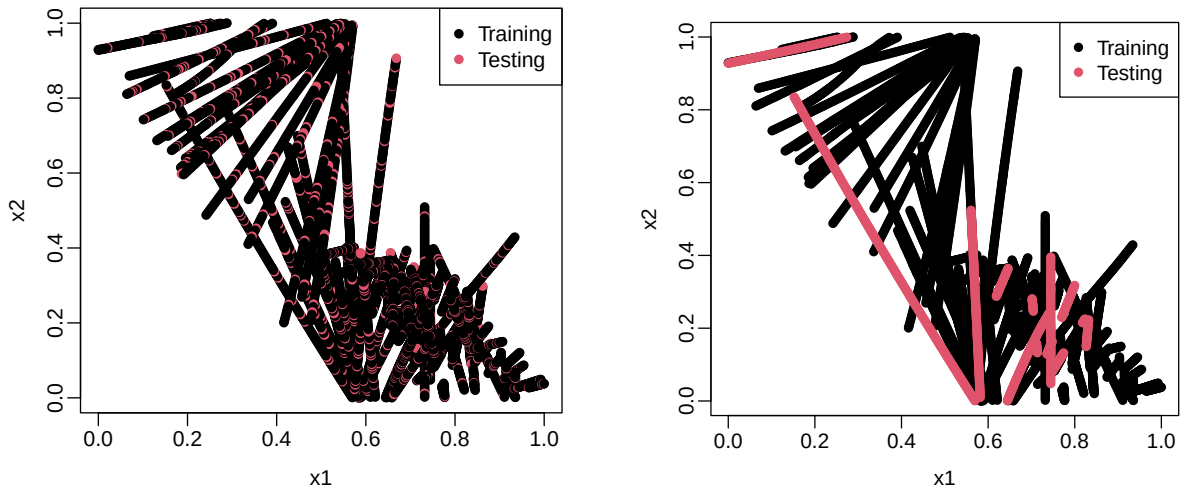


Figure 3.6: Testing and training sets for the 2d project from Figure 3.1 with ordinary CV on the left and borehole-preserving CV on the right.

Ore data set one

With this setup, Figure 3.7 shows log RMSE (3.1), score (score$_p$ from Eq. (3.2)) and compute time for each of the methods for the first, smaller data set. The big takeaways are that SVecchia, OK and SLAGP are all competitive with each other in terms of RMSE; SLAGP and SVecchia are competitive in score with similar medians. Observe that SLAGP improves upon ordinary LAGP for both RMSE and score. Score for OK could not be calculated because GSLIB does not furnish predictive variances. GSLIB provides standard errors on the mean of the prediction, but those can substantially under-estimate out-of-sample variance. Consequently, I cannot assess UQ for OK. In terms of computation and human time, SVecchia takes seconds and LAGP takes a couple minutes to run, both with essentially zero "human time." OK takes several hours of human time to perform a variography analysis, choosing between competing kernel formulations and parameterization and to determine an appropriate rotation and pre-scaling of the data in order to cope with an otherwise isotropic formulation. After that has been done, training and prediction takes about the same amount

of time as (S)LAGP. It is interesting that SLAGP is faster than LAGP despite involving more algorithmic steps: first fit a global subset model, then local models on transformed inputs. The explanation is that, after pre-scaling, local MLE calculations are easier: they require many fewer iterations to converge. Computation time for the subset-GP methods explodes with $\mathcal{O}(m^3)$ flops as m grows.

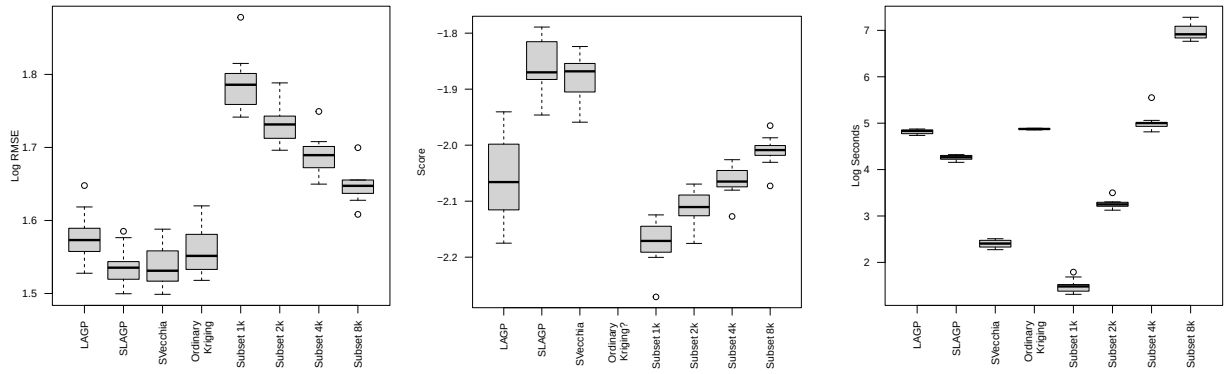


Figure 3.7: Drillhole-presererving 10-fold CV summary for the first data set. Left: RMSE (smaller is better); Middle: score$_p$ (higher is better). Right: compute time (smaller is better).

My conclusion from this experiment is that, although SLAGP edges out SVecchia on accuracy and UQ for this problem, SVecchia is slightly better overall because of its substantially lighter time commitment. However, for an individual (or entire borehole) prediction, SLAGP times are orders of magnitude faster – the times in the right panel of Figure 3.7 are for all boreholes in the fold – because each calculation is independent of others. For a one-off prediction it is the clear winner. All together, I conclude that this is a substantial win for modern machine learning methods. Although raw accuracy is similar compared to OK, the machine learning GP methods are hands-off, provide full UQ, and are faster to train/predict.

Ore data set two

A similar analysis for the second, larger data set, is nuanced because of the substantial left-censoring. One option is to ignore the censoring and treat the recorded values as the actual values. If there were a small number of such values, sporadically located in the input space, this might work well. However, there are a sizable number (more than 40%), and they cluster nearby one another. Having some responses smoothly vary in the input space, with others "flatlining" at 0.05, say, for most or all of a borehole represents an almost pathological contrast to typical smoothness assumptions underlying GP (and kriging) methods. So to start with, I removed these values, and dealt with borehole-preserving CV only on the remaining 259,555 data records. In Section 3.3.3 I shall discuss an imputation scheme for bringing these observations back into the folds. In the remaining data there is still a moderate degree of left-censoring which I largely ignore except when an entire borehole contains the same (thresholded) gold response. In that case, I collapse those records into three data points – two ends and midway point – all with the same gold value. This collapsing is especially important for LAGP and OK because, due to their local nature, those methods occasionally have neighborhoods consisting of data from one or two boreholes only. If those comprise of largely censored values, the lack of diversity in training can result in numerical singularities in predictive (2.2) and likelihood (2.6) equations.

Even after such modifications, I found that LAGP and OK struggled to predict at some testing sites. `GSLIB`, implementing OK, would simply refuse to provide a prediction in these instances, or similarly when there are no training data points within a user-specified radius (regardless of m), returning an error code. In these data, that amounts to about 300 testing sites per fold, a substantial proportion. The `laGP` software would furnish a prediction, but when comparing the corpus of other predictions in a fold it was obvious that something was amiss, particularly with the estimated (local) nugget parameter and, consequently, the

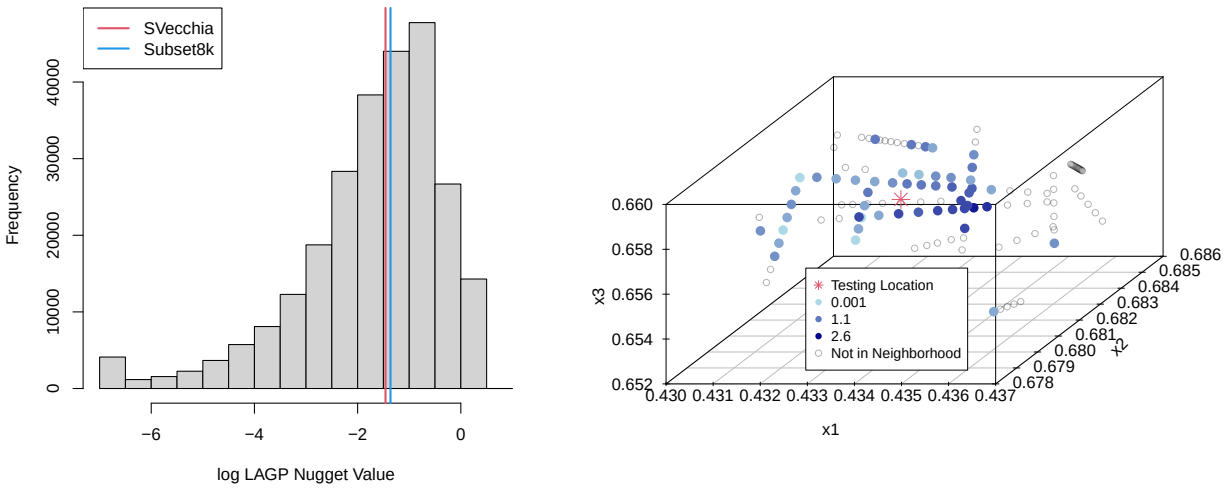


Figure 3.8: Left: Histogram of log LAGP nuggets with SVecchia and subset8k nuggets. Right: An example neighborhood using LAGP's ALC for a point with low estimated variance.

predicted variance. To obtain further insight, I plotted a histogram of the estimated nuggets from all of the local fits, shown in the left panel of Figure 3.8, and compared these against the global nugget(s) provided by subset and SVecchia methods. I observed that occasionally, local nuggets were being estimated at the lower-threshold imposed by the `laGP` default search range (leftmost-bin in the histogram). I looked at the local neighborhood for one, representative member of this group. See the right panel of the figure. At a glance, it appears that this point should be easy to predict given the local neighborhood; most of the points nearby have log gold values above 1 (are similar shades of blue) with some satellite points having lower log gold values. Thus a prediction of log gold a bit higher than 1 with low variance makes sense. However, the true log gold value for this point is about 0.1, meaning the model is very certain and is very wrong – which would lead to a poor score evaluation.

Of course, I do not know this location provides a bad prediction without looking at the testing value for this predictive location. So I decided to replace local nuggets estimated at the lower-bound of `laGP` search range with the median of the nuggets from the rest of the distribution. This led to a substantial improvement in out-of-sample scores, described

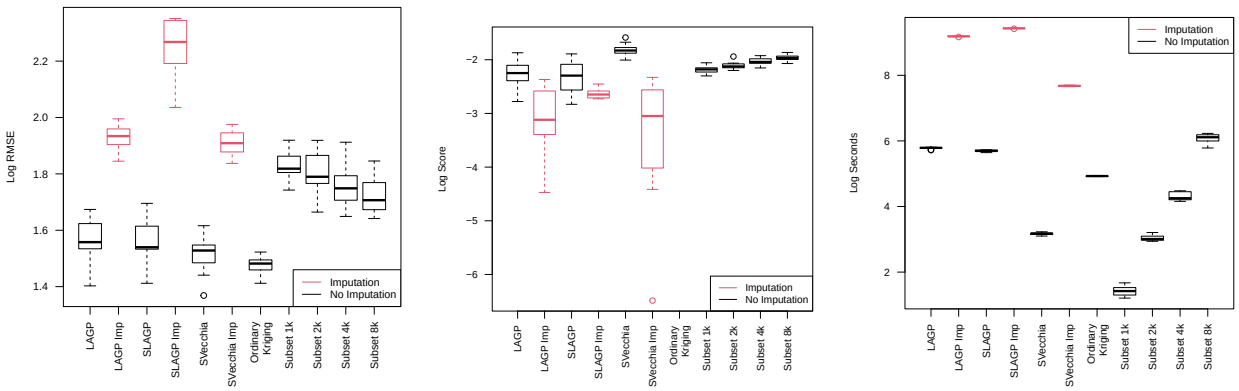


Figure 3.9: Drillhole-preserving 10-fold cross validation summary for the first data set. See Figure 3.7 caption. Red boxplots are discussion in Section 3.3.3.

momentarily. If this sounds *ad hoc*, that is because it is. But a prediction with UQ that is based on compromise and a limited degree of post-hoc human intervention is better than no prediction at all (OK/`GSLIB`).

Figure 3.9 shows these results with black boxplots. (The red ones involve a study on imputation in Section 3.3.3, so ignore those for now.) The story is similar to the first data set in Figure 3.7: SLAGP, SVecchia, and OK outperform subsetting in terms of RMSE. Here, OK appears to be the most accurate, but these RMSE calculations do not include any error-coded outputs (representing more than 300 presumably "bad" predictions per fold). So this is not a holistic assessment of OK accuracy. By score, which again cannot be calculated for OK, SVecchia is the clear winner, and the second-fastest in this comparison. (S)LAGP, which is similar in spirit to OK, has inferior scores despite modifications to address stability issues to do with the nugget (above).

3.3.3 Imputation

One of the advantages to a fully probabilistic generative model, besides a streamlined (fast) human-free inferential procedure, is that it provides a framework where extension and down-

stream application are readily supported by whatever procedures are already in place for such enterprises in other contexts. For example, if I wish to ask where a new borehole should be drilled, I could take advantage of methods in Bayesian optimization [49]. When the new data measurements arrive, fits and predictive equations can be be updated efficiently (i.e., in quadratic rather than cubic time) through standard updating equations [e.g., 36]. The whole procedure can be automated, creating a virtuous cycle between learning and data acquisition.

A more concrete example – i.e., one that does not require drilling to illustrate – involves coping with censored data. It is unsatisfying to discard data. Even a coarsely left-censored value contains information, which can be used to enhance training. Perhaps even more importantly, one may wonder how accurately those censored values may be predicted, thereby increasing the resolution of those measurements, by borrowing information from higher-accuracy (training) data measurements nearby. This is a standard enterprise in statistical learning when fully probabilistic generative modeling is used. There are many options when handling "missing data," of which censoring is one example [59, 83].

One way to incorporate censored ore values, without destroying smoothness or stationarity assumptions underlying GP spatial models, is through *imputation* [e.g., 59, Chapter 5]. In this context, imputation basically means generating a plausible response Y-value for censored locations that both respects the censored measurement, and the smoothness of the underlying spatial field learned through other, completely observed data. Once generated, the imputed value may be treated as if it were a completely observed value going forward, say for prediction. Of course, treating an imputed value as observed ignores the uncertainty in the imputation. *Multiple imputation* acknowledges that uncertainty by randomly imputing several possible values and performing inference based on the corpus of those imputed values, e.g., through averaging.

Illustrating how this could work in the ore context requires some notational scaffolding. Let $D_N = (D_{\text{obs}}, D_{\text{cens}})$ represent the partition of the complete data set into its fully observed and censored components, respectively. For example, $D_{\text{obs}} = (X_{\text{obs}}, Y_{\text{obs}})$, may be the portion of the second data set I was working with in Section 3.3.2, and $D_{\text{cens}} = (X_{\text{cens}}, Y_{\text{cens}})$ was the part I (temporarily) discarded. Imputed values $Y_{\text{imp}}(X_{\text{cens}})$, may be used to augment D_{obs} to obtain $D_{\text{imp}} = (D_{\text{obs}}, (X_{\text{cens}}, Y_{\text{imp}}))$ via truncated Gaussian simulation

$$Y_{\text{imp}} \sim \mathcal{N}_{N_{\text{cens}}}(\mu_{\text{obs}}(X_{\text{cens}}), \Sigma_{\text{obs}}(X_{\text{cens}}))\mathbb{I}_{\{Y_{\text{imp}} \leq Y_{\text{cens}}\}}, \tag{3.9}$$

where $\mathbb{I}_{\{Y_{\text{imp}} \leq Y_{\text{cens}}\}}$ is an indicator function, returning 1 if $Y_{\text{imp}} \leq Y_{\text{cens}}$ and 0 otherwise. Quantities $\mu_{\text{obs}}(X_{\text{cens}})$ and $\Sigma_{\text{obs}}(X_{\text{cens}})$ are the predictive moments (2.2) of a GP fit (or large scale approximation such as those from Section 3.2) conditioned on D_{obs}.

Although software exists to sample from a truncated MVN directly [e.g., 103], such as those in (3.9), in practice it can be difficult to generate a sufficient number of values below Y_{cens} when N_{cens} is of modest size, for example in the hundreds [58]. A more customized approach that acknowledges the form of the (approximate/large-scale) spatial surrogates helps. In the (S)LAGP context, one may use Eq. (3.4) to sample Y_{imp} from the truncated MVN (3.9) one at a time, conditioning on the previously sampled imputed values and the observed data. LAGP is designed to look at each location in the testing set independently which makes this setup work. On the other hand, SVecchia is designed to give a global model approximation, so doing a similar one-at-a-time conditional imputation is too crude. I instead prefer a bespoke rejection sampling [18] scheme that proceeds in epochs: first generate posterior samples from the MVN (3.9) unconstrained, keeping any values that satisfy the censoring threshold. Then condition on those imputations and the observed values, resampling at locations without an imputed value, repeating until Y_{imp} is completely filled in.

Algorithm 1: Multiple imputation for large scale GPs under left censoring

input D_{obs} and D_{cens} and testing locations $\mathcal{X}$

for $i = 1, \ldots, M$ **do**

  if *(S)LAGP* **then**

    $D = D_{\text{obs}}$

    for $j = 1, \ldots, N_{\text{cens}}$ **do**

      $(\mu, \sigma^2) = (\text{S})\text{LAGP}(D, X_{\text{cens}}[j])$ `// LAGP prediction: Sec. 3.2.1`

      $Y_{\text{imp}}[j] \sim N_1\left(\mu, \sigma^2\right) \mathbb{I}_{\left\{Y_{\text{imp}}[j] \leq Y_{\text{cens}}\right\}}$ `// Posterior tnorm: Eq. (3.9)`

      $D = (D, (X_{\text{cens}}[j], Y_{\text{imp}}[j]))$ `// Imputation step`

    $(\mu_i, \sigma_i^2) = (\text{S})\text{LAGP}(D, \mathcal{X})$ `// Predict at testing locations`

  if *SVecchia* **then**

    $D = D_{\text{obs}}$

    while $|X_{\text{cens}}| > 0$ **do**

      $(\mu, \Sigma) = \text{SVecchia}(D, X_{\text{cens}})$ `// SVecchia prediction: Eq. (3.8)`

      $Y_{\text{imp}} \sim N_{|X_{\text{cens}}|}(\mu, \Sigma)$ `// Unconstrained posterior MVN draw`

      $w = \text{which}(Y_{\text{samp}} < Y_{\text{cens}})$

      $D = (D, (X_{\text{cens}}[w], Y_{\text{imp}}[w]))$ `// Imputation step`

      $X_{\text{cens}} = X_{\text{cens}} \setminus X_{\text{cens}}[w]$

    $(\mu_i, \Sigma_i) = \text{SVecchia}(D, \mathcal{X})$ `// Predict at testing locations`

return (μ_i, Σ_i), for $i, \ldots, m$ `// where` $\Sigma_i = \text{diag}(\sigma_i^2)$ `for (S)LAGP`

Algorithm 1 provides pseudo-code for concreteness, wrapping a single imputation with a `for` to obtain M multiple imputations. [79] indicates that M between two and ten works well, so I use $M = 5$ in the exercises. Each of the M imputed values are plausible realizations of the censored measurements, which correspond to M posterior/predictive Gaussian distributions for each testing location. Thus I use Gaussian mixture moment equations [78] to report the mean and variance predictions for the testing set:

$$\mu_{\text{MI}}(\mathcal{X}) = \frac{1}{M} \sum_{i=1}^{M} \mu_i(\mathcal{X}) \tag{3.10}$$

$$\sigma^2_{\text{MI}}(\mathcal{X}) = \frac{1}{M} \sum_{i=1}^{M} \sigma_i^2(\mathcal{X}) + \frac{1}{M} \sum_{i=1}^{M} \mu_i^2(\mathcal{X}) - \left(\frac{1}{M} \sum_{i=1}^{M} \mu_i(\mathcal{X}) \right)^2$$

While this cannot completely account for all possible uncertainties due to imputation, be-

cause I have not looked at all possible imputation values (only M), one can always increase M if desired. It may be shown that these equations give an unbiased estimate of mean and variance for any M.

Imputation in practice

To begin with an illustration in a simple, controlled setting, the left panel of Figure 3.10 shows a classical GP model with and without imputation. The true function is $f(x) = 2\sin(4\pi x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.1)$ and an observation threshold of $y = 1$ with $n = 20$ randomly selected training points. There are two main regions of censoring, one in the center and one at the upper end of inputs. In the center, the model with imputation gets closer in mean to the truth. On the upper end, the variance of the predictions is much lower when conditioning on imputed values.

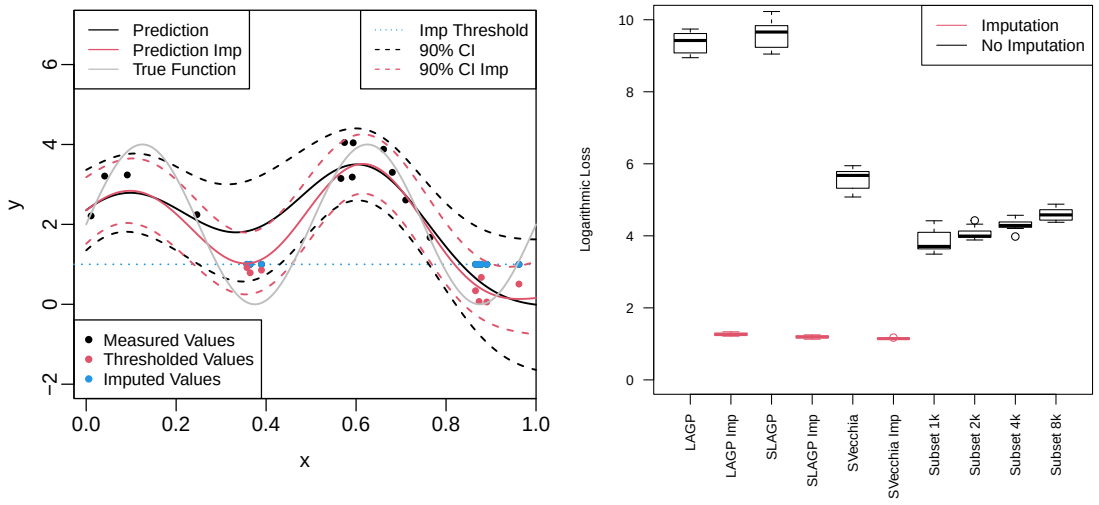


Figure 3.10: Left: Imputation illustration for 1d synthetic data. Right: CV results for log loss (3.11) on the values from ore data set two that are below the detection threshold using big data methods with and without imputation.

With these visuals, it is readily apparent that the model with imputation is a better fit for the true curve. When working with real data, one does not have the luxury of knowing

what the "true curve" is. Consequently, it is much tougher to assess improvement in the quality of fit through imputation. RMSE and score on an out-of-sample testing set, say via CV, are problematic because censored data values – as elements of the testing set – are not realizations from the same population as the model/predictive quantities are ($\mu_n(x)$ and $\sigma_n^2(x)$). The only "truth" one knows about these points is that they are measured to be below the threshold. To improve the resolution on a model fit's ability to accurately predict these values, I effectively turn the regression problem into one of binary classification: does the model accurately predict the "parity" of its recorded value, either above or below the threshold. The GP and kriging framework makes this transition simple. The probability of accurate prediction under a threshold may be obtained through inverse Gaussian CDF evaluated at the threshold. The proper scoring mechanism [33] for such probabilities is the logarithmic loss [LL; 34], also known as cross-entropy loss in the neural network literature. Lower LL is better. When all of the testing data are from one class (less than Y_{cens}), LL boils down to:

$$\begin{aligned} \text{LL}(\mathcal{X}_{\text{cens}}) &= -\frac{1}{N'_{\text{cens}}} \sum_{x \in \mathcal{X}_{\text{cens}}} \log(\mathbb{P}(Y(x) \leq Y_{\text{cens}})) \\ &= -\frac{1}{N'_{\text{cens}}} \sum_{x \in \mathcal{X}_{\text{cens}}} \log\left(\Phi^{-1}(Y_{\text{cens}}; \mu(x), \sigma^2(x))\right). \end{aligned} \tag{3.11}$$

Returning now to the second ore analysis from Section 3.3.2, I describe my experience with multiple imputation on these data. I only explore (S)LAGP and SVecchia in this context. When dealing with the subset methods, imputation is of limited additional value as completely observed data are plentiful relative to the subset size. It is cumbersome, but not impossible to entertain imputation under OK. Being faithful to the imputation scheme would require human intervention to re-fit variograms after each new imputation is obtained. That would result in over 100K variogram fits in this example! In this context, I see LAGP

as an equivalent, automatic variation on OK that can be more easily entertained in an imputation/censored values context.

The right plot of Figure 3.10 demonstrates that my imputation scheme yields improved LL (3.11) across the board. For reference, $-\log(0.001) = 6.9$ and $-\log(0.2) = 1.6$, so fits leveraging imputation provide a probability of 0.2 for being below thresholds, on average, compared to 0.001 or worse for the models without imputation. I take this as evidence of success, in particular when it comes to out-of-sample prediction in parts of the input space where low gold ore measurements lie. At first, it is hard to square this with what appears to be contrary messaging from the imputation results in Figure 3.9 (red boxplots). If a practitioner is certain that a particular region is high in gold concentration, *a priori*, then those results indicate that simply dropping the censored values (i.e., no imputation), which are all low-measurements, leads to more accurate results. However, dropping thresholded values exposes the practitioner to confirmation bias. Predictions appear more accurate in parts of the input space that conform to that bias, but can result in incorrect, massive over-predictions of those values. This is what the right panel of Figure 3.10 shows.

Although demanding more computational effort, imputation schemes with modern GP libraries are easily automated and not onerous. I see no reason not to do it both ways with modern GP libraries. Thus I highly recommend using imputation when modeling elements with large portions of the data measured at a detection threshold, particularly in the exploratory drilling phase.

3.4 Discussion

I have highlighted some of the similarities and differences in approaches to machine learning (GPs) and geospatial (kriging) modeling spatial data. In theory kriging and GPs are not all

that different. Terminology obscures a practically identical modeling apparatus. The biggest practical differences lie in respective approaches to hyperparameter estimation, treatment of anisotropy which is twinned with a different degree of automation/requisite human intervention. I demonstrated that with smaller training data sets, expert intervention through variography can lead to (slightly) better fits. However, I advocate for taking the human out of the loop even in this situation, and relying on likelihood based criteria for estimating hyperparameters. Although human intervention has the potential to improve results, it challenges reproducibility and limits the scope for downstream tasks in more ambitious modeling contexts, such as with with censored data.

By introducing modern kriging/GP approximations, I have shown that when working with large mining datasets in low dimension, the scaled Vecchia approximation is faster and at least as accurate as LAGP/OK with the benefit of giving full joint posterior distributions. Throughout, I leveraged off-the-shelf libraries with defaults settings. These represent the tip of an iceberg. See, e.g., Heaton et al. [44] for a survey of additional modern approaches to modeling spatial data. Mining data often have some non-stationarity due to geological structures (e.g., faults, folds, and fracture networks) which may lead to violations in stationarity. Recent work in machine learning and surrogate modeling have suggested that a deep GP [24, 82] handles such non-stationary data more gracefully than ordinary GPs: combining the accuracy of deep learning (deep neural networks) with the uncertainty quantification (UQ) features of fully probabilistic (GP) modeling.

Finally, GP-based machinery – as opposed to kriging – is less intimately tied to low-dimensional (2-3d) input spaces, allowing for the incorporation of more predictor variables. In the case of mining for gold, measurements for other minerals (in a similar domain) can be used to help build models that are more accurate and give better UQ. This can easily increase the input dimension to the tens of variables, which challenges variography, but in-

volves nearly identical MLE calculations in the GP context. In the modern age of plentiful measurement (i.e., training data), it is important to develop, and leverage, tools designed to cope with data at scale – i.e., as opposed to being tied down to two or three dimensions.

Chapter 4

Robust Bayesian Optimization

My goal in this chapter is to extend BO via GPs and EI to a richer, more challenging class of optimization problems. From Chapters 2 and 3, you hopefully see that using the kriging framework would be infeasible for BO. Kriging is hands-on, requiring constant supervision for each model creation and often limits data to two or three dimension which is not the case for many computer experiments that leverage BO. In situations where there are a multitude of competing, roughly equivalent local solutions to the global optimization problem, a practitioner may naturally (or at least intuitively) express a preference for obtaining or exploring, as the global solution, those which enjoy a wider *domain of attraction*; i.e., those whose troughs are larger. Borrowing from the previous chapter, one does not want to find the one specific spot with the most gold, but rather the best group of nearby spots to build a gold mine. As an easy to visualize 1d example, consider $f(x)$ defined in Eq. (4.7), provided later in Section 4.2.2, as characterized by the black line in Figure 4.1. It has two local minima, one of which has a narrow domain of attraction but lower objective value compared to the other. Such is the case in the robot pushing simulator [Section 4.3]: an "optimal" parameter setting can be thrown off by imprecisions in robotic movement leading to poor performance

of the robot. Although BO via GPs/EI would likely explore both domains of attraction to a certain extent *eventually*, it will in the near term (i.e., for smaller simulation budgets) focus on the deeper/narrower one, and thus provide lower resolution on the solution than many practitioners may prefer.

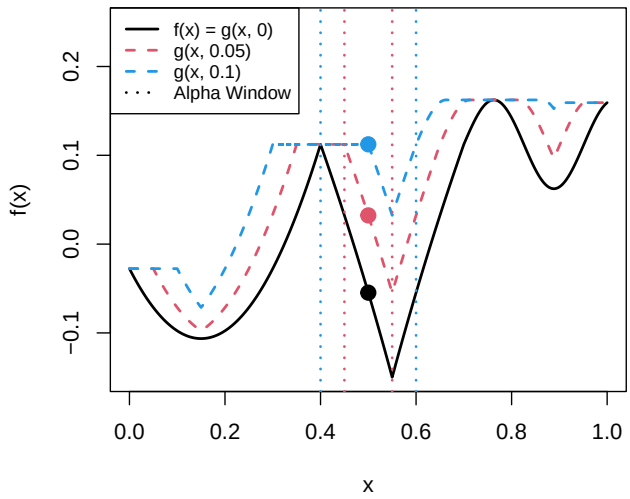


Figure 4.1: Surface for a 1d multimodal function shown with different adversary levels.

Finding/exploring the *robust* global solution space requires a modification to both the problem, and the BO strategy. Here I borrow a framework first introduced in the math programming literature on robust optimization [e.g., 67]: an *adversary*. Adversarial reasoning is popular in the wider, reinforcement learning literature [46], of which some regard AL and BO as special cases. To my knowledge the mathematical programming notion of an adversary has never been deployed for BO, where it is perhaps best intuited as a form of penalty on "sharp" local minima. Specifically, let x^α denote the *α-neighborhood* of an input x. There are many ways to make this precise depending on context. If x is one-dimensional, then $x^\alpha = [x - \alpha, x + \alpha] \equiv [x \pm \alpha]$ is sensible. In higher dimension, one can generalize to an α-ball or hyper-rectangle. More specifics will come later. Relative to that α-neighborhood,

an adversary $g(x, \alpha)$ and robust minimum x^r may be defined, respectively, as follows:

$$\text{Let} \quad g(x,\alpha) = \max_{x \in x^\alpha} \; f(x) \qquad \text{so that} \qquad x^r = \operatorname*{argmin}_{x \in [0,1]^d} \; g(x,\alpha). \tag{4.1}$$

Figure 4.1 provides some examples for the f introduced earlier. Observe that the larger α is, the more "flattened" $g(x, \alpha)$ is compared to $f(x)$. In particular, note how the penalization is more severe in the sharper minimum compared to the shallow trough which is only slightly higher than the original, unpenalized function.

The conventional, local, approach to finding the robust solution x^r is best understood as an embellishment of other, conventional, local schemes inspired by Newton's method [e.g., 20]: extract derivative (and adversarial) information nearby through finite differencing and evaluation at the boundary of x^α; then take small steps that descend the (adversarial) surface. Such schemes offer tidy local convergence guarantees but are profligate with evaluations. When f is expensive, this approach is infeasible. I believe that idea of building an adversary can be ported to the BO framework, making better use of limited simulation resources toward global optimization while still favoring wider domains of attraction.

The crux of my idea is as follows. A GP $\hat{f}_n$ that would typically be fit to a collection of n evaluations of f in a BO framework can be used to define the adversarial realization of those same values following the fitted analog of g: $\hat{g}(x_i, \alpha) = \max_{x \in x_i^\alpha} \hat{f}(x)$, and second GP $\hat{f}_n^\alpha$ can be fit to those $\hat{g}(x_i, \alpha)$-values, $i = 1, \ldots, N$. I call $\hat{f}_n^\alpha$ the *adversarial surrogate*. Then you can proceed as you wolud with $\hat{f}_n$ in a BO framework with $\hat{f}_n^\alpha$ instead, for example use EI to guide acquisitions. I call this *robust expected improvement (REI)*. There are, of course, myriad details and variations – simplifications and embellishments – that I am glossing over here, and that I shall be more precise about in due course. The most interesting of these may be how I suggest dealing with a practitioner's natural reluctance to commit to a particular

choice of α *a priori*.

Before jumping in with details, it is worth remarking that the term "robust" has many definitions across the statistical and optimization literature(s). My use of this term here is more similar to some than others. I do not mean robust to outliers [63], as may be applicable to when only noisy, leptokuric simulators f are involved [7] – though I shall have some thoughts on this setup later. Some refer to robustness to the choice of random initialization of a local optimizer [94], however my emphasis is global. My definition in Eq. (4.1) and its BO implementation bears some similarity to Oliveira et al. [72] and to so-called "unscented BO" [71], where robustness over noisy or imprecisely specified *inputs* is considered. However, the adversary entertains a worst-case [e.g., 13], rather than the stochastic/expected case. Marzat et al. [64] also consider worst-case robustness, however they are slightly different as they are focused on so-called "minimax" problems where some dimensions are perfectly controlled ($\alpha = 0$) and others are completely uncontrolled ($\alpha = 1$). Nonetheless, these methods and their test cases make for interesting empirical comparisons, as I demonstrate.

Now I will extend the EGO algorithm by incorporating adversarial thinking. The goal is to find x^r from Eq. (4.1) for some α. I begin by presuming a fixed, known α selected by a practitioner.

4.1 Novel Bayesian optimization contributions

Before I dive into my novel methodology, there are a couple additions to my general introduction to GPs from Section 2.1. First, I would like to point out that doing the following through kriging and variography would be infeasible. The human costs would far outweigh the benefits obtained from BO, so this chapter only uses GPs. For this chapter, I assume

the incoming data from $f(x)$ are deterministic, meaning $g = 0$ from Eq. (2.3). In practice, setting the nugget to exactly 0 often provides numerical instability, so I let $g = 10^{-8}$. I shall use the short hand of $\text{GP}_{\hat{\theta}}(D_n)$ to indicate a fitted GP surrogate $\hat{f}_n$ to data $D_n = (X_n, Y_n)$ emitting predictive equations $(\mu_n(\cdot), \sigma_n^2(\cdot))$ as in Eq. (2.2) conditioned on estimated hyperparameters $\hat{\theta}_n$. Note that I am also streamlining the notation here somewhat by subsuming $\hat{\tau}^2$ into $\hat{\theta}$.

4.1.1 The adversarial surrogate

If $\hat{f}_n(x)$ is a surrogate for $f(x)$, then one may analogously notate $\hat{f}_n^\alpha(x)$ as a surrogate for $g(x, \alpha)$, an adversarial surrogate. I envision several ways in which $\hat{f}_n^\alpha(x)$ could be defined in terms of $\hat{f}_n(x)$, but not many which are tractable to work with analytically or numerically. For example, suppose $Y^\alpha(x) = \max_{x' \in x^\alpha} Y(x')$, where $Y(x') \sim \mathcal{N}(\mu_n(x'), \sigma_n^2(x'))$, a random variable whose distribution follows the spirit of Eq. (4.1), but uses the predictive equations of Eq. (2.2). The distribution of $Y^\alpha(x)$ could be a version of $\hat{f}_n^\alpha(x)$, at least notionally. However a closed form remains elusive.

Instead, it is rather easier to define $\hat{f}_n^\alpha(x)$ as an ordinary surrogate trained on data derived through adversarial reasoning on the original surrogate $\hat{f}_n(x)$. Let Y_n^α denote these *adversarial responses*, where each y_i^α, for $i = 1, \ldots, n$, follows

$$y_i^\alpha = \max_{x \in x_i^\alpha} \mu_n(x) \quad \text{where } x_i^\alpha \text{ is the } \alpha\text{-neighborhood of the } i^{\text{th}} \text{ entry of } X_n, \text{ as usual.} \tag{4.2}$$

There are many sensible choices for how to find y_i^α numerically in practice. Newton-based optimizers, e.g., BFGS [16], could leverage closed form derivatives for $\mu_n(x)$, or utilize finite differencing. A simpler option that works well is to instead take $y_i^\alpha = \max_{x \in x_i^{\mathcal{B}_\alpha(x)}} \mu_n(x)$, where $\mathcal{B}_\alpha(x)$ is the discrete set of points on the corners of a box with sides of length α

emanating from x. Details for my own implementation are deferred to Section 4.2.1.

Given Y_n^α, an adversarial surrogate may be built by modeling *adversarial data* $D_n^\alpha = (X_n, Y_n^\alpha)$, i.e., the adversarial responses paired with their original inputs, as a GP. Let $\hat{\theta}_\alpha$ denote hyperparameter estimates for GP surrogate $\hat{f}_n^\alpha(x)$. Fill in the covariance matrix following Eq. (2.4), $\Sigma^\alpha(\cdot,\cdot)$. Finally, define the *adversarial surrogate* as

$$\hat{f}_n^\alpha(x) \equiv \mathrm{GP}_{\hat{\theta}_\alpha}(D_n^\alpha) \rightarrow (\mu_n^\alpha(\cdot), \sigma_n^{2\alpha}(\cdot)) \tag{4.3}$$

via novel hyperparameter estimates $\hat{\theta}_\alpha$ using Y_n^α rather than Y_n with predictive equations akin to (2.2). Since the Y_n^α are the original surrogate's ($\hat{f}_n$) estimate of adversarial response values according f_α, $\hat{f}_n^\alpha$ may serve as a surrogate for $g(x,\alpha)$ from the left half of Eq. (4.1).

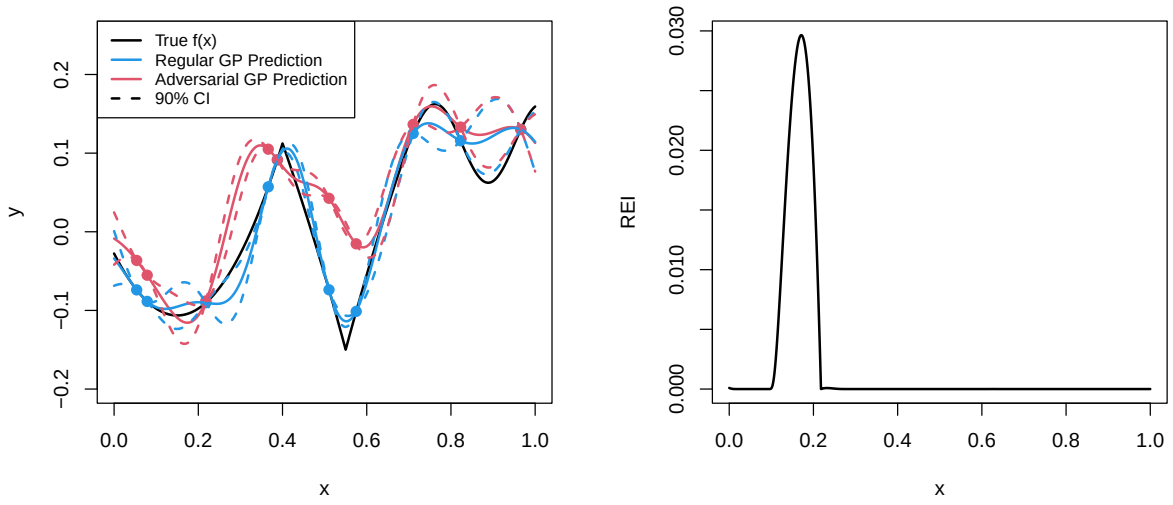


Figure 4.2: Left: The same black and blue lines from Figure 2.4. In red, Y_n^α are shown as the points with $\hat{f}_n^\alpha(x)$ as the curve. Right: Robust expected improvement for the current design.

As a valid GP, $\hat{f}_n^\alpha$ can be used in any way another surrogate might be deployed downstream. For example, $\hat{f}_n^\alpha$ may be used to acquire new runs via EI, but hold that thought for a moment. To illustrate $\hat{f}_n^\alpha$, the left panel of Figure 4.2 augments the analogous panel in Figure 2.4 to include a visual of $\hat{f}_n^\alpha$ via μ_n^α and error-bars $\mu_n^\alpha \pm \sigma_n^\alpha$. Perhaps the most notable takeaway

from this graph is that the new, red lines, are much higher near the sharp minimum (near $x = 0.5$), but less dramatically elevated for the wider, dull trough near $x = 0.2$.

Also observe in Figure 4.2 that, whereas for the most part $\mu_n(x) \leq \mu_n^\alpha(x)$, this reverses for a small portion of the input domain near $x = 0.2$. That would never happen when comparing $f(x)$ to $g(x, \alpha)$ via Eq. (4.1). This happens for these surrogates – original $\hat{f}_n$ and adversarial $\hat{f}_n^\alpha$ – for two reasons. One is that both are stationary processes because their covariance structure (2.4) uses only relative distances, resulting in a compromise surface between sharp and dull. Although this is clearly a mismatch to the data-generating mechanism, I have not found any downsides in our empirical work. The possibility of improved performance for non-stationary surrogates is entertained in Section 4.4. The other reason is that the adversarial data Y_n^α, whether via a stationary or (speculatively) a non-stationary surrogate $\hat{f}_n$, provide a low-resolution view of the true adversary $g(x, \alpha)$ when n is small. Consequently, $\hat{f}_n^\alpha$ is a crude approximation to $g(x, \alpha)$, but that will improve with more acquisitions (larger n). What is most important is the EI surface(s) that the surrogate(s) imply. These are shown in the right panel of Figure 4.2, and discussed next.

4.1.2 Robust expected improvement

Robust expected improvement (REI), $\mathrm{EI}_n^\alpha(\cdot)$, is the analog of in Eq. (2.13) except using $(\mu_n^\alpha(\cdot), \sigma_n^{2\alpha}(\cdot))$, towards solving the mathematical program given in the right half of Eq. (4.1). Let $f_n^{\alpha,\min} = \min(y_1^\alpha, \ldots, y_n^\alpha)$ denote the best estimated adversarial response (BEAR). Then, let

$$\begin{aligned} \mathrm{EI}_n^\alpha(x) = \mathbb{E}\{I(x) \mid D_n^\alpha\} = (f_n^{\alpha,\min} - \mu_n^\alpha(x))\Phi\left(\frac{f_n^{\alpha,\min} - \mu_n^\alpha(x)}{\sigma_n^\alpha(x)}\right) \\ +\sigma_n^\alpha(x)\phi\left(\frac{f_n^{\alpha,\min} - \mu_n^\alpha(x)}{\sigma_n^\alpha(x)}\right). \end{aligned} \tag{4.4}$$

The right panel of Figure 4.2 shows the REI surface for the same initial setup as Figure 2.4. There is only one peak, located at the shallower, wider trough compared to three peaks in the EI surface [Figure 2.4]. Generally speaking, REI surfaces have fewer local maxima compared to EI surfaces because $\hat{f}_n^\alpha(x)$ smooths over the peaked regions. Thus the inner optimization of REI often has fewer local minima, generally requiring fewer multi-starts.

Algorithm 2: Robust Expected Improvement

input $D_n = (X_n, Y_n)$, x, and α

$\hat{f}_n(x) = \mathrm{GP}_{\hat{\theta}}(D_n)$ `// with predictive moments in Eq. (2.2)`

for $i = 1, \ldots, n$ **do**

  $y_i^\alpha = \mathrm{argmax}_{x' \in x_i^\alpha} \mu_n(x')$ `// adversarial responses, (4.2)`

$D_n^\alpha = (X_n, Y_n^\alpha) = (X_n, \{y_i^\alpha\}_{i=1}^n)$

$\hat{f}_n^\alpha(x) = \mathrm{GP}_{\hat{\theta}_\alpha}(D_n^\alpha)$ `// the adversarial surrogate, (4.3)`

return $(\mathrm{EI}_n^\alpha(x))$ `// EI using` $\hat{f}_n^\alpha(x)$ `from the previous line, (4.4)`

REI is summarized succinctly in Eq. (4.4) above, but it is important to appreciate that it is a result of a multi-step process. Alg. 2 provides the details: fit an ordinary surrogate, which is used to create adversarial data, in turn defining the adversarial surrogate upon which EI is evaluated. The algorithm is specified for a particular reference location x, used toward solving $x_{n+1} = \mathrm{argmax}_{x \in [0,1]^d} \mathrm{EI}_n^\alpha(x)$. It may be applied identically for any x. In a numerical solver it may be advantageous to cache quantities unchanged in x.

With repeated acquisition via EI, over $n = n_0, \ldots, N-1$ being known as EGO, I dub robust efficient global optimization (REGO) as the repeated application of REI towards finding a robust minimum. REGO involves a loop over Alg. 2, with updates $D_n \to D_{n+1}$ after each acquisition. The final data set, $D_N = (X_N, Y_N)$ provides insight into both $f(x)$ and $g(x, \alpha)$ and their minima. Whereas EGO would report BOV $f_N^{\min}$, and/or the corresponding element

$x^\star_{\text{bov}}$ of X_N, REGO would report the BEAR $f_N^{\alpha,\min}$, the same quantity used to define REI in Eq. (4.4), and/or input $x^\star_{\text{bear}}$.

Post hoc adversarial surrogate

To quantify the advantages of REI/REGO over EI/EGO in our empirical work of Sections 4.2–4.3, I consider a *post hoc adversarial surrogate*. This is the surrogate constructed after running all acquisitions, $n_0+1, \ldots, N$ via EI/EGO, then at N fitting an adversarial surrogate Eq. (4.3), and extracting the BEAR $f_N^{\alpha,\min}$ rather than the BOV. In other words, the last step is faithful to the adversarial goal, whereas active learning aspects ignore it and proceed as usual. Whereas the BOV from EGO can be a poor approximation to the robust optimum $f(x^r)$ of Eq. (4.1), the BEAR from a post hoc adversarial surrogate can potentially be better. Comparing two BEAR solutions, one from REGO and one from a post hoc EGO surrogate, allows us to separately explore the value of REI acquisitions from post hoc adversarial surrogates, $\hat{f}_N^\alpha$.

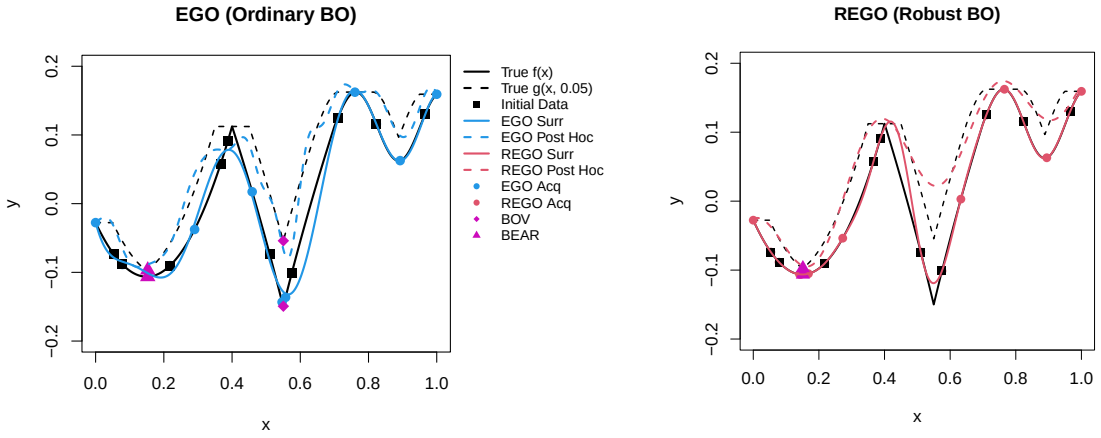


Figure 4.3: Left: Surrogate and post hoc surrogate fits after EGO acquisitions ($N = 20$ and $n_0 = 10$) on Eq. (4.7). Right: Same except REGO where BOV and BEAR are the same.

To illustrate on our running example, consider the outcome of EGO, from the left panel of Figure 2.4, recreated in the left panel of Figure 4.3. In both, the union of black and blue

points comprise D_N. In Figure 4.3 the dashed blue curve provides the post hoc adversarial surrogate, $\hat{f}_N^{\alpha}$, from these runs, with the BOV and BEAR indicated as magenta diamonds and triangles, respectively. Notice that for EGO, the BOV is around the global, peaked minimum, but the BEAR captures the shallow, robust minimum. The analogous REGO run, via EI acquisitions (red circles) and adversarial surrogate (red lines) is shown in the right panel of Figure 4.3. Here BEAR and BOV estimates are identical since REGO does not explore the peaked minimum.

Looking at both panels of Figure 4.3, the BEAR offers a robust solution for both EGO or REGO. However, EGO puts only one of its acquisitions in the wide trough compared to four from REGO. Consequently, REGO's BEAR offers a more accurate estimate for x^r. With only $N = 20$ points in 1d, any sensible acquisition strategy should yield a decent meta-model for $f(x)$ and $g(x, \alpha)$, so I may have reached the limits of the utility of this illustrative example. In Sections 4.2–4.3 I intend to provide more compelling evidence that REGO is better at targeting robust minima. Before turning to that empirical study, I introduce two REI variations that feature in some of those examples.

Unknown and vector-valued α

Until this point I have presumed fixed, scalar α, perhaps specified by a practitioner. In the case of inputs x coded to the unit cube $[0, 1]^d$, a choice of $\alpha = 0.05$, say, represents a robustness specification of 5% in each coordinate direction (totaling 10% of each dimension), i.e., where $x^{\alpha} \equiv [x_j \pm \alpha : j = 1, \ldots, d]$. This intuitive relationship can be helpful in choosing α, however one may naturally wish to be "robust" to this specification. Suppose instead I had an $\alpha_{\max}$ in mind, where $0 \leq \alpha \leq \alpha_{\max}$. One option is to simply run REGO with $\alpha \leftarrow \alpha_{\max}$, i.e., basing all REI acquisitions on $\alpha_{\max}$ instead. Although I do not show this, performance (via BEAR) of $\alpha_{\max}$ relative to a nominal α rapidly deteriorates as the gap

between $\alpha_{\max}$ and α widens.

A better method is to base REI acquisitions on *all* α-values between zero and $\alpha_{\max}$ by integrating over them: $\overline{\mathrm{EI}}_n^{\alpha_{\max}} = \int_0^{\alpha_{\max}} \mathrm{EI}_n^{\alpha}(x)\ dF(\alpha)$, say over uniform measure F. I am not aware of an analytically tractable solution, in part because embedded in $\mathrm{EI}_n^{\alpha}(x)$ are complicated processes such as determining adversarial data, and surrogates fit thereupon. But quadrature is relatively straightforward, either via Monte Carlo (MC) $\alpha^{(t)} \sim F$, for $t = 1, \ldots, T$, over a grid $\{0 = \alpha^{(1)}, \ldots, \alpha^{(T)} = \alpha_{\max}\}$. Define

$$\overline{\mathrm{EI}}_n^{\alpha_{\max}} \approx \frac{1}{T}\sum_{t=1}^{T} \mathrm{EI}_n^{\alpha^{(t)}}(x) \quad \text{for the former,} \quad \text{or} \quad \overline{\mathrm{EI}}_n^{\alpha_{\max}} \approx \frac{1}{T}\sum_{t=1}^{T} \mathrm{EI}_n^{\alpha^{(t)}}(x) F(\alpha^{(t)}), \tag{4.5}$$

for the latter. Both may be implemented by looping over Alg. 2 with different $\alpha^{(t)}$ values, and then averaging to take $x_{n+1} = \mathrm{argmax}_{x \in [0,1]^d} \overline{\mathrm{EI}}_n^{\alpha_{\max}}$. Both approximations improve for larger T.

With random $\alpha^{(t)} \sim F$, even a single draw ($T = 1$) suffices as an unbiased estimator – though clearly with high variance. This has the advantage of a simpler implementation, and faster execution as no loop over Alg. 2 is required. In the empirical work to follow, I refer to this simpler option as the "rand" approximation, and the others (MC or grid-based) as "sum." It is remarkable how similarly these options perform, relative to one another and to a nominal REI with a fixed α-value. One could impose $\alpha_{\min} > 0$ similarly, though I do not entertain this variation here.

As a somewhat orthogonal consideration, it may be desirable to entertain different levels of robustness – different α-values $\{\alpha_1, \ldots, \alpha_d\}$ – in each of the d coordinate directions. The only change this imparts on the description above is that now $x^{\alpha} = [x_j \pm \alpha_j : j = 1, \ldots, d]$, a hyperrectangle rather than hypercube. Other quantities such as $g(x, \alpha)$ and $\mathrm{EI}_n^{\alpha}(x)$ remain as defined earlier with the understanding of a vectorized $\alpha = \{\alpha_1, \ldots, \alpha_d\}$ under the hood.

Using vectorized $\alpha_{\max}$ in Eq. (4.5) requires draws from a multivariate F, which may still be uniform, or a higher dimensional grid of $\alpha^{(t)}$, recognizing that it is now approximating a higher dimensional integral.

4.2 Implementation and benchmarking

Here I provide a description of my implementation, variations, those of the main competitors, evaluation metrics and ultimately provide a suite of empirical comparisons.

4.2.1 Implementation details

My main methods (REI/REGO, variations and special cases) are coded in R [75]. These codes may be found, alongside those of the comparators and all empirical work in this paper, in my public Git repository: `https://bitbucket.org/gramacylab/ropt/`. GP surrogate modeling for my new methods – the competitors may leverage different subroutines/libraries – is provided by the `laGP` package [37], on CRAN. Those subroutines, which are primarily implemented in C, leverage squared exponential covariance structure (2.4), and provide analytic $\hat{\tau}^2$ conditional on lengthscale θ. In my experiments, θ is fixed to appropriate values in order to control MC variation that otherwise arises in repeated MLE-updating of $\hat{\theta}$, especially in small-n cases. More details are provided as I introduce our test cases, with further discussion in Section 4.4. Throughout I use $g = 10^{-8}$ in Eq. (2.3), as appropriate for deterministic blackbox objective function evaluations. For ordinary EI calculations I use add-on code provided by Gramacy [38, Chapter 7.2.2]. All empirical work was conducted on an 8-core hyperthreaded Intel i9-9900K CPU at 3.60 GHz with Intel MKL linear algebra subroutines.

Section 4.1.1 introduced several possibilities for solving y_i^α, whose definition is provided in Eq. (4.2). Although numerical optimization, e.g. BFGS, is a gold standard, in most cases I found this to be overkill, resulting in high runtimes for all $(N - N_0)$ acquisitions with no real improvements in accuracy over the following, far simpler alternative. I prefer quickly optimizing over the discrete set $x^{\mathcal{B}_\alpha(x)}$ comprising of a box extending α-units out in each coordinate direction from x, or its vectorized analog as described in Section 4.1.2. This "cornering" alternative occasionally yields $y_i^\alpha < y_i$, which is undesirable, but this is easily mitigated by augmenting the box $x^{\mathcal{B}_\alpha(x)}$ to contain a small number of intermediate, grid locations in each coordinate direction. Using an odd number of such intermediate points ensures $y_i^\alpha \geq y_i$. I use three additional grid points per input in our higher-dimensional exercises, and none in lower dimensional ones. I find that a d-dimensional grid formed from the outer product for each coordinate (i.e., a $2^5 = 32$-point-grid for $d = 2$) facilitates a nice compromise between computational thrift, and accuracy of adversarial y_i^α-values compared to the cumbersome BFGS-based alternative.

In the test problems, which are introduced momentarily and utilize inputs coded to $[0, 1]^d$, I compare each of the three variations of REGO described in Section 4.1. These comprise: (1) REI with known α (Alg. 2); (2) novel $\alpha \sim \text{Unif}(0, \alpha_{\max})$ at acquisition; and (3) averaging over a sequence of $\alpha \in [0, \alpha_{\max}]$ (4.5). For all examples, I set $\alpha_{\max} = 0.2$ and average over five equally spaced values. In figures, these methods are denoted as "known", "rand" and "sum," respectively. After each acquisition, I use the post hoc adversarial surrogate to find the BEAR operating conditions: x^*_{bear} and $f_N^{\alpha,\min}$ in order to track progress.

As representatives from the standard BO literature, I compare against the following "strawmen": EGO [48], where acquisitions are based on EI (2.13; and "EY" [38] which works similarly to EGO, except acquisitions are selected minimizing $\mathbb{E}[Y(x) \mid D_N]$: $x^{\text{EY}}_{\text{new}} = \text{argmin}_{x\in[0,1]^d} \mu_n(x)$. In other words, EY acquires the point that the surrogate predicts has the lowest mean. Since

it does not incorporate uncertainty ($\sigma_n(x)$), repeated EY acquisition often stagnates in one region – usually a local minima – rather than exploring new areas like EI does. In figures, these comparators are indicated, as follows: "ego", and "ey" respectively for EGO and EY with progress measured by BOV: x^*_{bov} and $f^{\min}_N$. Finally, I consider the post hoc adversary with progress measured by BEAR for EGO ("egoph") and uniform random sampling ("unif"). In the figures, REI methods are solid curves with robust competitors dashed and regular BO dotted.

Our final comparator is `StableOPT` from [13]. For completeness, I offer the following by way of a high-level overview; details are left to their paper. Bogunovic et al. assume a fixed, known α, although I see no reason why our extensions for unknown α could not be adapted to their method as well. Their algorithm relies on confidence bounds to narrow in on x^r. Let $\text{ucb}_n(x) = \mu_n(x) + 2\sigma_n(x)$ denote the upper 95% confidence bound at x for a fitted surrogate $\hat{f}_n$, and similarly $\text{lcb}_n(x) = \mu_n(x) - 2\sigma_n(x)$ for the analagous lower bound. Then one may translate their algorithm into our notation, shown in Alg. 3, furnishing the n^{th} acquisition. Similar to REGO, this may then be wrapped in a loop for multiple acquisitions. I could not find any public software for `StableOPT`, but it was relatively easy to implement in R; see my public Git repo.

Algorithm 3: `StableOPT`

input $D_{n-1} = (X_{n-1}, Y_{n-1})$ and α
$\hat{f}_{n-1}(x) = \text{GP}_{\hat{\theta}}(D_{n-1})$ `// with predictive moments in Eq. (2.2)`
$\tilde{x}_n = \text{argmin}_{x\in[0,1]^d} \max_{a\in[-\alpha,\alpha]} \text{lcb}_{n-1}(x+a)$ `// where it thinks` x^r `is`
$a_n = \text{argmax}_{a\in[-\alpha,\alpha]} \text{ucb}_{n-1}(\tilde{x}_n + a)$ `// where it thinks the worst point in` $\tilde{x}^\alpha_n$ `is`
return $(\tilde{x}_n + a_n, f(\tilde{x}_n + a_n)))$ `// sample from` f `and return`

Rather than acquiring new runs nearby likely x^r, `StableOPT` samples the worst point within

$\tilde{x}_n^\alpha$. Consequently, its final X_N does not contain any points thought to be x^r. Bogunovic et al. recommend selecting x^*_{bear} from all $\tilde{x}_n$ rather than the actual points sampled, all $\tilde{x}_n + a_n$, using notation introduced in Alg. 3. I generally think it is a mistake to report an answer at an untried input location. So in the experiments I calculate x^*_{bear} for `StableOPT` as derived from a final, post hoc adversary calculation [Section 4.1.2]. In figures, this comparator is denoted as "stable".

The general flow of the benchmarking exercises coming next [Section 4.2.2] is as follows. Each optimization, say in input dimension d, is seeded with a novel LHS of size $n_0 = 5 + 5d$ that is shared for each comparator. Acquisitions, $n = n_0 + 1, \ldots, N$, separate for each comparator up to a total budget N (different for each problem), are accumulated and progress is tracked along the way. Then this is repeated, for a total of 1,000 MC repetitions. To simplify notation, let $x^*_{\text{b,n}}$ be either x^*_{bear} or x^*_{bov}, depending on if using BEAR or BOV, after the n^{th} acquisition. I utilize the following two metrics to compare BEAR or BOV across methods using the true adversary:

$$r(x^*_{\text{b,n}}) = g(x^*_{\text{b,n}}, \alpha) - g(x^r, \alpha) \qquad\qquad d(x^*_{\text{b,n}}) = ||x^*_{\text{b,n}} - x^r||, \tag{4.6}$$

where x^r is the x location of the true robust minimum. The first metric is similar to the concept of regret from decision theory [12, 50] at the suggested $x^*_{\text{b,n}}$. Regret measures how much you lose out by running at $x^*_{\text{b,n}}$ compared to x^r. Regret is always nonnegative since by definition, $x^r = \text{argmin}_{x \in [0,1]^d} g(x, \alpha)$. The second metric is the distance from $x^*_{\text{b,n}}$ to x^r. For both metrics, lower is better with 0 being the theoretical minimum if a method correctly identifies exactly x^r as the BEAR or BOV.

4.2.2 Empirical comparisons

One-dimensional examples

The left panel of Figure 4.4 shows the 1d RKHS function used by [4], and an adversary with $\alpha = 0.03$. Observe how f has a smooth region for low x values and a wiggly region

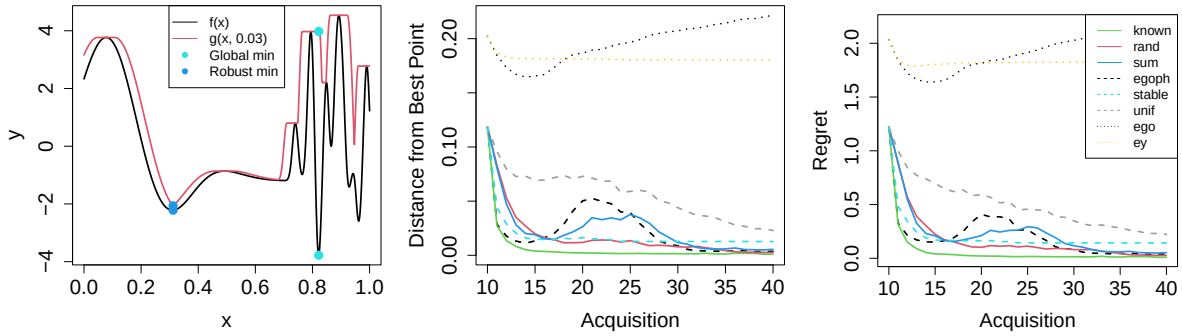


Figure 4.4: Left: The RKHS function and the robust surface with $\alpha = 0.03$. Middle: $d(x^*_{\text{b,n}})$ after each acquisition. Right: $r(x^*_{\text{b,n}})$ after each acquisition.

for high x. With $\alpha = 0.03$, $x^r = 0.312$ whereas $x^* = 0.822$. The adversary is bumped up substantially nearby x^* because the surface is so wiggly there. Here I consider a final budget of $N = 40$ with $\theta = 0.01$. Results are provided in the middle and right panels of Figure 4.4. Observe that REGO with known α performs best as it quickly gets close to x^r, and more-or-less stays there. EGO with post hoc adversary does well at the beginning, but after twenty acquisitions in, it explores around x^* more, retarding progress (explaining the "bump") toward x^r. "Sum"-based REI has a somewhat slighter bump. Averaging over smaller α-values favors exploring the wiggly region. `StableOPT` caps out at a worse solution than any of the REI-based methods. Those based on BOV, like "ego" and "ey" fare worst of all. Even worse than acquiring "unif"ormly at random.

Figure 4.5 shows a 1d test function of my own creation, first depicted in Figure 4.1, defined

as:

$$f(x) = \begin{cases} 3.5(x-0.15)^2 + \log(1.3) & x < 0.4 \\ \log(1 + |2(x-0.55)|) & 0.4 \leq x < 0.7 \\ \frac{1}{20}\sin(25x - 17.5) + \log(1.3) & 0.7 \leq x. \end{cases} \tag{4.7}$$

In this problem, $x^* = 0.55$ and $x^r = 0.15$ using $\alpha = 0.075$. All GP surrogates used $\theta = 0.25$. There is a similar story here as for our first test problem. The only notable difference here is that EGO does not get drawn into the peaky region, likely because it is less pronounced. EGO favors sampling around x^* initially, but eventually explores the rest of the input space, including around x^r. Interestingly, REGO with known α performs a little worse than either of the unknown α methods.

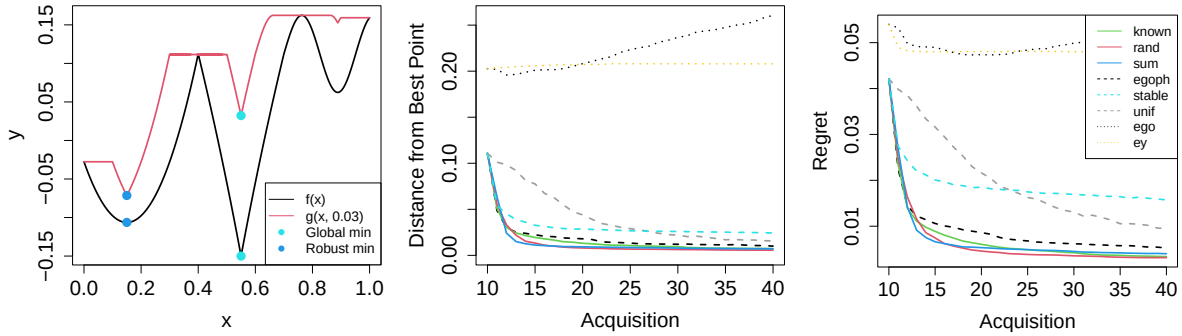


Figure 4.5: Left: The Figure 4.1 function and the robust surface with $\alpha = 0.075$. Middle: $d(x^*_{\text{b,n}})$ after each acquisition. Right: $r(x^*_{\text{b,n}})$ after each acquisition.

Two-dimensional examples

The top-left panel of Figure 4.6 shows a test problem from [8], defined as:

$$\begin{aligned} f(x_1, x_2) = &-2x_1^6 + 12.2x_1^5 - 21.2x_1^4 + 6.4x_1^3 + 4.7x_1^2 - 6.2x_1 - x_2^6 + 11x_2^5 - 43.3x_2^4 \\ &+ 74.8x_2^3 - 56.9x_2^2 + 10x_2 + 4.1x_1x_2 + 0.1x_1^2x_2^2 - 0.4x_1x_2^2 - 0.4x_1^2x_2. \end{aligned} \tag{4.8}$$

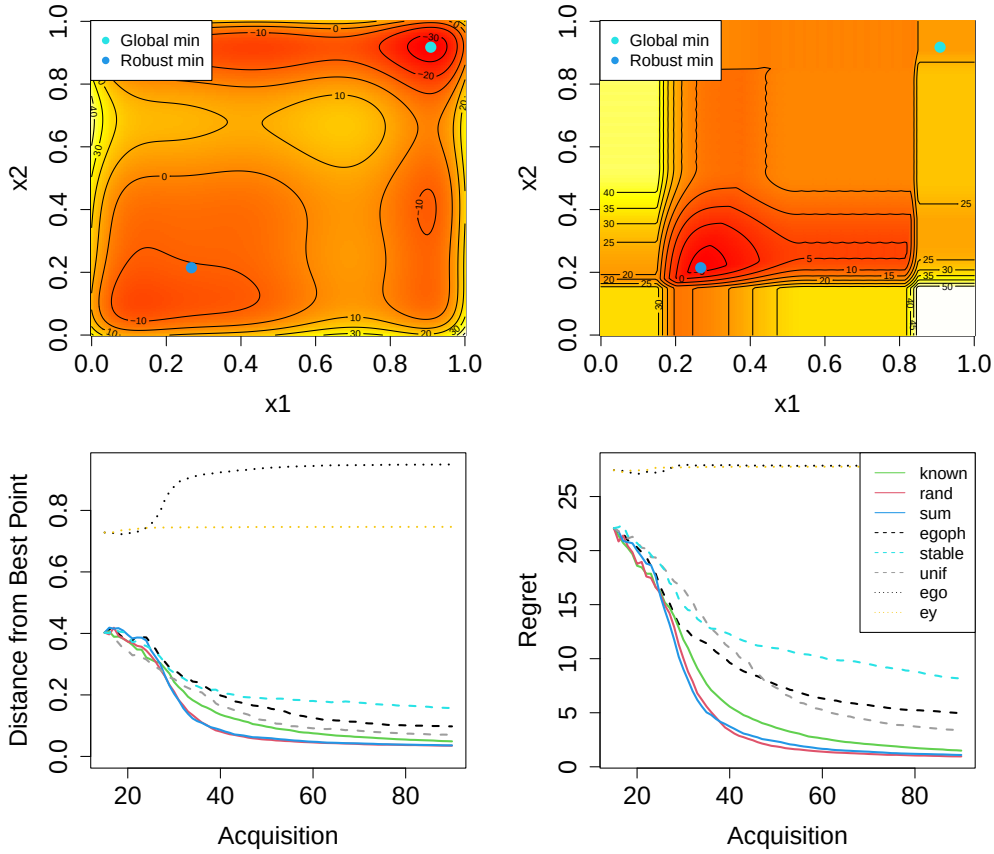


Figure 4.6: Top: The Bertsimas function on the left and the robust surface with $\alpha = 0.15$ on the right. Bottom: $d(x^*_{\mathrm{b,n}})$ (left) and $r(x^*_{\mathrm{b,n}})$ (right) after each acquisition.

I negated this function compared to Bertsimas et al., who were interested in maximization. Originally, it was defined on $x_1 \in [-0.95, 3.2]$ and $x_2 \in [-0.45, 4.4]$ with $x^* = (2.8, 4.0)$. I coded inputs to $[0, 1]^2$ so that the true minimum is at $(0.918, 0.908)$. In the scaled space, $x^r = (0.2673, 0.2146)$ using $\alpha = 0.15$. I used $\theta = 1.1$ and $N = 90$. The bottom row of panels in Figure 4.6 show that non-fixed α for REGO can give superior performance in early acquisitions. `StableOPT` performs worse with this problem because the objective surface near x^r is relatively more peaked, say compared to the 1d RKHS example. Regret is trending toward 0 when using BEAR for acquisitions, with REGO-based methods leading the charge. EGO with post hoc adversary performs much worse for this problem. EGO-based acquisitions heavily cluster near x^* which thwarts consistent identification of x^r even

with a post hoc surrogate.

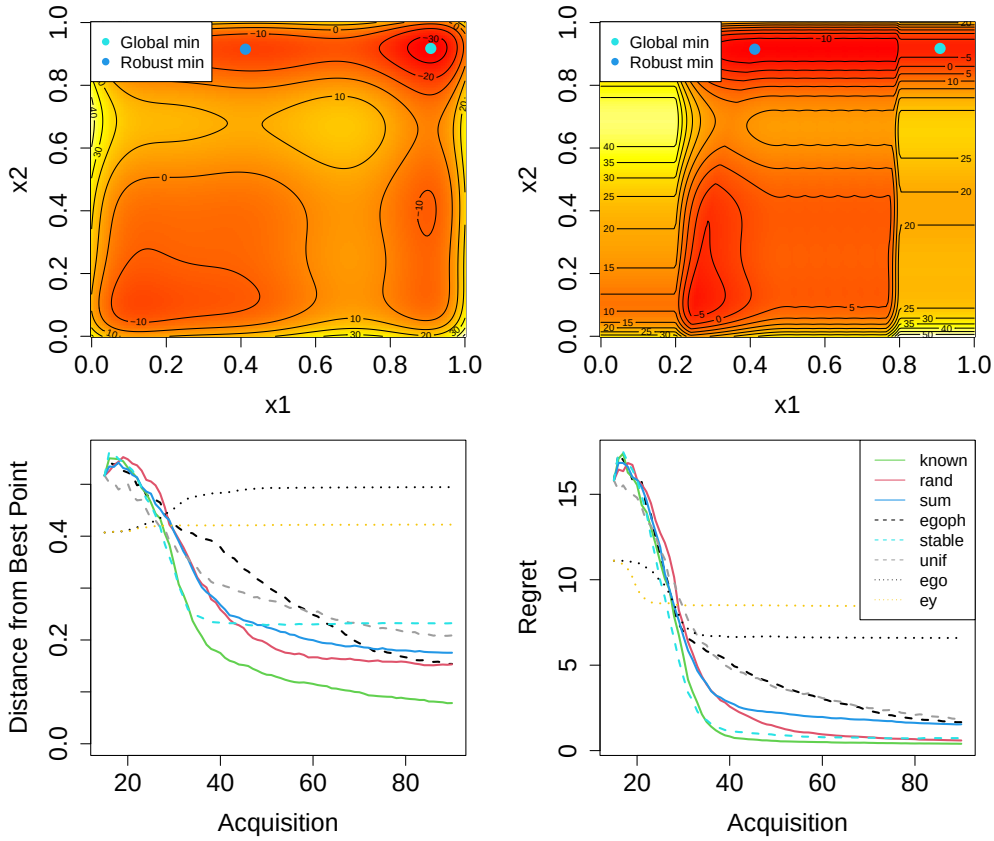


Figure 4.7: Top: The Bertsimas function on the left with the robust surface with $\alpha = (0.2, 0)$ on the right. Bottom: $d(x^*_{\text{b,n}})$ (left) and $r(x^*_{\text{b,n}})$ (right) after each acquisition.

Figure 4.7 considers the same test problem (4.8), except this time I use $\alpha = (0.2, 0)$, meaning no robustness required in x_2. This moves x^r to $(0.412, 0.915)$ in the scaled space. In this problem, the robust surface is fairly flat meaning that when an algorithm finds $x_2 = 0.915$, $x_1 \in [0.35, 0.75]$ gives a similar robust output. For that reason, all of the methods perform worse when trying to pin down the exact x_1 location, so by looking at metric $d(x^*_{\text{b}})$ from Eq. (4.6), all methods appear to be doing worse than looking at $r(x^*_{\text{b}})$ which goes to 0 relatively quickly. A shallower robust minimum favors `StableOPT`. Since that comparator never actually evaluates at x^r, on this problem it suffices to find $x_1 \in [0.35, 0.75]$, which it does quite easily. Knowing true α for REI helps considerably. This makes sense because

omitting an entire dimension from robust consideration is quite informative. Nevertheless, alternatives using random and aggregate α-values perform well.

Higher dimension

The final set of test functions comes from the Rosenbrock family [30],

$$\sum_{i=1}^{d-1} \left(100(x_{i+1} - x_i)^2 + (x_i - 1)^2\right). \tag{4.9}$$

defined in arbitrary dimension d. Originally in $[-2.48, 2.48]^d$, I again scale to $[0, 1]^d$. Although the focus here will be on $d = 4$, visualization easier in 2d. The top-left panel of

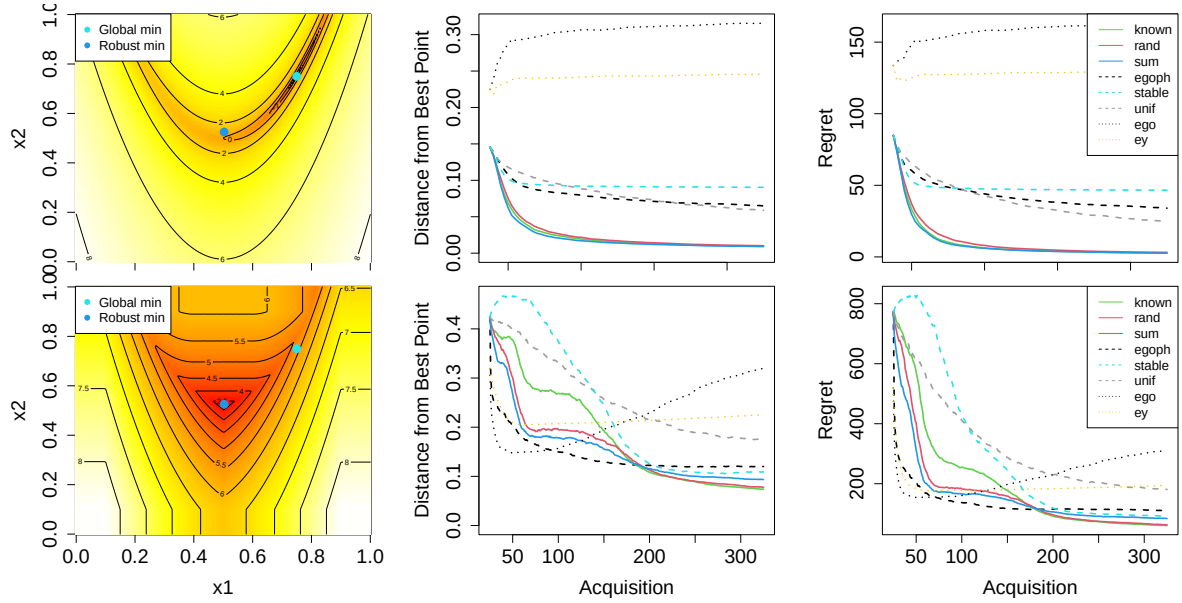


Figure 4.8: Left: 2d log Rosenbrock function on the top and the robust surface with $\alpha = 0.1$ on the bottom. Top: $d(x^*_{\text{b,n}})$ in the middle and $r(x^*_{\text{b,n}})$ on the right for 2d. Bottom: similarly in 4d.

Figure 4.8 shows a visual with outputs on the log scale for a 2d case. Here $x^* = (0.75, 0.75)$ and $x^r = (0.503, 0.525)$ when $\alpha = 0.1$. A 2d visual of this adversary is in the bottom-left panel. In 2d, I set $\theta = 0.9$ and in 4d, I use $\theta = 0.05$. Results for 2d are in the top row

(middle and right panels), and for 4d are in the bottom row, respectively. REGO shines in both cases because x^* is in a peaked region and x^r is in a shallow one. EGO with post hoc adversary does well early on, but stops improving much after about 150 acquisitions. `StableOPT` has the same issues of sampling around x^r that are present throughout – never actually sampling it.

4.2.3 Supplementary empirical analysis

An instructive, qualitative way to evaluate each acquisition algorithm is to inspect the final collection of samples (at N), to see visually if things look better for robust variations. Figure 4.9 shows the final samples of one representative MC iteration for EGO, REGO with random α and `StableOPT` for 2d Rosenbrock (4.9) (left panel) and both Bertsimas (4.8) variations (middle and right panels).

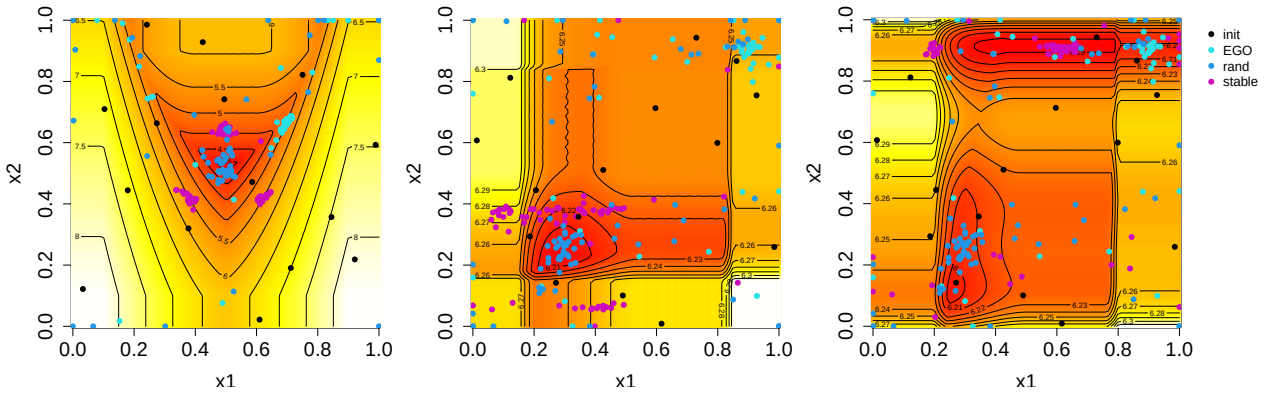


Figure 4.9: Sample acquisitions for EGO, REGO with random α and `StableOPT` for 2d Rosenbrock with $\alpha = 0.1$ (left), and Bertsimas functions with $\alpha = 0.15$ (middle), and $\alpha = (0.2, 0)$ (right).

Consider Rosenbrock first. Here, EGO has most of its acquisitions in a mass around x^* with some acquisitions dispersed throughout the rest of the space. This is exactly what EGO is designed to do: target the global minimum, but still explore other areas. REGO has a

similar amount of space-fillingness, but the target cluster is focused on x^r rather than x^*. On the other hand, `StableOPT` has almost no exploration points. Nearly all of its acquisitions are on the perimeter of a bounding box around x^r. While `StableOPT` does a great job of picking out where x^r is, intuitively you do not need 70+ acquisitions all right next to each other. Some of those acquisitions could better facilitate learning of the surface by exploring elsewhere.

Moving to the Bertsimas panels of the figure, similar behavior may be observed. REGO and EGO have some space-filling points, but mostly target x^r for REGO and x^* for EGO. `StableOPT` again puts almost all of its acquisitions near x^r with relatively little exploration. But the main takeaway from the Bertsimas plots is that, since REGO does not require setting α beforehand, it gives sensible designs for multiple α values (the blue points are the exactly the same in both panels). Looking more closely at the REGO design, observe that all three minima (global and robust with $\alpha = 0.15$ and $\alpha = (0.2, 0)$) have many acquisitions around them. This shows the power of REGO, capturing all levels of α and allowing the user to delay specifying α until after experimental design.

Timings for each method/problem are in Figure 4.10. Comparators "egoph" and "ego" report identical timings because they involve the same EI acquisition function. As you might expect, "unif" and EY have the lowest times because they do less work. For the more competitive methods, REGO is generally a little slower than EGO, but often faster than `StableOPT`. None of these timings are substantial compared to black-box evaluations typically involved in BO enterprises.

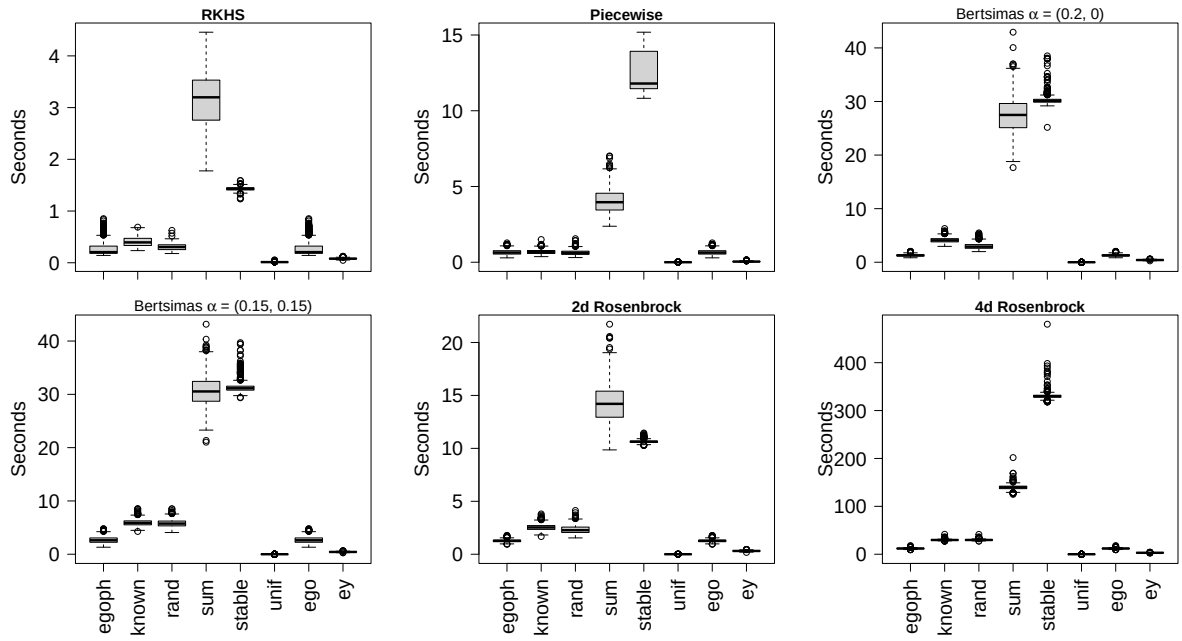


Figure 4.10: Cumulative timings for each method/test function.

4.3 Robot pushing

Here I consider a real world example to measure the effectiveness of REI. The robot pushing simulator [51, https://github.com/zi-w/Max-value-Entropy-Search] has been used previously to test ordinary BO [99, 100] and robust [13] BO methodology. The simulator models a robot hand, with up to 14 tunable parameters, pushing a box with the goal of minimizing distance to a target location. Following Wang and Jegelka [99], I consider varying four of the tunable parameters, as detailed in Table 4.1. This simulator is coded in Python and uses an engine

Parameter	Role	Range
r_x	initial x-location	$[-5, 5]$
r_y	initial y-location	$[-5, 5]$
r_θ	initial angle	$[-\frac{\pi}{2}, \frac{\pi}{2}]$
t_r	pushing strength	$[1, 30]$

Table 4.1: Parameters for the robot pushing simulator.

called `Box2D` [19] to simulate the physics of pushing. Also following Wang and Jegelka [99],

I consider two cases: one with a fixed hand angle, always facing the box, determined to be $r_\theta = \arctan(\frac{r_y}{r_x})$; and the other allowing for all 4 parameters to vary. These create 3d and 4d problems that I call "push3" and "push4", respectively

I consider two further adaptations. First, I de-noised the simulator, so that it is deterministic, which is more in line with the previous examples. Second, rather than look at a single target location, I take the minimum distance to two, geographically distinct target locations under squared and un-squared distances respectively. I do this in order to manifest a version of the problem that would require robust analysis. Having the box pushed the full distance toward either target, minimizing the objective, yields an output of 0 since $0^2 = 0$. However, the minimum around the unsquared target will be shallower because the unsquared surface increases slower when the distance to the target is greater than 1. Robust BO prefers exploring the unsquared minimum while an ordinary, non-robust method would show no preference. Similarly, a BOV performance metric is indifferent to the target locations, while BEAR would favor the unsquared target.

For both "push3" and "push4" variations, I somewhat arbitrarily fixed the target locations at $(-3, 3)$ and $(3, -3)$ with the latter being the target with squared distance outputs. Since the unsquared location is in the top-left quadrant, the optimal robust location involves starting the robot hand in the bottom-right so that it pushes the box up and left. Furthermore, because the robot cannot perfectly control the initial hand location, if it is close to the origin, a minor change in r_x or r_y leads to the hand pushing away from the target location. For "push3" I use $\alpha = 0.1$, $x^r = (5, -5, 25.4)$, meaning the robot hand starts as far in the bottom-right as possible and pushes quite hard. The true setting $g(x^r, 0.1) = 1.37$ is lower than analogously pushing toward the squared target location, $g((-5, 5, 25.4), 0.1) = 1.88$, solely due to squaring as $1.37^2 \approx 1.88$.

I compare each of the methods from Section 4.2.2 in a similar fashion by initializing with

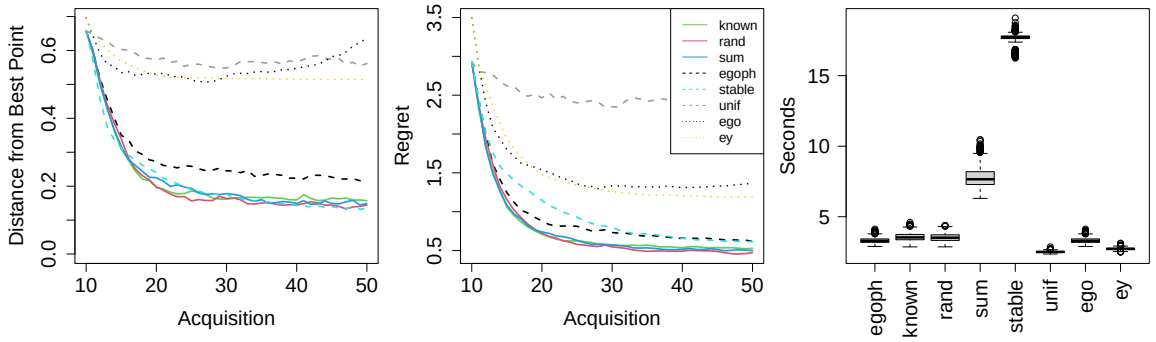


Figure 4.11: Results for "push3" $\alpha = 0.1$: $d(x^*_{\text{b,n}})$ (left), $r(x^*_{\text{b,n}})$ (middle) and cumulative time (right).

an LHS of size $n_0 = 10$ and acquiring forty more points ($N = 50$), repeating for 1,000 MC samples. Figure 4.11 summarizes our results. Observe that all of REI variations outperform the others by minimizing regret faster and converging at a lower value. EI with post hoc adversary and `StableOPT` perform fairly well but suffer from drawbacks similar to those described in Section 4.2.2. They cannot accommodate squared and unsquared target locations differently.

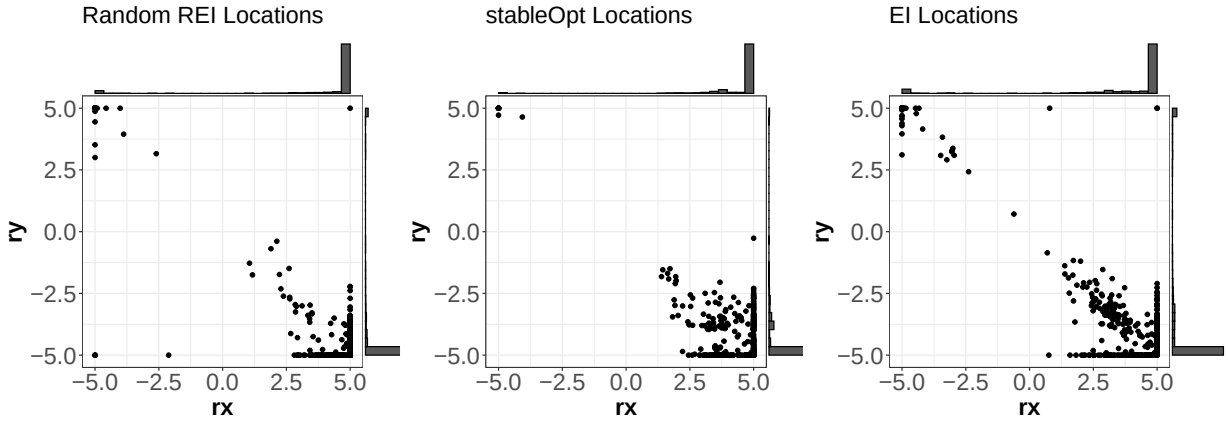


Figure 4.12: Distribution of r_x and r_y from x^*_{bear} for the push3 problem after the final acquisition for "rand" REI using $\alpha = 0.1$ (left), `StableOPT` (middle), and EI (right).

It is worth noting, that while REI performs well, the left panel of Figure 4.11 suggests that none of the methods are getting very close to always finding x^r when $d(x^*_{\text{b,n}}) = 0$. (Distances are not converging to zero.) To dig a little deeper, Figure 4.12 explores r_x and r_y from x^*_{bear} for "rand" REI, `StableOPT` and EI with post hoc adversary. Notice that it is rare for any

of these comparators to push toward the "wrong" target location, i.e., finding $(-5, 5)$ rather than $(5, -5)$ which would result in large $d(x^*_{\text{b,n}})$ and low $r(x^*_{\text{b,n}})$. However, EI is attracted to the peaked minimum more often than either of the other methods. REI has this problem slightly more often than `StableOPT`. However, `StableOPT` and EI do recommend x^*_{bear} in the bottom-right, but not all the way to $(5, -5)$. This happens more often than with REI. Only occassionally does REI miss entirely by finding the global minimum leading to large $d(x^*_{\text{b,n}})$. On the other hand, `StableOPT` identifies the correct area slightly more often, but struggles to pinpoint x^r. I conclude that both REI and `StableOPT` perform well – much better than ordinary EI – and any differences are largely a matter of taste or tailoring to specific use cases, modulo computational considerations (right panel).

For "push4", the location of the robust minimum is the same in that the robot hand starts in the bottom-right, pushing to the top-left. Thus $x^*_r = (5, -5, -0.79, 25.7)$ when using $\alpha = 0.1$. Here $g(x^*_r, 0.1) = 1.64$ compared to $g((-5, 5, -0.79, 25.7), 0.1) = 2.71$ by pushing to the squared target location. I again compared the methods from Section 4.2.2 with 1,000 MC iterations, but this time with an initial LHS of size $n_0 = 20$ and eighty acquisitions ($N = 100$). Results are presented in Figure 4.13. Note that increasing the dimension makes every method do worse, but relative comparisons between methods are similar. Here `StableOPT`'s performance is better, on par with REI variations, again modulo computing time (right panel).

Figure 4.14 demonstrates the added difficulty of the "push4" problem, as indicated by additional spread in the optimal (r_x, r_y) compared to Figure 4.12. This is because a slightly misspecified starting location can be compensated for by adjusting the hand angle. Incorporating that angle introduces a slew of local minima at each (r_x, r_y). For example, if I set $r_x = 3$ rather than 5 as is optimal, and check the regret for "push3" and "push4," I get 1.1 and 0.7, respectively, by changing r_θ to -0.69. I may also adjust $t_r = 24.2$ in both

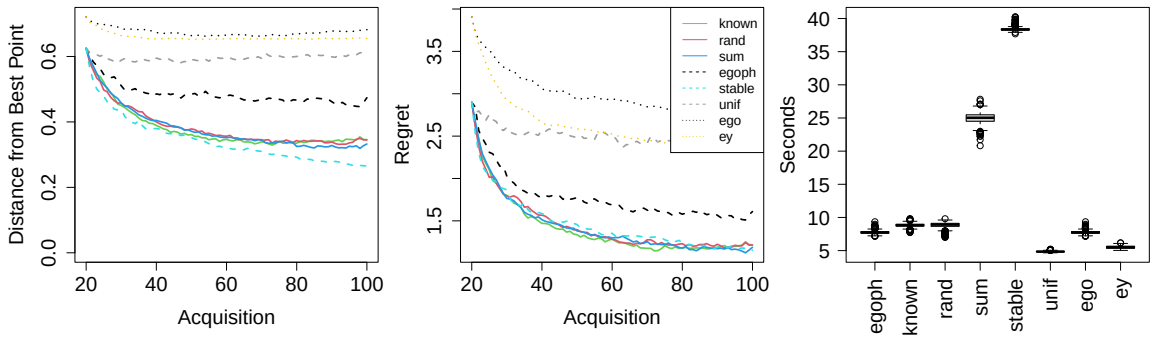


Figure 4.13: Results for "push4" $\alpha = 0.1$: $d(x^*_{\mathrm{b,n}})$ (left), $r(x^*_{\mathrm{b,n}})$ (middle) and cumulative time (right).

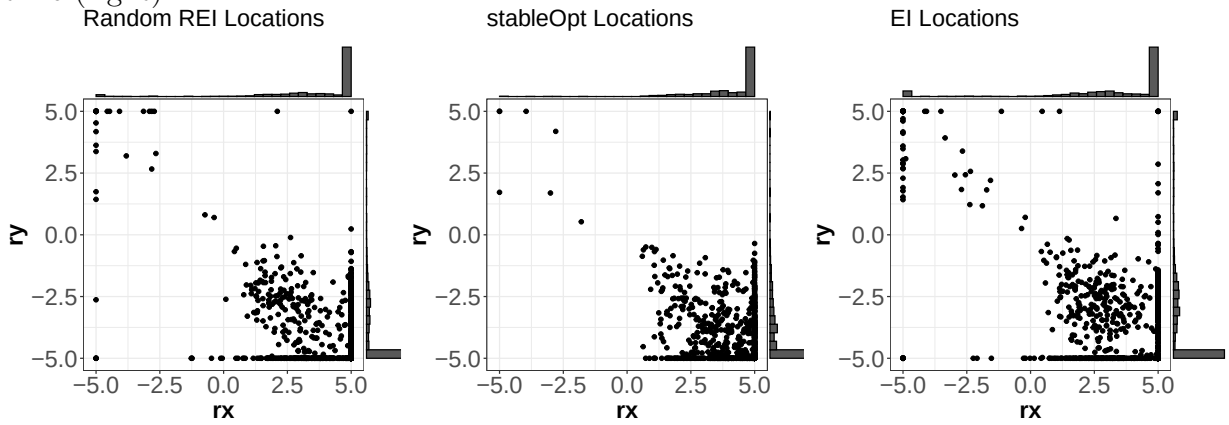


Figure 4.14: Distribution of r_x and r_y from x^*_{bear} for the push4 problem after the final acquisition for random REI using $\alpha = 0.1$ (left), `StableOPT` (middle), and EI (right).

cases to account for the fact the hand is closer to the box. This phenomenon is not unique to the example I chose. Any slight misspecification of r_x and r_y compensated for with a commensurate change in r_θ.

4.4 Discussion

I introduced a robust expected improvement (REI) criteria for Bayesian optimization (BO) that provides good experimental designs for targeting robust minima. REI is doubly robust, in a sense, because it is able to accomplish this feat even in cases where one does not know the proper level of robustness, α, to accommodate. In fact, I have shown that in some cases,

not fixing the α robustness level beforehand leads to faster discovery of the robust minimum. BO methods, e.g., ordinary EI, miss the robust target entirely. However, by blending EI with an (surrogate modeled) adversary, you are able to get the best of both worlds. Even when designs are not targeted to find a robust minimum, the adversarial surrogate can be applied post hoc to similar effect.

Despite this good empirical performance, the idea of robustness undermines one of the key assumptions of GP regression: stationarity. If one optimum is peaked and another is shallow, then that surface is technically nonstationary. When there are multiple such regimes it can be difficult to pin down lengthscale(s) θ, globally in the input space. The underlying REI acquisition scheme may perform even better when equipped with a non-stationary surrogate model such as the treed GP [40] or a deep GP [24, 82]. I surmise that non-GP-based surrogates, e.g., neural networks [87], support vector machines [88] or random forests [25] could be substituted without a fundamental change to the underlying REI criteria.

Throughout I assumed deterministic $f(x)$. Noise can be accommodated by estimating a nugget parameter. The idea of robustness can be extended to protect against output noise [7], in addition to the "input uncertainty" regime I studied here. In that setting, it can be helpful to entertain a heteroskedastic (GP) surrogate [9, 10, 11] when the noise level is changing in the input space.

Chapter 5

Conclusion

This dissertation focused on GPs and their applications to other fields. In Chapter 3, I showed how GPs compare to the industry standard from geosciences for modeling gold deposits as well as how to adapt modern GP methodology to fit the nuance of mining data. Chapter 4 showed how standard BO can be adapted to find robust optima by using the adversary from mathematical programming and showed a physics-based robot simulator is better optimized using robust BO.

Both of those chapters contain their own discussion sections pointing to some limitations and potential for further exploration of each project. I will not repeat that here, but I do think it is worth pointing out that both projects would benefit from a nonstationary extension to my models provided. For ore mining, fault lines often cause abrupt shifts in response surfaces and any robust surface is by definition nonstationary with one regime at the peaked minimum and a different one at the robust minimum. Regular GPs, SVecchia and (ordinary) kriging as I have presented in this dissertation do not account for nonstationarity at all. `laGP` goes in the other direction by making each prediction in isolation of the rest, which can pick up on nonstationary regime changes, but also can potentially provide discontinuous surfaces.

The deep GP [24, 82] handles nonstationarity and may provide even better results than what I showed for both projects.

With two different projects presented, I think the natural idea is to combine them. I see tremedous potential for sucessfully implementing REI for better exploration of potential ore deposits. From my discussions with geologists, for the most part when exploring a new potential mine site, miners drill holes on a grid pattern and if more data are required for better estimation, they drill a hole between each existing hole to make a denser grid. This strategy more than doubles the number of drill holes whenever more data are needed which seems excessive when a few strategically placed holes may provide all the needed information to adequately model the response surface. While the grid strategy will enhance predictions somewhat, this is the perfect setting for BO. Some of those gridded holes are likely in areas with low predictive variance meaning more holes there do not help much in understanding the response surface. While in other areas, many holes may be required to pin down a more complex response surface. With holes costing upwards of $100,000 a piece, BO can optimally layout hole locations in as few acquisitions as necessary and not waste time and money digging unnecessary holes.

Using BO for mining provides challenges that are not present in BO for computer experiments. The borehole structure of the data makes it infeasible to sample any given location as done throughout Chapter 4. Instead, an entire borehole needs to be sampled at a time. Rather than specify the latitute, longitude and depth of the next acquisition, the algorithm would provide an initial latitude, initial longitude and maximum depth to acquire all points in that hole. Further, an angle can be specified instead of drilling straight down to give more control over acquisitions. These practical considerations make BO for mining data more complex than for computer experiments and ripe for innovation to improve on the grid based approach currently applied in the mining industry.

Furthermore, because the goal of exploring for gold is to find the optimal gold mine location, it makes sense to adopt a robust framework, such as the one described in Chapter 4. The exact latitude, longitude and depth containing the most gold is somewhat irrelevant compared to finding the area with the most volume of gold to build a goal mine. Robust BO for mining may require some slight redefinition of robustness instead of the α-window approach if my unknown α setting is not sufficient. While not exactly following in line with the robust framework from Chapter 4 of protecting against a worst case scenario within some α, it is clear that when building a gold mine, one prefers the trough containing a larger volume of gold over a peaked optima with more concentrated gold, but less overall.

Bibliography

[1] P Abrahamsen. A review of Gaussian random fields and correlation functions. Norsk Regnesentral/Norwegian Computing Center Oslo, https://www.nr.no/directdownload/917_Rapport.pdf, 1997.

[2] Christine Anderson-Cook, Lu Lu, William Brenneman, Jeroen Mast, Frederick Faltin, Laura Freeman, William Guthrie, Roger Hoerl, Willis Jensen, L.A. Jones-Farmer, Dennis Leber, Angela Patterson, Marcus Perry, Stefan Steiner, and Nathaniel Stevens. Statistical engineering – part 1: Past and present. *Quality Engineering*, pages 1–20, 08 2022. doi: 10.1080/08982112.2022.2106439.

[3] Jesse Arnold. Virginia tech department of statistics: The first fifty years. *Journal of statistical computation and simulation*, 66:1–17, 04 2000. doi: 10.1080/00949650008812008.

[4] John-Alexander M. Assael, Ziyu Wang, Bobak Shahriari, and Nando de Freitas. Heteroscedastic treed bayesian optimisation. *ArXiv*, 2014. doi: 10.48550/ARXIV.1410.7172. URL https://arxiv.org/abs/1410.7172.

[5] Sudipto Banerjee. *Geostatistical Modeling for Environmental Processes*, pages 81–96. Chapman and Hall/CRC, 2017.

[6] Douglas Bates, Martin Maechler, and Mikael Jagan. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2022. URL https://CRAN.R-project.org/package=Matrix. R package version 1.4-1.

[7] Justin J. Beland and Prasanth B. Nair Nair. Bayesian optimisation under uncertainty. *NIPS BayesOPT 2017 workshop*, 2017.

[8] Dimitris Bertsimas, Omid Nohadani, and Kwong Meng Teo. Robust optimization for unconstrained simulation-based problems. *Operations Research*, 58:161–178, 02 2010. doi: 10.1287/opre.1090.0715.

[9] M Binois and RB Gramacy. `hetGP`*: Heteroskedastic Gaussian process modeling and design under replication*, 2019. URL https://CRAN.R-project.org/package=hetGP. R package version 1.1.1.

[10] M Binois, RB Gramacy, and M Ludkovski. Practical heteroscedastic Gaussian process modeling for large simulation experiments. *Journal of Computational and Graphical Statistics*, 27(4):808–821, 2018. doi: 10.1080/10618600.2018.1458625. URL https://doi.org/10.1080/10618600.2018.1458625.

[11] M Binois, J Huang, RB Gramacy, and M Ludkovski. Replication or exploration? Sequential design for stochastic simulation experiments. *Technometrics*, 27(4):808–821, 2019. doi: 10.1080/00401706.2018.1469433. URL https://doi.org/10.1080/00401706.2018.1469433.

[12] Avrim Blum and Yishay Mansour. Learning, regret minimization, and equilibria. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani, editors, *Algorithmic Game Theory*, chapter 4, pages 79–103. Cambridge University Press, New York, NY, USA, 2007.

[13] Ilija Bogunovic, Jonathan Scarlett, Stefanie Jegelka, and Volkan Cevher. Adversarially robust optimization with gaussian processes. *Advances in Neural Information Processing Systems*, 31:5760–5770, 2018.

[14] GEP Box and NR Draper. *Response Surfaces, Mixtures, and Ridge Analyses*, volume 649. John Wiley & Sons, 2007.

[15] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. The Prindle, Weber and Schmidt Series in Mathematics. PWS-Kent Publishing Company, Boston, fourth edition, 1989.

[16] Richardh Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16, 1995. doi: 10.1137/0916069.

[17] George Casella and Roger Berger. *Statistical Inference*. Duxbury, 6 2001. ISBN 0534243126.

[18] George Casella, Christian P. Robert, and Martin T. Wells. Generalized accept-reject sampling schemes. *Lecture Notes-Monograph Series*, 45:342–347, 2004. ISSN 07492170. URL http://www.jstor.org/stable/4356322.

[19] Erin Catto. Box2d, a 2d physics engine for games, 2011. URL http://box2d.org.

[20] E. W. Cheney and A. A. Goldstein. Newton's method for convex programming and tchebycheff approximation. *Numer. Math.*, 1(1):253–268, 12 1959. ISSN 0029-599X. doi: 10.1007/BF01386389. URL https://doi.org/10.1007/BF01386389.

[21] David A. Cohn. Neural network exploration using optimal experiment design. *Neural Networks*, 9(6):1071–1083, 1996. ISSN 0893-6080. doi: https://doi.org/10.1016/

0893-6080(95)00137-9. URL https://www.sciencedirect.com/science/article/pii/0893608095001379.

[22] Noel Cressie. Fitting variogram models by weighted least squares. *Journal of the International Association for Mathematical Geology*, 17, 07 1985.

[23] Noel Cressie. *Statistics for Spatial Data*. Wiley Series in Probability and Statistics. Wiley, 1993. ISBN 9780471002550.

[24] Andreas Damianou and Neil D. Lawrence. Deep Gaussian processes. In Carlos M. Carvalho and Pradeep Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215, Scottsdale, Arizona, USA, 5 2013. PMLR. URL https://proceedings.mlr.press/v31/damianou13a.html.

[25] Siva Krishna Dasari, Abbas Cheddad, and Pette Andersson. Random forest surrogate models to support design space exploration in aerospace use-case. In John MacIntyre, Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 532–544, Cham, 2019. Springer International Publishing.

[26] Abhirup Datta, Sudipto Banerjee, Andrew O. Finley, and Alan E. Gelfand. Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016. doi: 10.1080/01621459.2015.1044091. URL https://doi.org/10.1080/01621459.2015.1044091. PMID: 29720777.

[27] Henri J. De Wijs. Statistics of ore distribution. *Journal of the Royal Netherlands Geologie en Mijnbouw Society*, 13:365–375, 1953.

[28] A. Dean, M. Morris, J. Stufken, and D. Bingham. *Handbook of Design and Analysis of Experiments*. Chapman & Hall/CRC Handbooks of Modern Statistical Methods. CRC Press, 2015. ISBN 9781466504349.

[29] C. V. Deutsch and A. G. Journel. *GSLIB: Geostatistical Software Library and User's Guide*. Oxford University Press, New York, NY, 1997.

[30] L. C. W. Dixon and G. P. Szego. *The global optimization problem: an introduction*. North-Holland Pub. Co, Amsterdam, 1979.

[31] Dirk Eddelbuettel and Romain François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. doi: 10.18637/jss.v040.i08. URL https://www.jstatsoft.org/v40/i08/.

[32] Andrew Gelman, Jessica Hwang, and Aki Vehtari. Understanding predictive information criteria for bayesian models. *Statistics and computing*, 24(6):997–1016, 2014.

[33] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

[34] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):107–114, 1952. ISSN 00359246. URL http://www.jstor.org/stable/2984087.

[35] RB Gramacy and H Lian. Gaussian process single-index models as emulators for computer experiments. *Technometrics*, 54(1):30–41, 2012.

[36] RB Gramacy and NG Polson. Particle learning of Gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics*, 20 (1):102–118, 2011.

[37] Robert B. Gramacy. lagp : Large-scale spatial modeling via local approximate gaussian processes in r. *Journal of Statistical Software*, 72, 08 2016. doi: 10.18637/jss.v072.i01.

[38] Robert B. Gramacy. *Surrogates: Gaussian Process Modeling, Design and Optimization for the Applied Sciences*. Chapman Hall/CRC, Boca Raton, Florida, 2020.

[39] Robert B. Gramacy and Daniel W. Apley. Local gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24 (2):561–578, 2015. URL http://www.jstor.org/stable/24737281.

[40] Robert B. Gramacy and Herbert K. H. Lee. Bayesian treed gaussian process models with an application to computer modeling, 2007. URL https://arxiv.org/abs/0710.4536.

[41] Robert B. Gramacy, Annie Sauer, and Nathan Wycoff. Triangulation candidates for bayesian optimization, 2022.

[42] Joseph Guinness. Permutation and grouping methods for sharpening gaussian process approximations. *Technometrics*, 60(4):415–429, 2018.

[43] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[44] Matthew J. Heaton, Abhirup Datta, Andrew O. Finley, Reinhard Furrer, Joseph Guinness, Rajarshi Guhaniyogi, Florian Gerber, Robert B. Gramacy, Dorit M. Hammerling, Matthias Katzfuss, Finn Lindgren, Douglas W. Nychka, Furong Sun, and Andrew Zammit-Mangion. A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological, and Environmental Statistics*, 24:398 – 425, 2019.

[45] L. Hong and Barry Nelson. Discrete optimization via simulation using compass. *Operations Research*, 54:115–129, 02 2006. doi: 10.1287/opre.1050.0237.

[46] Ling Huang, A. Joseph, B. Nelson, Benjamin I. P. Rubinstein, and J. Tygar. Adversarial machine learning. In Yan Chen, editor, *AISec '11*, 2011.

[47] M.E. Johnson, L.M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26(2):131–148, 1990. ISSN 0378-3758. doi: https://doi.org/10.1016/0378-3758(90)90122-B. URL https://www.sciencedirect.com/science/article/pii/037837589090122B.

[48] Donald Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 12 1998.

[49] DR Jones, M Schonlau, and WJ Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[50] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[51] Leslie Kaelbling and Tomas Lozano-Perez. Learning composable models of parameterized skills. pages 886–893, 05 2017. doi: 10.1109/ICRA.2017.7989109.

[52] Damirand Kalpić and Nikica Hlupić. Multivariate normal distributions. In Miodrag Lovric, editor, *International Encyclopedia of Statistical Science*, pages 907–910. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_623. URL https://doi.org/10.1007/978-3-642-04898-2_623.

[53] Matthias Katzfuss and Joseph Guinness. A general framework for vecchia approximations of gaussian processes. *Statistical Science*, 36(1):124–141, 2021.

[54] Matthias Katzfuss, Joseph Guinness, Wenlong Gong, and Daniel Zilber. Vecchia approximations of gaussian-process predictions. *Journal of Agricultural, Biological and Environmental Statistics*, 25(3):383–414, 6 2020. doi: 10.1007/s13253-020-00401-7. URL https://doi.org/10.1007%2Fs13253-020-00401-7.

[55] Matthias Katzfuss, Marcin Jurek, Daniel Zilber, and Wenlong Gong. *GPvecchia: Scalable Gaussian-Process Approximations*, 2020. URL https://CRAN.R-project.org/package=GPvecchia. R package version 0.1.3.

[56] Matthias Katzfuss, Joseph Guinness, and Earl Lawrence. Scaled vecchia approximation for fast computer-model emulation, 2021.

[57] Danie G. Krige. Some basic considerations in the application of geostatistics to the valuation of ore in south african gold mines. *Journal of the Southern African Institute of Mining and Metallurgy*, 76(9):383–391, 1976. doi: 10.10520/AJA0038223X_601.

[58] Yifang Li and Sujit K. Ghosh. Efficient sampling methods for truncated multivariate normal and student-t distributions subject to linear inequality constraints. *Journal of Statistical Theory and Practice*, 9(4):712–732, 2015. doi: 10.1080/15598608.2014.996690. URL https://doi.org/10.1080/15598608.2014.996690.

[59] Roderick J. A. Little and Donald B. Rubin. *Statistical Analysis with Missing Data, Second Edition*. Wiley-Interscience, 2002.

[60] Kailong Liu, Yi Li, Xiaosong Hu, Mattin Lucu, and Widanalage Dhammika Widanage. Gaussian process regression with automatic relevance determination kernel for calendar aging prediction of lithium-ion batteries. *IEEE Transactions on Industrial Informatics*, 16(6):3767–3777, 2020. doi: 10.1109/TII.2019.2941747.

[61] Yufan Liu and Ying Hung. Latin hypercube design-based block bootstrap for computer experiment modeling. Technical report, Rutgers, 2015.

[62] DJC MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

[63] Ruben Martinez-Cantin, Nando Freitas, Eric Brochu, Jose Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Auton. Robots*, 27:93–103, 08 2009. doi: 10.1007/s10514-009-9130-2.

[64] Julien Marzat, Eric Walter, and Hélène Piet-Lahanier. A new expected-improvement algorithm for continuous minimax optimization. *J. Glob. Optim.*, 64(4):785–802, 2016. doi: 10.1007/s10898-015-0344-x. URL https://doi.org/10.1007/s10898-015-0344-x.

[65] Georges Matheron. Principles of geostatistics. *Economic Geology*, 58(8):1246–1266, 12 1963. ISSN 0361-0128. doi: 10.2113/gsecongeo.58.8.1246. URL https://doi.org/10.2113/gsecongeo.58.8.1246.

[66] Georges Matheron. *The Theory of Regionalized Variables and Its Applications*. Les Cahiers du Centre de Morphologie Mathematique, Fontaineblue, Paris, 1971.

[67] Matt Menickelly and Stefan M. Wild. Derivative-free robust optimization by outer approximations. *Mathematical Programming*, 179(1–2):157–193, 2020.

[68] J Mockus, V Tiesis, and A Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.

[69] David M. Morens. Cholera, Chloroform, and the Science of Medicine: A Life of John

Snow. *American Journal of Epidemiology*, 160(6):605–606, 09 2004. ISSN 0002-9262. doi: 10.1093/aje/kwh246.

[70] RH Myers, DC Montgomery, and CM Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. John Wiley & Sons, 2016.

[71] José Nogueira, Ruben Martinez-Cantin, Alexandre Bernardino, and Lorenzo Jamone. Unscented bayesian optimization for safe robot grasping. *CoRR*, abs/1603.02038, 2016. URL http://arxiv.org/abs/1603.02038.

[72] Rafael Oliveira, Lionel Ott, and Fabio Ramos. Bayesian optimisation under uncertain inputs. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1177–1184. PMLR, 04 2019. URL https://proceedings.mlr.press/v89/oliveira19a.html.

[73] M. R. Osborne. Fisher's method of scoring. *Int. Stat. Rev*, 60:271–286, 1992.

[74] Edzer Pebesma. The meuse data set: a tutorial for the gstat r package, 01 2009.

[75] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/.

[76] S. E. Randall, III Halsted, D. M., and D. L. Taylor. Optimum Vibration Absorbers for Linear Damped Systems. *Journal of Mechanical Design*, 103(4):908–913, 10 1981. doi: 10.1115/1.3255005.

[77] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[78] Douglas Reynolds. Gaussian mixture models. In Stan Z. Li and Anil Jain, editors, *Encyclopedia of Biometrics*, pages 659–663. Springer US, Boston, MA, 2009. ISBN 978-0-387-73003-5. doi: 10.1007/978-0-387-73003-5_196. URL https://doi.org/10.1007/978-0-387-73003-5_196.

[79] D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley, New York, NY, 1987.

[80] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and Analysis of Computer Experiments. *Statistical Science*, 4(4):409 – 423, 1989. doi: 10.1214/ss/1177012413. URL https://doi.org/10.1214/ss/1177012413.

[81] Annie Sauer, Robert B Gramacy, and David Higdon. Active learning for deep gaussian process surrogates. *Technometrics*, pages 1–39, 2021.

[82] Annie Sauer, Andrew Cooper, and Robert B. Gramacy. Vecchia-approximated deep gaussian processes for computer experiments, 2022. URL https://arxiv.org/abs/2204.02904.

[83] J. L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman Hall/CRC, New York, New York, 1997.

[84] Warren Scott, Peter Frazier, and Warren Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21:996–1026, 07 2011. doi: 10.1137/100801275.

[85] S Seo, M Wallat, T Graepel, and K Obermayer. *Gaussian process regression: active data selection and test point rejection*, pages 27–34. Institute of Electrical and Electronics Engineers, New York, NY, 2000.

[86] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2016. doi: 10.1109/JPROC.2015.2494218.

[87] M. Shahriari, D. Pardo, B. Moser, and F. Sobieczky. A deep neural network as surrogate model for forward simulation of borehole resistivity measurements. *Procedia Manufacturing*, 42:235–238, 2020. ISSN 2351-9789. doi: https://doi.org/10.1016/j.promfg.2020.02.075. URL https://www.sciencedirect.com/science/article/pii/S2351978920306399. International Conference on Industry 4.0 and Smart Manufacturing (ISM 2019).

[88] Maolin Shi, Liye Lv, Wei Sun, and Xueguan Song. A multi-fidelity surrogate model based on support vector regression. *Struct. Multidiscip. Optim.*, 61(6):2363–2375, jun 2020. ISSN 1615-147X. doi: 10.1007/s00158-020-02522-6. URL https://doi.org/10.1007/s00158-020-02522-6.

[89] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machinelearning algorithms. *Advances in Neural information Processing Systems (NIPS)*, pages 2951–2959, 2012.

[90] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

[91] Michael L Stein, Zhiyi Chi, and Leah J Welty. Approximating likelihoods for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(2):275–296, 2004.

[92] Jonathan R Stroud, Michael L Stein, and Shaun Lysen. Bayesian and maximum likelihood estimation for gaussian processes on an incomplete lattice. *Journal of computational and Graphical Statistics*, 26(1):108–120, 2017.

[93] F Sun, RB Gramacy, B Haaland, E Lawrence, and A Walker. Emulating satellite drag from large simulation experiments. *SIAM/ASA Journal on Uncertainty Quantification*, 7(2):720–759, 2019. preprint arXiv:1712.00182.

[94] Matthew A. Taddy, Herbert K. H. Lee, Genetha A. Gray, and Joshua D. Griffin. Bayesian guided pattern search for robust local optimization. *Technometrics*, 51(4): 389–401, 2009. doi: 10.1198/TECH.2009.08007.

[95] Hastagiri P. Vanchinathan, Isidor Nikolic, Fabio De Bona, and Andreas Krause. Explore-exploit in top-n recommender systems via gaussian processes. In Alfred Kobsa and Michelle Zhou, editors, *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 225–232. Association for Computing Machinery, 2014. ISBN 9781450326681. doi: 10.1145/2645710.2645733. URL https://doi.org/10.1145/2645710.2645733.

[96] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, New York, NY, 2013.

[97] Aldo V. Vecchia. Estimation and model identification for continuous spatial processes. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):297–312, 1988. ISSN 00359246. URL http://www.jstor.org/stable/2345768.

[98] Hans Wackernagel. *Multivariate Geostatistics: An introduction with applications*. Springer, Berlin, 2003.

[99] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3627–3635. JMLR.org, 2017.

[100] Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional Bayesian optimization via structural kernel learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3656–3664. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/wang17h.html.

[101] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963. PMLR, 2015.

[102] H Wendland. *Scattered Data Approximation*. Cambridge University Press, Cambridge, England, 2004.

[103] Stefan Wilhelm and Manjunath B G. *tmvtnorm: Truncated Multivariate Normal and Student t Distribution*, 2022. URL https://CRAN.R-project.org/package=tmvtnorm. R package version 1.5.

[104] Luhuan Wu, Geoff Pleiss, and John Cunningham. Variational nearest neighbor gaussian processes. *arXiv preprint arXiv:2202.01694*, 2022.

[105] Nathan Wycoff, Mickaël Binois, and Robert B Gramacy. Sensitivity prewarping for local surrogate modeling. *arXiv preprint arXiv:2101.06296*, 2021.

[106] B Zhang, DA Cole, and RB Gramacy. Distance-distributed design for Gaussian process surrogates. *To appear in Technometrics*, 2020. Preprint on arXiv:1812.02794.