Probabilistic Machine Learning for Online Decision-Making

by

Samuel Stanton

a dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
Center for Data Science
New York University

September, 2022

Andrew Gordon Wilson

Dedication

To my mother, who taught me to seek truth.

Acknowledgments

First and foremost I would like to thank my advisor, Dr. Andrew Gordon Wilson, for supporting and guiding the work that eventually became this thesis. Some of my fondest memories of grad school will be of our long discussions together in his office, which often greatly exceeded their allotted time in his busy schedule. Along with Andrew, I wish to thank the rest of my thesis committee, Benjamin Peherstorfer, Kyunghyun Cho, Lerrel Pinto, and Rob Fergus for their feedback. I also want to thank my labmates, Alex Wang, Greg Benton, Ian Delbridge, Marc Finzi, Nate Gruver, Pavel Izmailov, Polina Kirichenko, Sanyam Kapoor, and Wesley Maddox. We share the kind of camaraderie that is only forged by enduring hardship together, in our case the unforgiving pace of publication cycles, the vagaries of peer review, and a rather abrupt move from Cornell to NYU. Wesley Maddox deserves special recognition as my most closest collaborator, who contributed to every chapter of this thesis in some form, and as my loyal companion in all manner of pleasant diversions. I am also indebted to many mentors for their insight and assistance, including Aaron Klein, Alex Alemi, Brandon Amos, Cedric Archambeau, Hugh Gauch, Jake Gardner, Jonas Mueller, Mark van der Wilk, Rasool Fakoor, Renaud Chicoisne, and Shelley Rohde Pool. I also want to thank my parents — my father, Gerald, for encouraging me to pursue math and computer science from a young age, and especially my mother, Heather, who homeschooled me from kindergarten through high school. If I have had any academic success, much of the credit rightly belongs to her. Much thanks is also due to my long-suffering partner, Jen, for her love and patience through many difficult periods during which I would disappear into my work for weeks at a time. Last but not least there are countless unnamed friends and colleagues who also contributed to my intellectual development in smaller yet significant ways, and to these I wish to convey my heartfelt gratitude.

Preface

The story of this thesis in many ways mirrors the story of how I came to study machine learning itself. In my undergraduate studies in applied mathematics, I encountered the field of operations research (OR), which can broadly be defined as a collection of mathematical and computational methods for the study and analysis of problems involving decision-making. I found the topic very interesting, so in 2017 I applied and was accepted to Cornell's OR Ph.D. program, only a year after the publication of AlphaGo in Nature (Silver et al., 2017). It seemed clear that the sort of decision problems that were being solved with machine learning were of a different sort than canonical OR problems like bin-packing or facility-location. At this point I began working with my advisor, Dr. Andrew Gordon Wilson (with whom I later moved to NYU), on the topic of Bayesian machine learning, usually with a distinct lean towards decision-making applications. Over the course of many projects investigating how decision-making with machine learning could work, I gradually internalized normative principles for decision-making in general. In this thesis I have chosen to highlight some of these principles, and how to satisfy them with specific machine learning methods from a collection of papers published during the last three years. Sadly there is a great deal of material from various projects that has not been included due to space constraints and incompatibility with the theme of the thesis, which nevertheless significantly impacted my thinking. In the end my graduate studies conclude as they started, in the study of guiding principles and specific techniques for good decision-making.

Abstract

The study of machine learning is concerned with the development and analysis of general-purpose programs which receive data as input, extract useful patterns and statistics, and automatically modify their output accordingly. The purpose of many programs is to make consistent, justifiable decisions in a timely manner based on as much relevant information as possible. In this work we will focus particularly on programs that operate online, meaning programs whose output is transformed into more information and fed back in as input to create a sequence of conditional decisions. Online programs are particularly useful for applications involving a great deal of complexity or uncertainty, since they can break down difficult planning problems into easier steps, and can collect additional information as needed. We will discuss techniques to efficiently update specific statistical models on infinite streams of data, coherent data collection strategies to optimize program outputs for arbitrary objectives, and means for turning imperfect models into reliable, trustworthy decisions based on provably valid predictions. In addition to fundamental contributions to the body of machine learning methods, we will also present exemplar applications including public health surveillance, control of mechanical systems, and experimental design for scientific discovery.

Contents

List of Figures

List of Tables

List of Appendices

Every thought emits a roll of the dice.

Stéphane Mallarmé

0 Introduction

The maturation of two technologies, the Internet and high-performance parallel computing, has dramatically increased our ability to generate, distribute, and process information. Flexible machine learning models trained on large historical databases have gradually augmented or replaced hand-crafted algorithms for computer vision (Chai et al., 2021), natural language processing

(Sun et al., 2022), and recommendation systems (Khan et al., 2021). These past successes have encouraged efforts to extend machine learning to every conceivable application, including self-driving vehicles (Gupta et al., 2021), supply chain management (Pournader et al., 2021), and drug design (Muller et al., 2022; Kim et al., 2021). Many of these applications are *online*, meaning operational data is continuously collected and processed as it arrives, which presents two fundamental challenges. First, learned system components are not fitted to static, offline datasets instead they are incrementally fitted to dynamic streams of data, requiring adjustments to the basic learning algorithms. Second, in many applications machine learning systems are not passive recipients of information, but agents whose actions (i.e. decisions) directly impact future observations, introducing dilemmas such as the explore-exploit tradeoff.

The structure of this thesis is as follows: in Chapter 1 we review fundamental concepts of probability and statistics, Bayesian machine learning (particularly Gaussian processes), and basic online decision-making with Bayesian optimization. Chapter 2 is devoted to the theme of making *non-myopic* decisions based on newly available information. To that end we discuss techniques for efficiently conditioning different sparse Gaussian process approximations on infinite streams of data or self-generated "fantasy samples" with a constant compute and memory footprint, with supplementary material in Appendix A and B. The material for these sections was originally published in Stanton et al. (2021) and Maddox et al. (2021c). The theme of Chapter 3 is making *coherent* decisions, that is decisions based on coherent beliefs and preferences, particularly in the context of biological sequence design, with supplementary material in Appendix C. The material for these sections was originally published in Stanton et al. (2022). Next the theme of Chapter 4 is making decisions with reliably predictable outcomes, specifically decisions for which our model can construct *valid* prediction sets. One notable feature of this chapter is our exceptionally mild set of assumptions, in particular we do not assume that our model is basically correct, instead we focus on accounting for uncertainty *relating to the hypothesis class itself*, with supplementary material in Appendix D. At the time of writing the material for these sections is under review for publication. Finally in Chapter 5 we end with some specific conclusions for the reader to take away, along with interesting directions for future research.

Probability [is] a concept that vexes philosophers,
a concept with two faces.

Ian Hacking

1
Preliminaries

In this chapter we introduce the basic concepts we will need for the following chapters. Those less philosophically inclined may wish to skip to Chapter 1.2. The exposition is brief by necessity, but we cite essential references and high-quality tutorials for further reading. Some notation may differ from common convention for holistic consistency between chapters or to avoid overloading specific symbols too frequently.

1.1 Statistical Foundations

In the beginning, there was a dataset, $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=0}^{n-1}$. We consider each datum $(\mathbf{x}_i, \mathbf{y}_i)$ to be an objective observation of an instance of some event. In general the event is signified by the tuple of variables (X, Y). The word *objective* is fraught with philosophical difficulties, however for our purposes, an objective event need only be an entity of interest to a specific group of observers, who have previously agreed upon its ontological nature (i.e. what the entity is and is not), and the empirical means (i.e. senses and tools shared by the group) by which to observe the entity if or when it occurs. Once an observation has been made, within this group of observers there is no debate as to what is explicitly signified by the observation, and what is signified does not depend on any characteristics of the observer (other than their membership of the group) nor does it depend on the time the observation is perceived. This definition accommodates traditional Western notions of truth such as correspondence theory, as well as more skeptical perspectives such as those articulated in Kuhn (1970).

Statistical inference is concerned with two distinct concepts, 1) the objective frequencies with which potential observations occur over some number of repetitions, and 2) our subjective preferences between different types of potential observations. Given internally coherent preferences between observations (i.e. preferences that satisfy properties like transitivity) and some fairly uncontroversial axioms, we can logically deduce the need for internally coherent subjective beliefs regarding the plausibility of potential observations in order to act in a way that is consistent with our preferences (Savage, 1972). By convention, we act upon X, which we call the *state*, and observe the effect on the *outcome* Y.

1.2 Bayesian Inference

Statistical inference begins by postulating a likelihood $p(D|f)$ that defines a relation between real observations and the idealized abstractions and assumptions we use to explain how those observations could have come about. Explanations coalesce into hypotheses $f \in \mathcal{F}$, where $\mathcal{F}$ represents the space of all distinct possibilities we are able to express with this idealized structure. Typically we think of hypotheses as functions $f : \mathcal{X} \to \mathcal{Y}$, which is a way of encoding the basic assumption that underlying the apparent randomness of the world there are invariant relations between state and outcome that make inductive inference possible.

Bayesian inference begins by specifying a prior $p(f)$ which formally represents the degree to which a hypothesis f is plausible before seeing any real data. Once we have D, we apply Bayes rule to obtain the Bayes posterior,

$$p(f|D) = \frac{p(D|f)p(f)}{\int p(D|f)p(f)df}, \quad p(\mathbf{y}|\mathbf{x}, D) = \int p(\mathbf{y}|f(\mathbf{x}))p(f(\mathbf{x})|D)df, \tag{1.1}$$

where $p(f|D)$ and $p(\mathbf{y}|\mathbf{x}, D)$ respectively represent the subjective plausibility (i.e. credibility) of f overall, and of a potential outcome $\mathbf{y}$ for a specific input $\mathbf{x}$, given D.[*] By subjective plausibility, we mean the degree of coherence with D, if our model assumptions do in fact hold (i.e. at least one element of our hypothesis class actually is the process which generated our data).

A Bayesian β-credible set can be defined over both abstract hypotheses and real outcomes, where $\beta \in (0, 1]$ is the level of subjective confidence that the credible set contains all relevant possibilities. For example a prediction set $Y_{\text{cred}} \subset \mathcal{Y}$ is credible at the level β if

$$\beta = \int_{\mathbf{y} \in Y_{\text{cred}}} p(\mathbf{y}|\mathbf{x}, D) d\mathbf{y}.$$

A change to the model assumptions can (and often does) significantly affect which possibilities are considered relevant, before and after seeing D.

1.3 Bayesian Machine Learning and Gaussian Processes

Though Bayesian inference is fairly simple conceptually, in practice computing Bayesian posteriors as defined in Eq. 1.1 is extremely difficult. In fact for many commonly used machine learning models full Bayesian posteriors cannot be exactly computed at all, and must be approximated instead. Notable exceptions are linear models and Gaussian processes (GPs), which have closed-form posteriors for certain choices of likelihoods. See Bishop & Nasrabadi (2006) for a thorough introduction to Bayesian linear regression, and Rasmussen &

[*]Parametric models typically define the prior and posterior over $\mathcal{F}$ implicitly via the model architecture and a prior and posterior over the model weights θ.

Williams (2008) for a complete treatment of GPs.

Because we will make heavy use of GP regression models throughout this thesis, we will briefly introduce them here. In this setting $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}^m$. There are several ways to derive a GP, for our purposes it is most useful to see them as a prior over functions defined by a choice of mean function $\mu : \mathcal{X} \to \mathbb{R}^m$ and kernel function $\kappa_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ with hyperparameters θ. In particular, given a finite collection of inputs $X_{\text{train}} = [\mathbf{x}_0 \ \cdots \ \mathbf{x}_{n-1}]^\top$, the outputs $\mathbf{a} := [f(\mathbf{x}_0) \ \cdots \ f(\mathbf{x}_{n-1})]^\top$ have the prior distribution

$$p(\mathbf{a}|\theta) = \mathcal{N}(\mu_{\mathbf{a}}, K_{\mathbf{aa}}),$$

where $(\mu_{\mathbf{a}})_i = \mu(\mathbf{x}_i)$ and $(K_{\mathbf{aa}})_{ij} = \kappa_\theta(\mathbf{x}_i, \mathbf{x}_j)$. In shorthand a GP prior is simply written $p(f) = \mathcal{GP}(\mu, \kappa_\theta)$. For clarity of notation and without loss of generality, we typically take $\mu(\mathbf{x}) = 0$ for all $\mathbf{x}$.

Since $p(\mathbf{a}|\theta)$ is Gaussian, it has a natural conjugate Gaussian likelihood $p(\mathbf{y}|f(\mathbf{x})) = \mathcal{N}(f(\mathbf{x}), \Sigma)$. This likelihood is often written as the generative model $\mathbf{y} = f(\mathbf{x}) + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \Sigma)$. Because the prior and likelihood are conjugate, the Bayes posterior over outputs $\mathbf{b} := [f(\mathbf{x}'_0) \ \cdots \ f(\mathbf{x}'_{q-1})]$ at a new collection of inputs $X_{\text{test}} = [\mathbf{x}'_0 \ \cdots \ \mathbf{x}'_{q-1}]^\top$ is again Gaussian, specifically

$$p(\mathbf{b}|D) = \mathcal{N}(\mathbf{m}_{\mathbf{b}|D}, S_{\mathbf{b}|D}), \text{where}$$

$$\mathbf{m}_{\mathbf{b}|D} = K_{\mathbf{ba}}(K_{\mathbf{aa}} + \Sigma)^{-1}\mathbf{y}_{\mathbf{a}}, \tag{1.2}$$

$$S_{\mathbf{b}|D} = K_{\mathbf{bb}} - K_{\mathbf{ba}}(K_{\mathbf{aa}} + \Sigma)^{-1}K_{\mathbf{ab}}. \tag{1.3}$$

To fit a GP regression model to D, we optimize the kernel hyperparameters θ to maximize the log-marginal likelihood of D,

$$\begin{aligned}\log p(D|\theta) &= \log \int p(\mathbf{y_a}|\mathbf{a})p(\mathbf{a}|\theta)d\mathbf{a} \\ &= -\frac{1}{2}\left(\mathbf{y_a}^\top (K_{\mathbf{aa}} + \Sigma)^{-1}\mathbf{y_a} + \log|K_{\mathbf{aa}} + \Sigma| + n\log 2\pi\right)\end{aligned} \tag{1.4}$$

Although GPs are kernel-based models, there is a range of well-known methods to scale them to large offline datasets, notably inducing point methods like stochastic variational GPs (SVGPs) which admit the use of stochastic optimizers (Hensman et al., 2013a). In Chapter 2 we discuss different approaches to scaling GPs to online datastreams in great detail.

The inductive biases of a GP are primarily determined by the choice of kernel. Most commonly used GP kernels (e.g. RBF or Matérn) rely on ℓ_2 distance between inputs to determine the prior covariance between outputs. When the inputs are low-dimensional (e.g. $d = 10$) such kernels work well, but in high dimensions the ℓ_2 norm is often a poor choice of distance metric (Srinivas et al., 2010; Wang et al., 2016). This limitation has motivated the development of *deep kernel learning* (DKL), which learns a low-dimensional continuous embedding via an encoder such as a convolutional neural network (CNN) (Wilson et al., 2016a).

1.4 Bayesian Optimization

Bayesian optimization (BayesOpt) is an archtypical example of Bayesian decision theory applied to decision problems of the form

$$\max_{\mathbf{x} \in \mathcal{X}} (f_0(\mathbf{x}) \ \cdots \ f_{m-1}(\mathbf{x})).$$

BayesOpt has recently demonstrated great promise in the control of complex systems and the design of sensitive experiments, with applications ranging from hyperparameter tuning of large models, to control of lasers, to antibody design (Chen et al., 2018; Duris et al., 2020; Moss et al., 2020; Maddox et al., 2021a; Maus et al., 2022).

Conceptually BayesOpt is a simple combination of a utility function $u(f(\mathbf{x}), D)$ and a Bayesian machine learning model (a *surrogate*) which computes $p(f(\mathbf{x})|D)$. GP regression models are often preferred as surrogates because their closed-form posteriors are convenient to work with and because they tend to generalize well in data-scarce regimes when d is small. See Brochu et al. (2010) and Frazier (2018) for a more detailed introduction.

BayesOpt alternates between *inference* and *selection*, comparing the expected utility $\mathbb{E}_{\mathbf{b}|D}[u(\mathbf{b}, D)]$ of potential queries (i.e. "test" points) to select the next batch of observations, which are added to D to update $p(\mathbf{b}|D)$, completing one iteration of a repeating loop (Algorithm 1). BayesOpt literature often refers to the expected utility of a query point as its *acquisition value*, and a function mapping a query to its acquisition value as an acquisition function

Algorithm 1: The BayesOpt outer loop

```
Inputs: hypothesis space $\mathcal{F}$, acquisition $a$, dataset $\mathcal{D}_0$.
for $i = 0, \dots, i_{\max} - 1$ do
    Fit $\hat{f} \in \mathcal{F}$ to $\mathcal{D}_i$.
    $\mathbf{x}_i^* = \min_{\mathbf{x} \in \mathcal{X}} a(\mathbf{x}, \hat{f})$.  $\leftarrow$ the inner loop
    Observe $\mathbf{y}_i \sim p(\cdot|\mathbf{x}_i^*)$.
    $\mathcal{D}_{i+1} = \mathcal{D}_i \cup (\mathbf{x}_i^*, \mathbf{y}_i)$.
    $\mathcal{P}_{i+1} = \text{nondominated}(\mathcal{D}_{i+1})$.
end
return $\mathcal{P}_{i_{\max}}$
```

$a : \mathcal{X} \to \mathbb{R}$,

$$a(\mathbf{x}) = \int u(f(\mathbf{x}), D) p(f(\mathbf{x})|D) df, \tag{1.5}$$

There are two important considerations when it comes to choosing a utility function for BayesOpt. First the utility function must account for the specific needs of the task, such as batched queries, multiple objectives, multiple measurement options with varying cost and quality, or multi-step planning horizons. Second the utility function should be structured in a way such that the resulting acquisition function can be efficiently optimized. The two considerations are often in tension.

1.4.1 Example acquisition functions

Taking $u(f(\mathbf{x}), D) = [f(\mathbf{x}) - \max_{\mathbf{y}_i \in D} \mathbf{y}_i]_+$, where $[g(\cdot)]_+ = \max(g(\cdot), 0)$, yields the expected improvement (EI) acquisition function (Jones et al., 1998). Other acquisition functions look ahead into the future to see how the model will change if we query a specific point, a procedure known as "fantasization" (Hennig & Schuler, 2012; Wu & Frazier, 2016; Jiang et al., 2020a). Fantasization is done by drawing samples from the current surrogate posterior at some set of points and conditioning the surrogate on those samples. For example, the batch knowledge gradient (qKG, Wu & Frazier, 2016; Balandat et al., 2020a) is given by

$$u(f(\mathbf{x}), D) := \max_{\mathbf{x}' \in \mathcal{X}} \mathop{\mathbb{E}}_{f(\mathbf{x}') \sim p(\cdot \mid D_{+\mathbf{x}})} f(\mathbf{x}') - \max_{\mathbf{x}' \in \mathcal{X}} \mathop{\mathbb{E}}_{f(\mathbf{x}') \sim p(\cdot \mid D)} f(\mathbf{x}'), \tag{1.6}$$

where $D_{+\mathbf{x}} := D \cup \{(\mathbf{x}, f(\mathbf{x})\}$. Computing the first term requires conditioning the surrogate model on posterior samples at $\mathbf{x}$, before optimizing through predictions of the conditioned surrogate model.

If we can find a maximizer $\mathbf{x}^*$ of a, then by definition $\mathbf{x}^*$ will be Bayes-optimal w.r.t. u. Bayes-optimality simply indicates whether a decision is coherent with our posterior beliefs. Bayesian inference (and thus Bayes-optimality) is not directly concerned with how those beliefs correspond to the external world (De Finetti, 1975). The remainder of this thesis is devoted partly to addressing difficulties that arise when searching for Bayes-optimal decisions, and partly to modifying the concept of Bayes-optimality itself to account for what could be called *meta-epistemic* uncertainty, or uncertainty in our prior and likelihood, without resorting to an infinite recursion of hierarchical models.

The main interest of physical statistics lies in fact not so much in the distribution of the phenomena in space, but rather in their succession in time.

Richard von Mises

2

Incrementally Conditioning Sparse GP Posteriors

2.1 Motivation

The ability to repeatedly adapt to new information is a defining feature of intelligent agents. Indeed, these *online* or *streaming* settings, where we observe data in an incremental fashion, are ubiquitous — from real-time adaptation in robotics (Nguyen-Tuong et al., 2008) to click-through rate predictions for ads (Liu et al., 2017).

2.2 Overview

Bayesian inference is naturally suited to the online setting, where after each new observation, an old posterior becomes a new prior. However, these updates can be prohibitively slow. For Gaussian processes, if we have already observed n data points, observing even a single new point requires introducing a new row and column into an $n \times n$ covariance matrix, which can incur $\mathcal{O}(n^2)$ operations for the predictive distribution and $\mathcal{O}(n^3)$ operations for kernel hyperparameter updates.

Since Gaussian processes are now frequently applied in online settings, such as Bayesian optimization (Yamashita et al., 2018; Letham et al., 2019), or model-based robotics (Xu et al., 2014; Mukadam et al., 2016), this scaling is particularly problematic. Moreover, despite the growing need for scalable online inference, recent research on this topic is scarce.

Existing work has typically focused on data sparsification schemes paired with low-rank kernel updates (e.g., Nguyen-Tuong et al., 2008), or sparse variational posterior approximations (Cheng & Boots, 2016; Bui et al., 2017a). Low-rank kernel updates are sensible but still costly, and data-sparsification can incur significant error. Variational approaches, while promising, can provide miscalibrated uncertainty representations compared to exact inference

(Jankowiak et al., 2020a; Lázaro-Gredilla & Figueiras-Vidal, 2009; Bauer et al., 2016), and often involve careful tuning of many hyperparameters. In the online setting, these limitations are especially acute. Uncertainty representation can be particularly crucial for determining the balance of exploration and exploitation in choosing new query points. Moreover, while tuning of hyperparameters and manual intervention may be feasible for a fixed dataset, it can become particularly burdensome in the online setting if it must be repeated after we observe each new point.

Intuitively, we ought to be able to recycle computations to efficiently update our predictive distribution after observing an additional point, rather than starting training anew on $n + 1$ points. However, it is extremely challenging to realize this intuition in practice, for if we observe a new point, we must compute its interaction with every previous point. In this paper, we show it is in fact possible to perform constant-time $\mathcal{O}(1)$ updates in n, and $\mathcal{O}(p^2)$ for p inducing points, to the Gaussian process predictive distribution, marginal likelihood, and its gradients, *while retaining exact inference.* We achieve this scaling through a careful combination of caching, structured kernel interpolation (SKI) (Wilson & Nickisch, 2015), and reformulations involving the Woodbury identity. We name our approach **W**oodbury **I**nversion with **SKI** (WISKI). We find that WISKI achieves promising results across a range of online regression and classification problems, Bayesian optimization, and an active sampling problem for estimating malaria incidence where fast online updates, exact inference for calibrated uncertainty, and fast test-time predictions are particularly crucial. The WISKI method and results were originally published in Stanton et al. (2021).

As a motivating example, in Figure 2.1, we fit GPs with spectral mixture kernels (Wilson & Adams, 2013) on British pound to USD foreign exchange data.* In this task, we observe points one at a time, after observing the first 10 points in batch, and update the predictive distributions for WISKI, O-SVGP and O-SGPR (Bui et al., 2017a), state-of-the-art streaming sparse variational GPs. We illustrate snapshots after having observed $n = 10, 20$, and 30 points. We see that WISKI is able to more easily capture signal in the data, whereas O-SVGP tends to underfit and O-SGPR underfits on the random data setting. In addition to the general tendency of stochastic variational GP (SVGP) models to underfit the data and overestimate noise variance (Lázaro-Gredilla & Figueiras-Vidal, 2009; Bauer et al., 2016), the variational posterior of an O-SVGP is discouraged from adapting to surprising new observations (See Appendix A.2). We also see that O-SVGP particularly struggles when we observe new points in a time-ordered fashion, which is a standard setup in the online setting.

The initialization heuristics used to train SVGPs in the batch setting, such as initializing the inducing points with k-means or freezing the GP hyperparameters at the beginning of training, are not effective for O-SVGPs since the full dataset is not available. In order to obtain reasonable fits with O-SVGP

*https://raw.githubusercontent.com/trungngv/cogp/master/data/fx/fx2007-processed.csv, fourth column. We rescaled the inputs to $[-1, 1]$ and standardized the responses to 0 mean and unit variance.

on even this motivating example, we carefully tuned tempering parameters using generalized variational inference (Knoblauch et al., 2019), executed 6 optimization steps for each new observation, and trained in batch on the first 10 points. WISKI, by contrast, requires no tuning, only 1 optimization step for each new observation, and does not require any batch training to find reasonable solutions.

These issues with O-SVGP[*] are particularly visible when we move beyond time series. In Figure 2.2, we plot the incremental RMSE on a held out test set on the UCI powerplant dataset, while optimizing for only a single step as we observe new data points, finding that O-SVGP underfits and sub-optimal solution, while WISKI matches the performance of an exact GP also fit incrementally. The exact GP also uses pre-conditioned conjugate gradients (Gardner et al., 2018a) here. However, WISKI and O-SVGP are both constant time (shown in the left panel), while using an exact GP with Cholesky factorization is cubic time, and using CG with the GP is quadratic time. Both are much slower than WISKI and O-SVGP after $t = 5000$.

Intelligent systems should be able to quickly and efficiently adapt to new data, adjusting their prior beliefs in response to the most recent events. These characteristics are desirable whether the system in question is controlling the actuators of a robot, tuning the power output of a laser, or monitoring the changing preferences of users on an online platform. What these applications share in common is a constant stream of new information. In this paper, we

[*]O-SGPR has different weaknesses, including numerical instability. We further consider O-SGPR in our larger study of incremental regression.

are interested in efficient *conditioning*, meaning that we wish to efficiently update a posterior distribution after receiving new data.

The ability of Gaussian process (GP) regression models to condition on new data in closed form has made them a popular choice for Bayesian optimization (BO), active learning, and control (Frazier, 2018). All of these settings share similar characteristics: there is an "outer loop", where new data is acquired from the real world (e.g. an expensive simulator), interleaved with an "inner loop", which chooses where to collect data. In BO, for example, the "inner loop" is the optimization of an acquisition function evaluated using a surrogate model of the true objective. Simple acquisition functions, e.g. expected improvement (EI), consider only the current state of the surrogate, while more sophisticated acquisition functions "look ahead" to consider the effect of hypothetical observations on future surrogate states. One such acquisition function, batch knowledge gradient (qKG), defines the one-step Bayes-optimal data batch as the batch that maximizes the expected surrogate maximum *after* the batch has been acquired (Balandat et al., 2020a; Wu & Frazier, 2016). Advanced acquisition functions like qKG require the surrogate to have both efficient posterior sampling and efficient conditioning on new data.

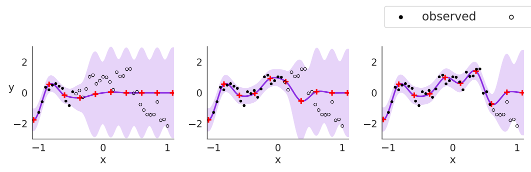
GP regression has two major limitations that have prevented its large scale deployment for online decision-making. First, the computational and memory consumption of exact GPs grows at least quadratically with the amount of data (Gardner et al., 2018b; Rasmussen & Williams, 2008), generally limiting their usage to BO problems with fewer than $1,000$ function evaluations

(Frazier, 2018; Balandat et al., 2020a; Wang et al., 2018). Second, they are limited to applications that have continuous real-valued responses, enabling modelling with solely a Gaussian likelihood. Stochastic variational Gaussian processes (SVGPs) (Hensman et al., 2013b) have constant computational and memory footprints and are applicable to non-Gaussian likelihoods, but they sacrifice closed form expressions for updated posteriors on receiving new data. The SVGP posterior is *optimized* through the evidence lower bound (ELBO). In the online setting, training with the ELBO has two primary difficulties: the need to specify a fixed number of observations to properly scale the ELBO gradient (Broderick et al., 2013) and the need to adjust the inducing points without looking at past data (Bauer et al., 2016; Bui et al., 2017a). Thus, we are presented with a choice between the simplicity and analytic tractability of exact GPs and the scalability and flexibility of SVGPs.

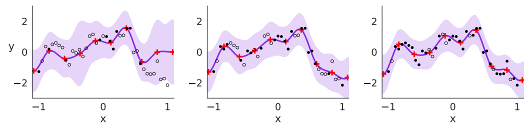
In this work, we develop **O**nline **V**ariational **C**onditioning (OVC) to allow SVGPs to be conditioned on-the-fly, as shown in Figure 2.3. In the top row of each subplot, we fit the data points shown in red, shifting to another batch of data points in the bottom row. We use an exact GP in Figure 2.3a, with exact conditioning shown in the bottom panel. The SVGP emulates the exact GP very well before seeing the new data and again after conditioning on the new data by using OVC (Figure 2.3b). In Figure 2.3c, we consider a non-Gaussian data model (a Gaussian copula volatility model (Wilson & Ghahramani, 2010)), where we cannot use exact GPs; the SVGP is still able to update its posterior over the latent volatility in response to new data without "forgetting" old observations. OVC is inspired by a new, simple rederivation of streaming sparse GPs (O-SGPR), originally proposed by Bui et al. (2017a). OVC makes SVGPs truly compelling models for online decision-making, augmenting their existing strengths with efficient, closed-form conditioning on new data points. In short, our contributions, originally published in Maddox et al. (2021c), are:

- The development of OVC, a novel method to condition SVGPs on new data without re-optimizing the variational posterior through an evidence lower bound.
- OVC provides both stable inducing point initialization for SVGPs while enabling the inducing points and variational parameters to update in response to the new data.
- Enabling the effective application of SVGPs, through OVC, in look-ahead acquisitions in BO for black-box optimization, controlling dynamical systems, and active learning.

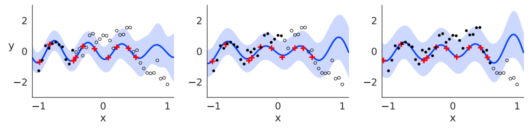
Please see Appendix B.1 for discussion of the limitations and broader impacts of our work. Our code is available at `https://github.com/wjmaddox/online_vargp`.



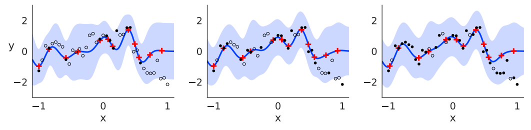
(a) WISKI, time-ordered observations



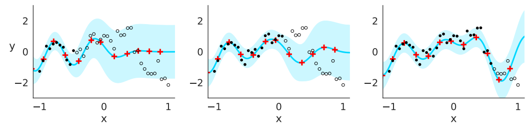
(b) WISKI, randomly-ordered observations



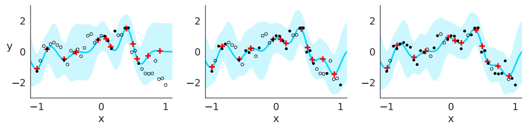
(c) O-SVGP, time-ordered observations



(d) O-SVGP, randomly-ordered observations



(e) O-SGPR, time-ordered observations



(f) O-SGPR, randomly-ordered observations

Figure 2.1: Online GP regression on exchange rate time series data ($N = 40$). The shaded regions in each panel corresponds to a 95% credible interval. In each subplot, the left subpanel shows the predictive distribution of the corresponding model after training in batch on an initial set of 10 observations. The middle and right subpanels show the evolution of the predictive distribution after 10 and 20 online updates, respectively. The **left** plots, **(a,c,e)**, show WISKI, O-SVGP, and O-SGPR using spectral mixture kernels (Wilson & Adams, 2013) trained on observations in a time-ordered fashion. O-SVGP heavily overfits to the initial data by interpolating the first batch of data points, and struggles to recover on the next batches. WISKI and O-SGPR perform well in this situation by picking up the signal on the first batches and updating the mean as the data comes in. The **right** plots, **(b,d,f)** show the methods trained on observations in a randomly ordered fashion. Here, O-SVGP is still very under-confident, while O-SGPR clumps its inducing points in the middle of the data. By comparison, WISKI learns more of the high frequency trend than either variational approach.

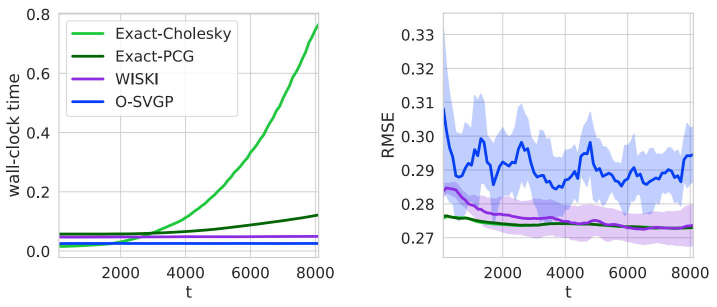
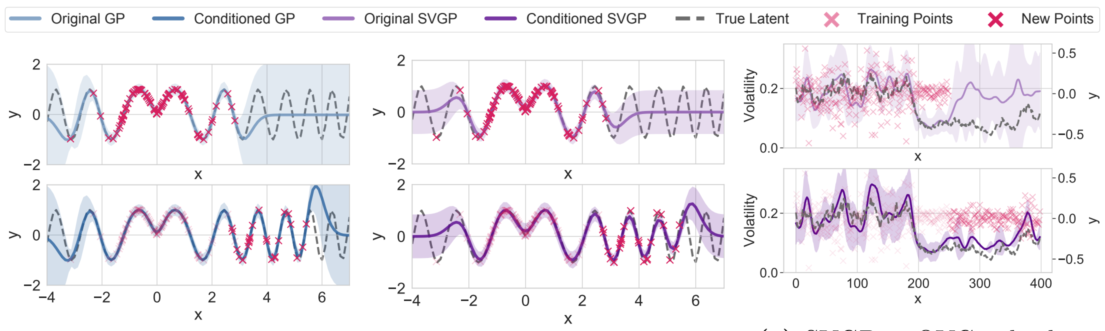


Figure 2.2: Left: Incorporating new observations becomes increasingly expensive for exact GPs (Exact-Cholesky), even when preconditioned conjugate gradients (Exact-PCG), as quantified in the left panel by the wall-clock time per iteration on the UCI Powerplant dataset. Variational GPs (O-SVGP) are an economical alternative by virtue of being constant time. WISKI has the constant-time profile of a variational method, but retains exact inference, is simple to train, and does not underfit. **Right:** RMSE on the UCI power plant dataset. Shown are mean and two standard deviations over 10 trials. O-SVGP tends to overestimate noise and converges to a sub-optimal solution, while WISKI matches the performance of the exact methods trained in an incremental fashion.



(a) Exact GP

(b) SVGP + OVC

(c) SVGP + OVC volatility model

Figure 2.3: An exact GP updates its predictive distribution after conditioning on new data points **(a,** moving from top row to bottom row). With OVC, we can condition SVGPs on both Gaussian responses **(b)** and non-Gaussian models **(c)** such as the Gaussian copula volatility model (Wilson & Ghahramani, 2010).

2.3 Scaling GPs in the Offline Setting

2.3.1 Caching and Parallelism

The cost of exact GP regression is dominated by solving systems of linear equations in Eq. (1.2), (1.3), and (1.4), resulting in $\mathcal{O}(mn^3)$ computational complexity during training and $\mathcal{O}(qmn^2)$ complexity at test time. When repeatedly computing $p(\mathbf{b}|D)$ at different sets of query points X_{test}, it is more efficient to cache the terms which depend only on the training data.* Specifically, we store $\mathbf{v} := (K_{\mathbf{aa}} + \Sigma_{\mathbf{y}})^{-1}\mathbf{y}_{\mathbf{a}}$ (the predictive mean cache) and $RR^\top := (K_{\mathbf{aa}} + \Sigma_{\mathbf{y}})^{-1}$ (the root decomposition cache), resulting in simplified forms for the predictive distribution, $\mathbf{m}_{\mathbf{b}|D} = K_{\mathbf{ba}}\mathbf{v}$ and $S_{\mathbf{b}|D} = K_{\mathbf{bb}} - K_{\mathbf{ba}}RR^\top K_{\mathbf{ab}}$.

The complexity of the root decomposition is $\mathcal{O}(kn^2)$, requiring $k \leq n$ iterations of the Lanczos algorithm (Lanczos, 1950) and a subsequent eigendecomposition of the resulting $k \times k$ symmetric tridiagonal matrix. Further details on Lanczos decomposition and the caching methods of (Pleiss et al., 2018a) are in Appendix A.1.

2.3.2 Online Conditioning and Low-Rank Matrix Updates

GP models are conditioned on new observations through Gaussian marginalization (Williams & Rasmussen, 2006, Chapter 2). Suppose we have past observations $D = \{(\mathbf{x}'_i, y'_i)\}_{i=1}^n$ used to make predictions via $p(y|\mathbf{x}, D)$. We subsequently observe a new data point $(\mathbf{x}_n, y_n)$. For clarity, let $\mathbf{a}' = [f(\mathbf{x}'_0) \cdots$

*We refer to entities that can be computed, stored in memory, and used in subsequent computations as caches. We use orange font to identify which cached expressions.

$f(\mathbf{x}'_{n-1})]^\top$, $\mathbf{b} = [f(\mathbf{x}_n)]^\top$. The new kernel matrix is

$$K_{\mathbf{aa}} = \begin{pmatrix} K_{\mathbf{a}'\mathbf{a}'} & K_{\mathbf{a}'\mathbf{b}} \\ K_{\mathbf{ba}'} & K_{\mathbf{bb}} \end{pmatrix} \tag{2.1}$$

We would like to update our posterior predictions to incorporate the new data point without recomputing our caches that are not hyper-parameter dependent from scratch. If the hyperparameters are fixed, this can be a $\mathcal{O}(n^2)$ low-rank update to the root decomposition cache (e.g. a Schur complement update or low rank Cholesky update to a decomposition of R). If we additionally wish to update hyper-parameters, we must recompute the log marginal likelihood in Eq. 1.4, which costs $\mathcal{O}(n^3)$. Similarly, if we naively use the SKI approximations in Eq. 2.1 we additionally have an $\mathcal{O}(n)$ cost for both adding a new data point and to update the hyper-parameters afterwards. Thus, as n increases, training and prediction will slow down (Figure 2.2).

Adding a new observation is equivalent to adding a single row and column to $K_{\mathbf{a}'\mathbf{a}'}$ and an entry to $\mathbf{y}$, which enables efficient low-rank updates to the predictive caches (Osborne, 2010; Gardner et al., 2018b; Pleiss et al., 2018b; Jiang et al., 2020a). We build on previous work on scaling GP training and prediction by exploiting kernel structure and efficient GPU matrix vector multiply routines to quickly compute gradients (CG) of Eq. 1.4 for training, and by caching terms in Eq. 1.2 and Eq. 1.3 for fast prediction (Gardner et al., 2018a). Conjugate gradient methods improve the asymptotic complexity of GP regression and to $\mathcal{O}(jn^2)$, where j is the number of CG steps used. These recent advances in GP inference have enabled exact GP regression on datasets of up to one million data points in the batch setting (Wang et al., 2019).

2.3.3 Sparse Kernel Approximations

GPs are often *sparsified* through the introduction of inducing points $Z \in \mathbb{R}^{p \times d}$ (also known as pseudo-inputs), which are small subset of fixed points (Snelson & Ghahramani, 2006). In particular, Wilson & Nickisch (2015) proposed structured kernel interpolation (SKI) to approximate the kernel matrix as

$$K_{\mathbf{a}'\mathbf{a}'} \approx \tilde{K}_{\mathbf{a}'\mathbf{a}'} = W_{\mathbf{a}'} K_{\mathbf{uu}} W_{\mathbf{a}'}^{\top}, \tag{2.2}$$

where $\mathbf{u} = [f(\mathbf{z}_0) \cdots f(\mathbf{z}_{p-1})]$ and $W_{\mathbf{a}'} = [w(\mathbf{x}'_0, Z) \;\; \cdots \;\; w(\mathbf{x}'_{n-1}, Z)]^{\top}$ is an $n \times p$ interpolation matrix. Each vector $(W_{\mathbf{a}'})_i$ is sparse, containing 4^d non-zero entries, where d is the dimensionality of the input data. SKI places the inducing points on a multi-dimensional grid. When k_θ is stationary and factorizes across dimensions, $K_{\mathbf{uu}}$ can often be expressed as a Kronecker product of Toeplitz matrices, leading to fast multiplies. Overall multiplies with $\tilde{K}_{\mathbf{a}'\mathbf{a}'}$ take $\mathcal{O}(n + g(p))$ time, where $g(p) \approx p$ (Wilson & Nickisch, 2015), compared to the $\mathcal{O}(np^2 + p^3)$ complexity associated with most inducing point methods (Quinonero-Candela & Rasmussen, 2005). In short, SKI provides scalable exact inference, through introducing an approximate kernel that admits fast computations.

Pleiss et al. (2018a) propose to cache (i.e. to store in memory) all parts of the predictive mean and covariance that can be computed before prediction, enabling constant time predictive means and covariances. Directly substituting the SKI kernel matrix, $\tilde{K}_{\mathbf{a}'\mathbf{a}'}$, into Eq. 1.2, the predictive mean becomes

$$\mathbf{m}_{\mathbf{b}|D} = W_{\mathbf{b}} \underbrace{K_{\mathbf{uu}} W_{\mathbf{a}'}^{\top} \left(\tilde{K}_{\mathbf{a}'\mathbf{a}'} + \sigma^2 I \right)^{-1} \mathbf{y}_{\mathbf{a}'}}_{\mathbf{v}},$$

where $\mathbf{v}$ is the *predictive mean cache*. Similarly, Eq. 1.3 becomes

$$S_{\mathbf{b}|D} = K_{\mathbf{bb}} - W_{\mathbf{b}} \underbrace{K_{\mathbf{uu}} W_{\mathbf{a}'}^{\top} \left(\tilde{K}_{\mathbf{a}'\mathbf{a}'} + \sigma^2 I \right)^{-1} W_{\mathbf{a}'} K_{\mathbf{uu}}}_{VV^{\top}} W_{\mathbf{b}}^{\top},$$

where where $V = K_{\mathbf{uu}} W_{\mathbf{a}'}^{\top} R$, which we call the *predictive covariance cache.*[*]

2.3.4 Sparse Variational Posterior Approximations

We now briefly review variational sparse Gaussian processes, introducing sparse Gaussian process regression (SGPR), sparse variational Gaussian processes (SVGP), and streaming sparse GPs (O-SGPR). Please see Appendix B.2 for further background and Appendix B.2.1 specifically for further related work.

Variational sparse GPs reduce the computational burden of GP inference through sparse approximations of the kernel matrix. For further reference, see Matthews (2017) and Van der Wilk (2019). These methods define a variational distribution $\phi(\mathbf{u}) = \mathcal{N}(\mathbf{m}_{\mathbf{u}}, S_{\mathbf{u}})$ over the inducing *outputs* $\mathbf{u} = [f(\mathbf{z}_1) \cdots f(\mathbf{z}_p)]^{\top}$, defined at inducing *inputs* $Z = [\mathbf{z}_1, \cdots, \mathbf{z}_p]^{\top}$, where $\mathbf{z}_i \in \mathcal{X}$. The variational methods we discuss here assume the latent function values

[*]Note that for exact GP regression $V = R$.

$f(\mathbf{x}), f(\mathbf{x}')$ are conditionally independent given $\mathbf{u}$ and $\mathbf{x}, \mathbf{x}' \notin Z$, so as to cheaply approximate the predictive posterior $p(\mathbf{b}|\mathcal{D}) \approx q(\mathbf{b}) = \mathcal{N}(\mathbf{m}_{\mathbf{b}|\mathbf{u}}, S_{\mathbf{b}|\mathbf{u}})$. Like exact GP regression, we can compute $q(\mathbf{b})$ in closed form (given $\mathbf{m}_\mathbf{u}, S_\mathbf{u}$),

$$\mathbf{m}_{\mathbf{b}|\mathbf{u}} := K_{\mathbf{bu}} \underbrace{K_{\mathbf{uu}}^{-1}\mathbf{m}_\mathbf{u}}_{\mathbf{v}}, \tag{2.3}$$

$$S_{\mathbf{b}|\mathbf{u}} := K_{\mathbf{bb}} - K_{\mathbf{bu}} \underbrace{K_{\mathbf{uu}}^{-1}(K_{\mathbf{uu}} - S_\mathbf{u})K_{\mathbf{uu}}^{-1}}_{VV^\top} K_{\mathbf{ub}}, \tag{2.4}$$

reducing the test-time complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(np^2)$, which is a significant improvement if $p \ll n$.[*]

Unlike exact GP regression, $\mathbf{m}_\mathbf{u}$ and $S_\mathbf{u}$ are viewed as variational parameters to be optimized. There are two common approaches to finding optimal variational parameters $\mathbf{m}_\mathbf{u}$ and $S_\mathbf{u}$. The seminal work of Titsias (2009a) proposed sparse GP regression (SGPR), which optimizes $\mathbf{m}_\mathbf{u}$ and $S_\mathbf{u}$ in closed form, resulting in a "collapsed" evidence lower bound[†] (ELBO) that only depends on θ and Z. The computational cost of each gradient update to the remaining model parameters is still linear in n, and like exact GP regression, SGPR requires a Gaussian likelihood. Stochastic variational GPs (SVGPs) remedies both these limitations by using gradient-based optimization to learn $\mathbf{m}_\mathbf{u}$ and $S_\mathbf{u}$ alongside Z and θ (Hensman et al., 2013b; 2015). The SVGP objective is an "uncollapsed" ELBO which decomposes additively across the training ex-

[*]The SGPR covariance cache we use is slightly different from the implementation in the prediction strategy in GPyTorch, which stores $K_{\mathbf{a}'\mathbf{u}}K_{\mathbf{uu}}^{-1/2}$. However, they reduce to the same strategy.

[†]A lower bound of the true GP marginal log-likelihood

amples, allowing gradients to be estimated from minibatches of data, reducing the complexity of each gradient update to $\mathcal{O}(qp^2 + p^3)$, where q is the minibatch size.

Here we should emphasize the distinction between constant-time minibatch gradients, and constant-time conditioning. Given an SVGP already trained on some existing data, conditioning jointly on both the old and new data requires storing all the data and making multiple gradient updates to the variational parameters. As the size of the dataset grows, so does the number of gradient steps needed. In contrast by constant-time conditioning we mean a procedure that takes a posterior conditioned on existing data and produces a new posterior conditioned jointly on the old and new data with a fixed amount of compute and memory, regardless of the number of past observations.

One example of constant-time conditioning is found in Bui et al. (2017a), who proposed streaming sparse GPs (which we call online SGPR, or O-SGPR, to distinguish from sparse spectrum GPs (Lázaro-Gredilla et al., 2010)) for incremental learning. We extend their work, providing an alternative, simpler derivation of their model that highlights the connection with SGPR (Titsias, 2009a). Furthermore, our perspective enables us to construct a principled approach to updating inducing point locations as new data arrives, that prevents the “forgetting” of old data induced by the resampling heuristic used by Bui et al. (2017a).

2.4 Related Work

Use of sparse GPs in BO: Sparse GPs have not seen wide adoption in the BO community, with only several preliminary studies that have mostly used basic acquisitions. Nickson et al. (2014) and Krityakierne & Ginsbourger (2015) used expected improvement (EI) with SGPR on several test problems, while McIntire et al. (2016) proposed a sparse GP method using EI to tune free electron lasers (Duris et al., 2020). Stanton et al. (2021) proposed WISKI, an online implementation of a scalable kernel approach called SKI (Wilson & Nickisch, 2015), for low-dimensional BO problems using batch upper confidence bound (qUCB) (Balandat et al., 2020a).

2.4.1 Prior Approaches

Despite its timeliness, there has not been much recent work on online learning with GPs. Older work considers sparse variational approximations to GPs in the streaming setting. Csató & Opper (2002) proposed a variational sparse GP based algorithm in $\mathcal{O}(np^2 + p^3)$ time, specifically for deployment in streaming tasks; however, it assumes that the hyperparameters are fixed. Nguyen-Tuong et al. (2008) proposed local fits to GPs with weightings based on the distance of the test point to the local models. More recently, Koppel (2019) extended the types of distances used for these types of models while using an iteratively constructed coreset of data points. Evans & Nair (2018) proposed a structured eigenfunction based approach that requires one $\mathcal{O}(n)$ computation of the kernel and uses fixed kernel hyper-parameters but learns interpolation weights. Cheng & Boots (2016) proposed a variational stochastic functional gradient descent method in incremental setting with the same time complexity; however, like stochastic variational GPs (Hensman et al., 2013c), Cheng & Boots (2016) assumes the number of data points the model will see is known and set before training begins. Hoang et al. (2015a) proposed a similar variational natural gradient ascent approach, but assumed that the hyper-parameters are fixed during the training procedure, a major limitation for flexible kernel learning.

2.4.2 Streaming SVGP and Streaming SGPR

The current state-of-the-art for streaming Gaussian processes is the sparse variational O-SVGP approach of Bui et al. (2017a) and its "collapsed" non-stochastic variant, O-SGPR, which does not use an explicit variational distribution, like its batch equivalent, SGPR (Titsias, 2009b).

O-SVGP: Unlike its predecessors, O-SVGP is fully compatible with online inference, since it has no requirements to choose the number of data points a priori, and it can update both model parameters and inducing point locations; however, it has the same time complexity as its predecessors: $\mathcal{O}(qp^2 + p^3)$, where q is the size of the batch used to update the predictive distribution and model hyper-parameters. Bui et al. (2017a)'s experiments primarily focused on large batch sizes — practically $q = \mathcal{O}(n)$ — rather than the incremental streaming setting where $q << n$. A major limitation of variational methods in the streaming setting is that conditioning on new observations effectively requires the model parameters to be re-optimized to a minima after every new batch, increasing latency. In Appendix A.2 we include a detailed discussion of the requirements of the original O-SVGP algorithm, and provide a modified generalized variational update by downweighting the prior by a factor of $\beta < 1$ better adapted to the streaming setting to have a strong baseline for comparison. We compare to the generalized O-SVGP implementation in our experiments as O-SVGP.

O-SGPR: O-SGPR is also potentially promising but like O-SVGP falls prey to several key limitations. First, O-SGPR relies on analytic marginalization and so can only be used for Gaussian likelihoods. In Figure 2.1, we implemented the O-SGPR bound in GPyTorch (Gardner et al., 2018a) and it has fair performance for both the random ordering and time ordering settings, though not as good as WISKI. However, this performance comes with two caveats. First, we need to re-sample the inducing points to include some of the new data at each iteration, as is done in Bui et al. (2017a)'s implementation. Second, we found that even in double precision we needed to add a large amount of jitter $\epsilon = 0.01$, while doing the required Cholesky decompositions (there is a matrix subtraction) to prevent numerical instability.

2.5 WISKI: Incrementally Conditioning SKI-GPs in Constant Time

We now propose WISKI, which through a careful combination of caching, SKI, and the Woodbury identity, achieves constant time (in n) updates in the streaming setting, while retaining exact inference. To begin, we present two key identities that result from the application of the Woodbury matrix identity to the inverse of the updated SKI approximated kernel, $\tilde{K}_{\mathbf{a}'\mathbf{a}'}$, after having received n data points. First, we can rewrite the SKI kernel inverse as

$$\begin{aligned}(\tilde{K}_{\mathbf{a}'\mathbf{a}'} + \sigma^2 I)^{-1} &= (W_{\mathbf{a}'} K_{\mathbf{uu}} W_{\mathbf{a}'}^\top + \sigma^2 I_n)^{-1} \\ &= \frac{1}{\sigma^2} I - \frac{1}{\sigma^2} W_{\mathbf{a}'} M_{\mathbf{a}'} W_{\mathbf{a}'}^\top. \qquad (2.5) \\ M_{\mathbf{a}'} &:= (\sigma^2 K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}'}^\top W_{\mathbf{a}'})^{-1}. \qquad (2.6)\end{aligned}$$

Second, we observe that $W_{\mathbf{a}'}^\top W_{\mathbf{a}'}$ is a sum of outer products,

$$\begin{aligned}W_{\mathbf{a}'}^\top W_{\mathbf{a}'} &= \sum_{i=0}^{n-1} w(\mathbf{x}_i, Z) w(\mathbf{x}_i, Z)^\top \\ \Rightarrow M_{\mathbf{a}}^{-1} &= \sigma^2 K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}'}^\top W_{\mathbf{a}'} + W_{\mathbf{b}}^\top W_{\mathbf{b}}, \\ &= M_{\mathbf{a}'}^{-1} + W_{\mathbf{b}}^\top W_{\mathbf{b}}. \qquad (2.7)\end{aligned}$$

If the batch size is small, then Eq. 2.7 is a low-rank update that can be efficiently computed.

Computing Eq. 2.7 as written requires explicit computation of $K_{\mathbf{uu}}^{-1}$. In general, $K_{\mathbf{uu}}$ will have exploitable algebraic structure since Z is a dense grid, which yields fast matrix inversion algorithms; however, the inverse will be very ill-conditioned because many kernel matrices on gridded data have super-exponentially decaying eigenvalues (Bach & Jordan, 2002). We will instead focus on reformulating SKI into expressions that depend only on $K_{\mathbf{uu}}$, W, and $\mathbf{y}$

with a constant $\mathcal{O}(p^2)$ memory footprint and which can be computed in $\mathcal{O}(p^3)$ time.

2.5.1 Computing the Marginal Log-Likelihood, Predictive Mean and Predictive Variance

Substituting Eq. (2.5) into Eqs. (1.4), (1.2), and (1.3), we obtain the following expressions for the marginal log-likelihood (MLL), predictive mean, and predictive variance[*]:

$$\begin{aligned}\log p(D|\theta) = -\frac{1}{2\sigma^2}(\mathbf{y}_\mathbf{a}^\top \mathbf{y}_\mathbf{a} - \mathbf{y}_\mathbf{a}^\top W_\mathbf{a} M_\mathbf{a} W_\mathbf{a}^\top \mathbf{y}_\mathbf{a}) - \\ \frac{1}{2}\left(\log|K_{\mathbf{uu}}| - \log|M_\mathbf{a}| + (n-p)\log\sigma^2\right), \qquad (2.8)\end{aligned}$$

$$\mathbf{m}_{\mathbf{b}|D} = W_\mathbf{b} M_\mathbf{a} W_\mathbf{a}^\top \mathbf{y}_\mathbf{a}, \qquad (2.9)$$

$$S_{\mathbf{b}|D} = K_{\mathbf{bb}} - \sigma^2 W_\mathbf{b} M_\mathbf{a} W_\mathbf{b}^\top. \qquad (2.10)$$

For all derivations see Appendix A.1. We begin by constructing a rank r root decomposition of the matrix $W_\mathbf{a}^\top W_\mathbf{a} \approx LL^\top$, along with the factorization of the (pseudo-)inverse, $(W^\top W)^\dagger = JJ^\top$. The root decomposition $LL^\top$ can be a full Cholesky factorization ($r = p$) for relatively small p (i.e. $p \leq 1000$) or an approximate Lanczos decomposition for larger p, at a one-time cost of $\mathcal{O}(p^2 r)$. Applying the Woodbury matrix identity to Eq. (2.6) and substituting $LL^\top$ for

[*]A similar result holds for fixed noise heteroscedastic likelihoods as well. See Appendix A.1.5 for further details.

$W_{\mathbf{a}}^{\top} W_{\mathbf{a}}$, we have

$$M_{\mathbf{a}} = \sigma^{-2} K_{\mathbf{uu}} - \sigma^{-4} K_{\mathbf{uu}} L Q_{\mathbf{a}}^{-1} L^{\top} K_{\mathbf{uu}}, \tag{2.11}$$

$$Q_{\mathbf{a}} := I_p + L^{\top} \sigma^{-2} K_{\mathbf{uu}} L. \tag{2.12}$$

$Q_{\mathbf{a}}^{-1} L^{\top}$ is a $r \times r$ system, so directly computing Eq. (2.11) requires $\mathcal{O}(r^2 p)$ time for the solve using conjugate gradients, $\mathcal{O}(rp \log p)$ time for the matrix multiplications with $K_{\mathbf{uu}}$ if it has Toeplitz structure, and $\mathcal{O}(p^2)$ for the dense matrix additions, and $\mathcal{O}(kp^2)$ for the root decomposition of $W^{\top} W$, for a final total of $\mathcal{O}(r^2 p + kp^2)$. However, we do not *explicitly* store the matrix M as doing so would require p solves of a $r \times r$ system since $L \in \mathbb{R}^{p \times r}$.

Eqs. (2.8) - (2.10) involve computations of the form

$$M_{\mathbf{a}} \mathbf{c} = \sigma^{-2} K_{\mathbf{uu}} \mathbf{c} - \sigma^{-4} K_{\mathbf{uu}} L Q_{\mathbf{a}}^{-1} L^{\top} K_{\mathbf{uu}} \mathbf{c},$$

which can be computed using only a *single* solve against the matrix $Q_{\mathbf{a}}$ via first multiplying out $\mathbf{h} = \sigma^{-2} L K_{\mathbf{uu}} \mathbf{c}$, and then computing $Q_{\mathbf{a}}^{-1} \mathbf{h}$. Applying the matrix determinant identity to $\log |M_{\mathbf{a}}|$ results in a simplified expression in terms of $\log |Q|$. Taking $\mathbf{c} = W_{\mathbf{a}}^{\top} \mathbf{y}$, we obtain a practical expression for the MLL,

$$\begin{aligned} \log p(D|\theta) = -\frac{1}{2\sigma^2} \big(& \mathbf{y}_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}} - \mathbf{y}_{\mathbf{a}}^{\top} W_{\mathbf{a}} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}} \\ & + \mathbf{h}^{\top} Q_{\mathbf{a}}^{-1} \mathbf{h} \big) - \frac{1}{2} \left(- \log |Q_{\mathbf{a}}| + (n - p) \log \sigma^2 \right). \end{aligned} \tag{2.13}$$

Computing $\mathbf{h}$ costs $\mathcal{O}(p\log p + rp)$, so computing the two quadratic forms are $\mathcal{O}(p\log p + p)$ and $\mathcal{O}(jr^2)$ respectively, assuming j steps of conjugate gradients. We use stochastic Lanczos quadrature to compute the log determinant of $|Q|$ which costs $\mathcal{O}(jr^2)$ (Gardner et al., 2018a). Overall, computation of the MLL becomes $\mathcal{O}(rp + p\log p + jr^2)$.

The predictive mean is similarly computed by taking $\mathbf{c} = W_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}}$, resulting in the expression

$$\mathbf{m}_{\mathbf{b}|D} = \sigma^{-2} W_{\mathbf{b}} K_{\mathbf{uu}}(\mathbf{c} - \sigma^{-2} L Q_{\mathbf{a}}^{-1}\mathbf{h}). \tag{2.14}$$

Similarly taking $\mathbf{c} = W_{\mathbf{b}}^\top$, we derive the predictive covariance

$$S_{\mathbf{b}|D} = K_{\mathbf{bb}} - W_{\mathbf{b}} K_{\mathbf{uu}}(\mathbf{c} - \sigma^{-2} L Q_{\mathbf{a}}^{-1}\mathbf{h}). \tag{2.15}$$

2.5.2 Conditioning on New Observations

When we observe a new batch of observations at time t, we need to update $(W_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}})_t$, $(\mathbf{y}_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}})_t$, and $L_t L_t^\top$. The update to the first two terms is simple:

$$(W_{\mathbf{a}}^\top y_{\mathbf{a}})_{t+1} = (W_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}})_t + (W_{\mathbf{b}}^\top \mathbf{y}_{\mathbf{b}})_t \tag{2.16}$$

$$(\mathbf{y}_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}})_{t+1} = (\mathbf{y}_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}})_t + \mathbf{y}_{\mathbf{b}}^\top \mathbf{y}_{\mathbf{b}} \tag{2.17}$$

We can update L_t in $\mathcal{O}(bpr)$ time by exploiting the low-rank (for small b)

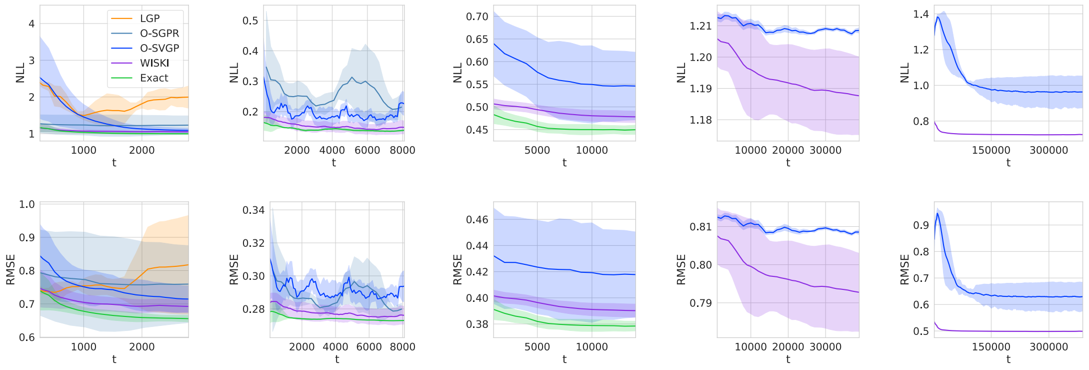
structure of the expression

$$(W_{\mathbf{a}}^{\top} W_{\mathbf{a}})_{t+1} = (W_{\mathbf{a}}^{\top} W_{\mathbf{a}})_t + (W_{\mathbf{b}}^{\top} W_{\mathbf{b}})_t$$

Recalling that $J_t J_t^{\top} = (W^{\top} W)_t^{\dagger}$, let $\mathbf{p} = J_t{}^{\top} W_{\mathbf{b}}$. We compute the decomposition $BB^{\top} = I_r + J_t^{\top} W_{\mathbf{b}} W_{\mathbf{b}}^{\top} J_t$ and obtain the expression for the updated root $L_{t+1} = L_t B$. Since $BB^{\top}$ is a decomposition of I_r plus a rank-b correction, it can be computed in $\mathcal{O}(br)$ time. Since the updates to the first two caches are $\mathcal{O}(b)$ and $\mathcal{O}(bp)$, respectively, the overall complexity of conditioning on a new observation is $\mathcal{O}(pr^2)$. Further details and an extended proof are given in Appendix A.1.

2.5.3 Updating Kernel Hyperparameters

Conventionally, learning the kernel hyperparameters θ of a GP online presents two major challenges. First, the basic form of the gradient of the MLL naively costs at least $\mathcal{O}(n)$, even if scalable methods are employed. Second, after a parameter update, any cached terms that depend on the kernel matrix must be recomputed (e.g. a new root decomposition of $(K_{\mathbf{aa}} + \sigma^2)^{-1}$). The reformulation of the MLL in Eq. (2.13) addresses the first challenge, with a complexity of $\mathcal{O}(rp + p \log p + jr^2)$ (after computing the necessary caches). To address the second challenge, we observe that the combination of the SKI approximation to the kernel matrix and the Woodbury matrix identity in Section 2.5.1 has allowed us to reformulate GP inference entirely in terms of computations whose cost depends only on the number of inducing points and the rank of the ma-



(a) Skillcraft ($n = 2855$) **(b)** Powerplant ($n = 8182$) **(c)** Elevators ($n = 14193$) **(d)** Protein ($n = 39100$) **(e)** 3DRoad ($n = 391000$)

Figure 2.4: Online homoskedastic regression on UCI datasets. We compare to local GPs (LGP), O-SGPR, O-SVGP, and exact GPs. Due to memory constraints or numerical issues for other methods, only O-SVGP and WISKI were easily capable of running on the larger tasks (Protein and 3DRoad). WISKI has comparable accuracy to exact methods, with comparable runtime to scalable approximations like OSVGP. **Top:** Test set NLLs. **Bottom:** Test set RMSEs.

trix decompositions (which is at most p, and typically much less than p). As a result, we can recompute the necessary caches as needed without any increase in computational effort as n increases.

The computational efficiency of SKI is a direct result of the grid structure imposed on the inducing points. The reduced computational complexity comes at the cost of memory complexity that is exponential in the dimension of the input. In practice, if the input data has more than three or four dimensions, the inputs must be projected into a low-dimensional space. The projection may be random (Delbridge et al., 2020) or learned (Wilson et al., 2016b), depending on the requirements of the task. If the projection is learned, then the parameters ϕ of the projection operator h are treated as additional kernel hyperparameters and trained through the marginal log-likelihood.

In the batch setting the interpolation weights $W_{\mathbf{a}}$ are updated after every optimization iteration to adapt to the new projected features $\mathbf{h} = [h_\phi(\mathbf{x}_0) \cdots h(_\phi\mathbf{x}_{n-1})]^\top$. In the online setting, updating $W_{\mathbf{a}}$ for every previous observation would be $\mathcal{O}(n)$. Since we cannot update the interpolation weights for old observations, the gradient for the projection parameters at time t can be rewritten as follows:

$$\begin{aligned}\nabla_\phi \mathcal{L}(\phi) = \nabla_\phi \frac{1}{2}\bigg(& (\mathbf{y}^\top W)_t M_{t-1} (W^\top \mathbf{y})_t \\ -\frac{1}{1+\mathbf{v}^\top \mathbf{w}} & \left(\mathbf{v}_t^\top (W^\top \mathbf{y})_t\right)^2 - \log(1 + \mathbf{v}_t \mathbf{w}_t)\bigg), \\ \mathbf{w}_t &= w(h(\mathbf{x}_t;\phi)), \quad \mathbf{v}_t = M_{t-1}\mathbf{w}_t, \\ (W^\top \mathbf{y})_t &= (W^\top \mathbf{y})_{t-1} + y_t \mathbf{w}_t. \end{aligned} \tag{2.18}$$

The gradient in Eq. (2.18) will move the projection parameters ϕ in a direction that maximizes the marginal likelihood, assuming $\mathbf{w}_{1:t-1}$ are fixed. That is, only projections on new data are updated, while the old projections remain fixed. In contrast to the batch setting, where ϕ is jointly optimized with θ, the online update is sequential; whenever a new observation is received ϕ is updated through Eq. (2.18), then the GP is conditioned on $(h(\mathbf{x}_t; \phi_t), y_t)$, and finally θ is updated through Eq. (1.4). See Appendix A.1.4 for the full derivation.

2.6 WISKI: Empirical Results

To evaluate WISKI, we first consider online regression and binary classification. We then demonstrate how WISKI can be used to accelerate Bayesian optimization, a fundamentally online algorithm often applied to experiment design and hyperparameter tuning (Frazier, 2018). Finally we consider an active learning problem for measuring malaria incidence, and show that the scalability of WISKI enables much longer horizons than a conventional GP.

We compare against exact GPs (no kernel approximations), O-SGPR, O-SVGP (Bui et al., 2017a), sparse variational methods that represents the current gold standard for scalable online Gaussian processes, and local GPs (LGPs) (Nguyen-Tuong et al., 2008). All experimental details (hyper-parameters, data preparation, etc.) are given in Appendix A.3, where we also include ablation studies on the β parameter for O-SVGP as well as the the number of inducing points (as we had to modify it to achieve good results in the incremental setting for O-SVGP), m, for WISKI and O-SVGP. Unless stated otherwise, shaded regions in the plots correspond to $\overline{\mu} \pm 2\overline{\sigma}$, estimated from 10 trials.

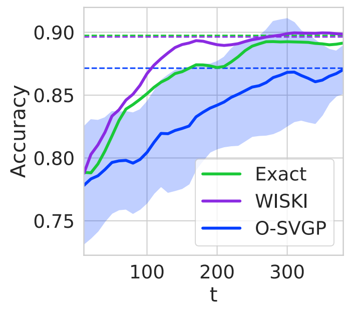
2.6.1 Regression

We first consider online regression on several datasets from the UCI repository (Dua & Graff, 2017). In each trial we split the dataset into a 90%/10% train/test split. We scaled the raw features to the unit hypercube $[-1, 1]^d$ and normalized the targets to have zero-mean and unit variance. Each model learned a linear projection from $\mathbb{R}^d$ to $\mathbb{R}^2$ that was transformed via a batchnorm operation and the non-linear tanh activation to ensure that the features were constrained to $[-1, 1]^2$. Each model learned an RBF-ARD kernel on the transformed features, except on the *3DRoad* dataset, which did not require dimensionality reduction. We used the same number of inducing points for WISKI, O-SGPR, and O-SVGP and set $n_{\max} = m$ for local GPs. The models were pretrained on 5% of the training examples, and then trained online for the remaining 95%. When adding a new data point, we update with a single optimization step for each method, such that the runtime is similar for the scalable methods. Since O-SVGP can be sensitive to the number of gradient updates per timestep, in Figure A.2 in the Appendix we provide results for an ablation.

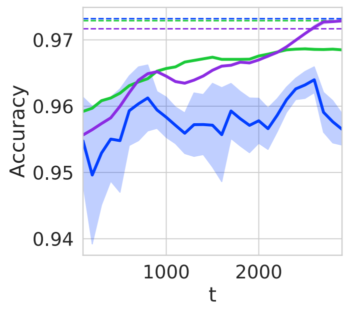
We show the test NLL and RMSE for each dataset in Figure 2.4. For the two largest datasets we only report results for WISKI and O-SVGP. We found that O-SGPR was fairly unstable numerically, even after using an exceptionally large jitter value (0.01) and switching from single to double precision. The exact baseline and the WISKI model overfit less to the initial examples than O-SGPR or O-SVGP. Note that O-SVGP is a much stronger baseline in this experiment since the observations are independently observed than would typically be the case for correlated time-series data, as seen in Figure 2.1.

2.6.2 Classification

We extend WISKI to classification though the Dirichlet-based GP (GPD) classification formulation of Milios et al. (2018), which reformulates classification



(a) Banana ($n = 400$)

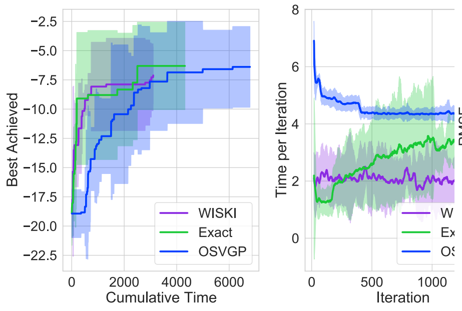


(b) SVM Guide 1 ($n = 3000$)

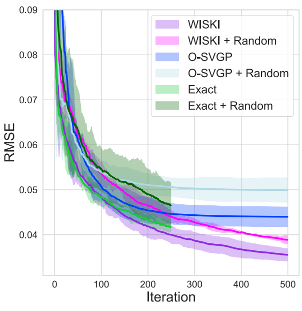
Figure 2.5: A comparison of Dirichlet-based exact and WISKI GP classifiers against an O-SVGP with a binomial likelihood. The exact and WISKI models overfit less to the initial data and ultimately match the performance of their hindsight counterparts trained on the full dataset, shown as a dotted line.

as a regression problem with a fixed noise Gaussian likelihood. Empirically the approach has been found to be competitive with the conventional softmax likelihood formulation. In Figure 2.5 we compare exact and WISKI GPD classifiers to O-SVGP with binomial likelihood on two binary classification tasks, Banana[*] and SVM Guide 1 (Chang & Lin, 2011). Banana has 2D features, and SVM Guide has 4D features, so we did not need to learn a projection. As in the UCI regression tasks, WISKI and O-SVGP both had 256 inducing point, each model used an RBF-ARD kernel, and the classifiers were pretrained on 5% of the training examples and trained online on the remaining 95%. In both cases the WISKI classifier outperformed the O-SVGP baseline, and matched the accuracy of the the exact baseline.

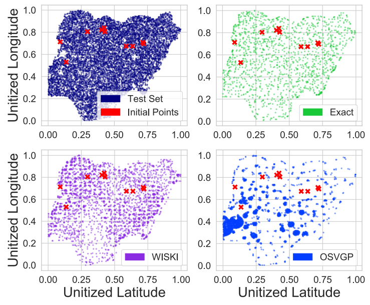
[*]https://raw.githubusercontent.com/thangbui/streaming_sparse_gp/master/data



(a) Bayesian optimization.



(b) Active learning, RMSE.



(c) Active learning, fantasy points.

Figure 2.6: **(a):** Objective value as a function of cumulative time and time per iteration on the Levy test problem with noise standard deviation 10.0, while performing Bayesian optimization with EI acquisition for 1500 steps with a batch size of 3 so that by the end 4500 data points have been acquired. WISKI allows rapid updates of the posterior surrogate objective out to thousands of observations, while preserving the rapid convergence rate and asymptotic optimality of the exact GP. **(b):** RMSE on the test set after choosing new points either randomly or with qnIPV (for WISKI and exact GPs) or by the maximal posterior variance (for O-SVGP). WISKI is able to continue improving the downstream error throughout the entire experiment matching the performance of the exact GP, while O-SVGP's performance flatlines. We also compare against the RMSE of models that have data points randomly selected (shown with Random in the legend). **(c):** The test set (navy), as well as points chosen for all three methods with initial points (red). WISKI and the exact GP query the entire support, while O-SVGP queries clump together.

2.6.3 Bayesian Optimization

In Bayesian optimization (BO) one optimizes a black-box function by iteratively conditioning a surrogate model on observed data and choosing new observations by optimizing an acquisition function based on the model posterior (Frazier, 2018). Thus, BO requires efficient posterior predictions, updates to caches as new data are observed, and hyperparameter updates. While BO has historically been applied only to expensive-to-query objective functions, we demonstrate here that large-scale Bayesian optimization is possible with WISKI. Our BO experiments are conducted as follows: we choose 5 initial observations using random sampling, then iteratively optimize a batched version of upper confidence bound (UCB) (with $q = 3$) using BoTorch (Balandat et al., 2020b) and compute an online update to each GP model, before re-fitting the model. Accurate model fits are critical to high performance; therefore, we wish to use as many inducing points for WISKI and O-SVGP as possible. For both methods, we 1000 inducing points. We show the results over four trials plotting mean and two standard deviations in Figure 2.6a for the Ackley benchmarks. On both problems, WISKI is significantly faster than the exact GP and O-SVGP, while achieving comparable performance on Levy. We show results over a wider range of test functions in Appendix A.3.2, along with the best achieved point plotted against the number of steps and the average time per step.

2.6.4 Active Learning

Finally, we apply WISKI to an active learning problem inspired by Balandat et al. (2020b). We consider data describing the infection rate of *Plasmodium falciparum* (a parasite known to cause malaria)[*] in 2017. We wish to choose spatial locations to query malaria incidence in order to make the best possible predictions on withheld samples. To selectively choose points, we minimize the negative integrated posterior variance (NIPV, Seo et al., 2000), defined as

$$\mathrm{NIPV}(x) := -\int_{\mathcal{X}} \mathbb{E}(\mathbb{V}(f(x)|\mathcal{D}_{\boldsymbol{x}})|\mathcal{D})dx.$$

[*]Downloaded from the Malaria Global Atlas.

Optimizing this acquisition function amounts to finding the batch of data points $\mathbf{x}_{1:q}$, the fantasy points, which when added into the GP model will reduce the variance on the domain of the model the most. Here, we randomly sample $10,000$ data points in Nigeria to serve as a test set that we wish to reduce variance on and select $q = 6$ data points at a time from a held-out training set (to act as a simulator) at a time; the inner expectation drops out because the posterior variance only depends on the fantasy points and the currently observed data, and not any fantasized responses. Stochastic variational models do not have a straightforward mechanism for fantasizing (i.e. recomputing the posterior variance after updating a new data point conditional on the fantasy points), so we instead query the test set predictive variance and then choose the training points closest to the six test set points with maximum predictive variance.

As both mean and variances are available for the given locations, we model the data with a fixed noise Gaussian process with scaled Matern 0.5 kernels, beginning with an initial set of 10 data points, and iterating out for 500 iterations for WISKI and O-SVGP and 250 iterations for an exact GP model (the limit of data points that a single GPU could handle due to the large amount of test points). We show the results of the experiment in Figure 2.6b across three trials, finding that all of the models reduce the RMSE considerably from the initial fits; however, O-SVGP and its random counterpart stagnate in RMSE after about 250 trials, while both the exact GP and WISKI continue improving throughout the entire experiment. Closer examination of the points queried by all of the three methods in Figure 2.6c, we find that the points queried by O-SVGP tend to clump together, locally reducing variance, while WISKI and the exact GP choose points throughout the entire support of the test set, choosing points which better reduce global variance.

```
Algorithm 2: Online Variational Conditioning (OVC)
Input: Data batch $(X_{\text{batch}}, \mathbf{y})$, SVGP with inducing points $Z'$ and
 $\phi(\mathbf{u}') = \mathcal{N}(\mathbf{m}_{\mathbf{u}'}, S_{\mathbf{u}'})$. 1. Compute $\mathbf{c}'$, $C'$ (Eq. 2.20).
2. Compute $\hat{\mathbf{y}} = K'_{\mathbf{u}'\mathbf{u}'} C'^{-1} \mathbf{c}'$ and $\Sigma_{\hat{\mathbf{y}}} = K'^{-1}_{\mathbf{u}'\mathbf{u}'} C' K'^{-1}_{\mathbf{u}'\mathbf{u}'}$ (Eq. 2.23).
3. Construct GP with $\mathcal{D} = ([X_{\text{batch}}\ Z'], [\mathbf{y}\ \hat{\mathbf{y}}])$ and $\Sigma = \text{blkdiag}(\Sigma_{\mathbf{y}}, \Sigma_{\hat{\mathbf{y}}})$.
4. Compute predictive mean and covariance caches, $\mathbf{v}$ and $R$ as in Section
 1.3.
5. Use caches to compute conditioned GP posterior on test points, $X_{\text{test}}$.
```

2.7 OVC: Incrementally Conditioning Variational GPs in Constant Time

We now briefly describe the key ideas behind OVC with the goal of devising an efficient and stable method for updating the variational parameters with respect to newly observed data. We begin by highlighting an alternative parameterization of SGPR that will prove useful. Then we describe the OVC update to the variational distribution from two equivalent points of view, namely the *projection view* and the *pseudo-data view*. The pseudo-data view is summarized in Algorithm 2. Next we address a critical detail for good performance, which is how the inducing point locations should be selected. We then demonstrate how OVC can be applied to compute updated posterior distributions, e.g. $p(f|\mathcal{D}_{+\mathbf{x}})$ in Eq. 1.6, and quantities of the posterior, during gradient-based acquisition function optimization in BO, with reference to how this can be performed practically in Section 2.8. Finally, we discuss how to apply OVC to models with non-Gaussian likelihoods.

2.7.1 Updating the Variational Posterior

We assume that we have trained a SVGP model (e.g. with the ELBO) on a fixed set of data and have already trained the inducing point locations and variational parameters, $\mathbf{m}_{\mathbf{u}}, S_{\mathbf{u}}$. Instead of the traditional $\mathbf{m}_{\mathbf{u}}, S_{\mathbf{u}}$ parameterization used by Titsias (2009a); Hensman et al. (2013b; 2015), we focus for now on an alternative parameterization which was favored in early work on sparse GP inference (Seeger et al., 2003; Opper & Archambeau, 2009). The parameterization is also similar to those used in both dual space functional variational inference (Khan & Lin, 2017) and expectation propagation (Bui et al., 2017b). More recently, Panos et al. (2018) used a similar parameterization in the context of large scale multi-label learning with SVGPs.

The SGPR predictive posterior $q(\mathbf{b})$ relies on two terms dependent on the training data,

$$\mathbf{c} = K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} \mathbf{y}, \qquad C = K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}}, \tag{2.19}$$

where $\Sigma_{\mathbf{y}}$ is the covariance of the likelihood $p(\mathbf{y}|\mathbf{f})$. The optimal $\mathbf{m}_{\mathbf{u}}, S_{\mathbf{u}}$ are then given by

$$\mathbf{m}_{\mathbf{u}} = K_{\mathbf{uu}}(K_{\mathbf{uu}} + C)^{-1} \mathbf{c}, \qquad S_{\mathbf{u}} = K_{\mathbf{uu}}(K_{\mathbf{uu}} + C)^{-1} K_{\mathbf{uu}}, \tag{2.20}$$

which can be substituted into Eq. (2.4) to obtain $q(\mathbf{b})$. Our first observation is that if $\Sigma_{\mathbf{y}}$ is block-diagonal, then $\mathbf{c}$ and C are additive across blocks of observations. For some intuition, consider i.i.d. Gaussian noise (i.e. $\Sigma_{\mathbf{y}} = \sigma^2 I_n$), which implies

$$\mathbf{c}_i = \sum_j \sigma^{-2} y_j k_\theta(\mathbf{z}_i, \mathbf{x}_j) = \phi(\mathbf{z}_i)^\top \sum_j \sigma^{-2} y_j \phi(\mathbf{x}_j),$$
$$C_{ik} = \sum_j \sigma^{-2} k_\theta(\mathbf{z}_i, \mathbf{x}_j) k(\mathbf{x}_j, \mathbf{z}_k) = \phi(\mathbf{z}_i)^\top \sum_j \sigma^{-2} \phi(\mathbf{x}_j)\phi(\mathbf{x}_j)^\top \phi(\mathbf{z}_k),$$

where ϕ is the (potentially infinite-dimensional) feature map associated with k_θ. Hence the entries of $\mathbf{c}$ and C are both inner products between projected inducing points and weighted sums of features. For fixed inducing points, Z, and hyper-parameters θ, we can use these updates to produce a streaming version of SGPR by exploiting the additive structure of c and C. Furthermore, this streaming version of SGPR is exactly Gaussian conditioning for SGPR as we show in Appendix B.3.1. We can also allow the inducing points and hyper-parameters to vary, which we address next.[*]

The projection view: We assume we have

$$\mathbf{c}' = K'_{\mathbf{u}'\mathbf{a}'}\Sigma_{\mathbf{y}'}^{-1}\mathbf{y}' \text{ and } C' = K'_{\mathbf{u}'\mathbf{a}'}\Sigma_{\mathbf{y}'}^{-1}K'_{\mathbf{a}'\mathbf{u}'},$$

computed with inducing point locations Z' from data $(X'_{\text{batch}}, \mathbf{y}')$ with kernel hyperparameters θ' (using shorthand $k_{\theta'} = K'$).[†] After obtaining the next parameters Z and θ (perhaps from gradient based optimization of the ELBO),

[*]For full generality, we allow the hyper-parameters to vary; however, in our BO experiments, we only consider varying the inducing points as that's all we need to update when computing acquisition functions.

[†]Cached computations that depend on $(X'_{\text{batch}}, \mathbf{y}')$ are highlighted in blue.

we observe new data $(X_{\text{batch}}, \mathbf{y})$ and would like to continue with inference. One challenge is translating $\mathbf{c}', C'$ (whose elements are inner products of the old features) to the new feature space associated with θ. To resolve this challenge, we construct a representative set of responses, $\hat{\mathbf{y}} = P^\top \mathbf{c}'$ and likelihood covariance $\hat{\Sigma}_{\hat{\mathbf{y}}} = P^\top C' P$ to project from the old feature space into the new feature space by passing back through data space. The choice that minimizes reconstruction error is the pseudo-inverse $P = (K'_{\mathbf{a}'\mathbf{u}'} K'_{\mathbf{u}'\mathbf{a}'})^{-1} K'_{\mathbf{a}'\mathbf{u}'}$, but requires storage of the full dataset, $(X'_{\text{batch}}, \mathbf{y}')$. Instead we take $P = K'^{-1}_{\mathbf{u}'\mathbf{u}'}$, resulting in the following modifications to Eq. (2.19):

$$\mathbf{c} = K_{\mathbf{uv}} \Sigma_{\mathbf{y}}^{-1} \mathbf{y} + K_{\mathbf{uu}'} K'^{-1}_{\mathbf{u}'\mathbf{u}'} \mathbf{c}', \tag{2.21}$$

$$C = K_{\mathbf{uv}} \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{vu}} + K_{\mathbf{uu}'} (K'^{-1}_{\mathbf{u}'\mathbf{u}'} C' K'^{-1}_{\mathbf{u}'\mathbf{u}'}) K_{\mathbf{uu}'}, \tag{2.22}$$

Note that $K'^{-1}_{\mathbf{u}'\mathbf{u}'} \mathbf{c}' = \Sigma_{\mathbf{y}'}^{-1} \mathbf{y}'$ and $K'^{-1}_{\mathbf{u}'\mathbf{u}'} C' K'^{-1}_{\mathbf{u}'\mathbf{u}'} = \Sigma_{\mathbf{y}'}^{-1}$ in the special case where $X'_{\text{batch}} = Z'$. We also want to emphasize that although we have only considered two batches of data for the sake of clarity, the approach applies to any number of incoming batches.

The pseudo-data view: The above update is equivalent to having an SGPR model with a Gaussian likelihood with covariance $\Sigma = \texttt{blkdiag}(\Sigma_{\hat{\mathbf{y}}}, \Sigma_{\mathbf{y}})$ on the data $\{\texttt{cat}(Z', X_{\text{batch}}), \texttt{cat}(\hat{\mathbf{y}}, \mathbf{y})\}$, where

$$\hat{\mathbf{y}} = K'_{\mathbf{u}'\mathbf{u}'} C'^{-1} \mathbf{c}', \quad \Sigma_{\hat{\mathbf{y}}}^{-1} = K'^{-1}_{\mathbf{u}'\mathbf{u}'} C' K'^{-1}_{\mathbf{u}'\mathbf{u}'}. \tag{2.23}$$

This interpretation is reminiscent of prior online variational approaches of

Csató & Opper (2002) and Opper (1998). That is, in the context of conditioning a SVGP, we can assume that we began with data $\{Z', \hat{\mathbf{y}}\}$ and are now observing the new data $\{X_{\text{batch}}, \mathbf{y}\}$. See Appendix B.3.2 for a more details.

Extending to SVGPs: SGPR computes $\mathbf{m}_\mathbf{u}$ and $S_\mathbf{u}$ as a function of c and C in Eq. (2.20). However the equations can be reversed to solve for c and C given $\mathbf{m}_\mathbf{u}$ and $S_\mathbf{u}$, allowing us to condition *any* variational sparse GP into an SGPR model, without touching any previous observations due to the conditional independence assumptions of variationally sparse GPs.[*] Note that if the variational parameters are not at the optimal solution when the variational distribution is projected back to the pseudo-data, the projection will be to the targets and likelihood *for which the current variational parameters would be optimal*, which may not correspond well to the data that originally created the model. This potential pitfall is mitigated if the variational parameters are well optimized and is offset by the practical advantages of SVGPs.

Connection to O-SGPR (Bui et al., 2017a): Formally, the updates described in Eqs. (2.21) and (2.22) are equivalent to the O-SGPR approach of Bui et al. (2017a), as we show in Appendix B.3.2. The original derivation of O-SGPR is very technical, and does not highlight the similarities between the batch and online SGPR variants. Both the projection and pseudo-data views we have just described provide a much more intuitive way to reason about the behavior of O-SGPR models. Our formulation also eliminates a matrix subtraction operation, which is beneficial for numerical stability.

[*]Alternatively, we could construct SVGPs via direct optimization of c and C.

2.7.2 Inducing Point Selection

Here, we describe inducing point selection during the conditioning procedure to enable better variance reduction on new inputs. While heuristics including re-sampling (Bui et al., 2017a) and data sufficient statistics (Hoang et al., 2015b) have been proposed, they either require the number of inducing points to grow or gradually forget old observations. We show in Appendix B.3.4 that relying exclusively on gradient-based optimization of inducing locations works very poorly in the online setting.

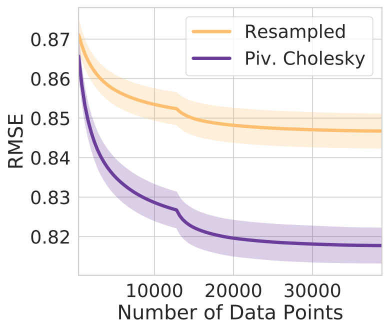


Figure 2.7: Incremental learning RMSE on the UCI *protein* dataset. Pivoted cholesky initialization outperforms resampling.

To update the inducing point locations during conditioning, we extend Burt et al. (2019)'s batch inducing point initialization approach to heteroskedastic Gaussian likelihoods. They consider theoretical bounds on the marginal likelihood, finding that for homoscedastic Gaussian likelihoods a good strategy is to minimize ε, the trace of the error of a rank-p Nyström approximation, i.e.

$$\begin{aligned}\varepsilon &= \texttt{trace}\left(\Sigma_{\mathbf{y}}^{-1/2}(K_{\mathbf{aa}} - K_{\mathbf{au}}K_{\mathbf{uu}}^{-1}K_{\mathbf{ua}})\Sigma_{\mathbf{y}}^{-1/2}\right) \\ &= \sigma^{-1}\texttt{trace}(K_{\mathbf{aa}} - Q_{\mathbf{aa}})\end{aligned}$$

for $Q_{\mathbf{aa}} = K_{\mathbf{au}}K_{\mathbf{uu}}^{-1}K_{\mathbf{ua}}$. They follow a classical approach of Fine & Scheinberg (2001) by a greedy minimiziation strategy: choosing as inducing points the pivots of a rank p pivoted Cholesky factorization of $K_{\mathbf{aa}}$.

We denote the function values over the batch+pseudo dataset as

$$\hat{\mathbf{a}} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_b), f(\mathbf{z}'_1), \ldots, f(\mathbf{z}'_p)]^\top.$$

In our case the covariance of the pseudo-likelihood is no longer homoscedastic, so the slack term becomes $\varepsilon = \texttt{trace}(\Sigma^{-1/2}(K_{\hat{\mathbf{v}}\hat{\mathbf{v}}} - Q_{\hat{\mathbf{v}}\hat{\mathbf{v}}})\Sigma^{-1/2})$ and hence the pivoted Cholesky decomposition is instead performed over $\Sigma^{-1/2}K_{\hat{\mathbf{v}}\hat{\mathbf{v}}}\Sigma^{-1/2}$ to select the top p pivots of the $p + n_{\text{new}}$ matrix. When compared to re-sampling the inducing points (Bui et al., 2017a), pivoted cholesky updates perform significantly better, as shown in Figure 2.7 on the UCI protein dataset (Dua & Graff, 2017). Experimental details are given in Appendix B.4.2.

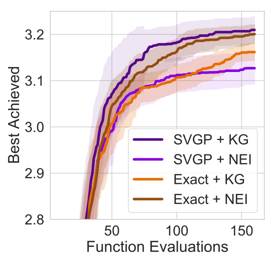
Application to Bayesian Optimization: In the context of BO, we condition on *hypothetical* observations, and the conditioned surrogates are discarded after each acquisition function evaluation. Since the SVGP will not be conditioned on more than a few batches of observations, we can sidestep the issue of updating inducing locations entirely by instead conditioning into an *exact GP* trained the combined pseudo-data through the pseudo-likelihood. That is, we model the data as $(\mathbf{y}, \hat{\mathbf{y}}) \sim \mathcal{N}(f, \Sigma)$ (Gaussian with block-diagonal covariance) assuming $f \sim \mathcal{GP}(\mu_\theta, k_\theta(\cdot, \cdot))$. We reach the same model by choosing $\hat{X}$ as the inducing points in our conditioned SGPR (Section 2.7.1). For small n_t, an exact GP is not much slower, taking only $(n_t + p)^3$ computations instead of p^3 computations, further reduced by using low rank updates.

2.7.3 Local Laplace Approximations for Non-Gaussian Observations

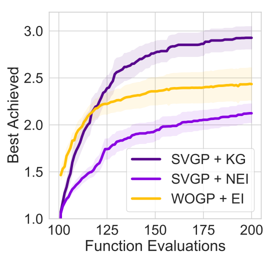
Thus far, we have solely considered Gaussian observations. The introduction of a non-Gaussian likelihood presents a new challenge, since it implies that the current observation batch and the pseudo-data are no longer jointly Gaussian. To adapt the conditioning procedure to the non-Gaussian setting, we can simply perform a Laplace approximation of the likelihood at the new points (Rasmussen & Williams, 2008, Ch. 3). Specifically, this gives us an approximate likelihood, $\hat{p}(y|f) = \mathcal{N}(\tilde{y}; f, \mathcal{H}_*^{-1})$, where $\mathcal{H}_* = \nabla_f^2 \log p(y|f)|_{f^*(y)}$ and $f^*(y)$ is the maximizer of $\log p(y|f) + f^\top K^{-1} f$, computed via Newton iteration.[*] When conditioning on new observations $\mathbf{y}$, we substitute $f^*(\mathbf{y})$ instead. That is, our new model has responses $(f^*(\mathbf{y}), \hat{\mathbf{y}})$ instead of $(\mathbf{y}, \hat{\mathbf{y}})$, and the pseudo-likelihood remains Gaussian with covariance $\Sigma = \text{blkdiag}(\mathcal{H}_*^{-1}, \Sigma_{\hat{\mathbf{y}}})$. We primarily consider natural parameterizations of one-dimensional exponential families, so that $\mathcal{H}_*$ is positive, diagonal and depends solely on f. Computing $\nabla \log p(y|f)$ and $\mathcal{H}_*$ is possible by hand but one can also use automatic differentiation (AD, Pearlmutter, 1994).[†] In Appendix B.4, Figure B.3 we show the effect of repeated Laplace approximations across several batches for online classification.

[*]One could consider using the posterior covariance instead of K. Our experiments with the posterior covariance produced more extreme values of f and thus less regularization.

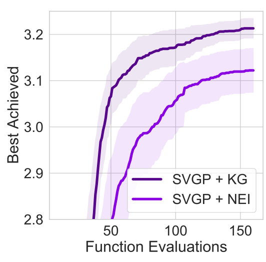
[†]Specifically, we use PyTorch's functional API, https://pytorch.org/docs/stable/autograd.html#functional-higher-level-api.



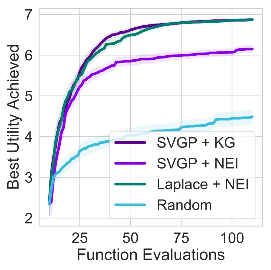
(a) Hartmann6



(b) Laser



(c) Pois.-Hartmann6



(d) Pref. Learning

Figure 2.8: **(a)** Hartmann6 test problem with one constraint. Here, SVGPs with noisy expected improvement (qNEI) and KG match the performance of exact GPs with qNEI and qKG. **(b)** Free electron laser problem from McIntire et al. (2016); SVGPs with knowledge gradient outperforming weighted sparse GPs. **(c)** Constrained Hartmann6 test problem with count responses (Poisson likelihood). Only SVGPs can be used here, and qKG outperforms qNEI. **(d)** Preference learning; SVGPs with qKG are similar to Laplace approximations with NEI, and outperform qNEI with SVGPs.

2.8 OVC: Empirical Results

Our experimental evaluation demonstrates that SVGPs using OVC can be successfully used as surrogate models with advanced acquisition functions in Bayesian optimization, even in the large batch and non-Gaussian settings. In keeping with the BO literature, we will refer to the query batch size as q (not to be confused with the variational posterior $q(f)$ in previous sections). *All SVGP models that use conditioning (or fantasization) require OVC to even be practical to implement.*

Using OVC as a Building Block inside of BO

In all of our experiments, we use OVC as a *building block* to enable fantasization (Algorithm 2) within a standard BO acquisition function that requires fantasiziation. These acquisitions are generally "look-ahead" as a result;

specifically, qKG (Balandat et al., 2020a; Jiang et al., 2020a), LTSs, our version of qGIBBON which uses a fantasy batch (Moss et al., 2021), and qMultiStepLookAhead (Jiang et al., 2020a) all use the fantasization model. After adding in OVC as the `condition_on_observations` function within a BoTorch model class (Balandat et al., 2020a), we can simply optimize qKG or qGIBBON with an SVGP exactly as an exact GP surrogate, by using gradient based optimizers such as L-BFGS-B. In general, we need to differentiate through the fantasy model with respect to the inputs and then use gradient based methods to find the optimum. Please see Balandat et al. (2020a) and Frazier (2018) for description of how a BO loop is constructed and how acquisition functions are optimized.

Experimental Setup In general, a Bayesian optimization loop consists of the steps of training the model and then using the trained model to optimize an acquisition function to acquire new data points, which are then added into the training data for the next model. All experiments use PyTorch (Paszke et al., 2019a), GPyTorch (Gardner et al., 2018b), and BoTorch (Balandat et al., 2020a). Unless otherwise specified, we run each experiment 50 times and report the mean and two standard deviations of the mean.

In the first step, we first train the inducing points, variational distribution, and kernel hyper-parameters using the evidence lower bound given in Eq. B.3. As all components are differentiable, we use the Adam optimizer with a learning rate of 0.1 and optimize for 1000 steps or until the loss converges, whichever is shorter. To initialize the inducing points, we compute a pivoted cholesky factorization on the initial kernel on the training data (described in Section 2.7.2 following Burt et al. (2019)). The kernel hyper-parameters are initialized to GPyTorch defaults (which sets all lengthscales to one), while the variational distribution is initialized to $\mathbf{m}_{\mathbf{u}} = 0$, $S_{\mathbf{u}} = I$ (again, GPyTorch defaults). Further experimental details and dataset descriptions are in the Appendix.

2.8.1 Knowledge Gradient with SVGPs

These experiments use the one-shot formulation of the batch knowledge gradient (qKG) (Eq. 1.6) from Balandat et al. (2020a), who demonstrated that qKG outperforms other acquisitions due to being able to plan two steps into the future. *Using and optimizing qKG has only been available for exact GPs previously.* By using OVC, we have enabled SVGPs to also efficiently and tractably optimize qKG, even for non-Gaussian observations. We compare to batch noisy expected improvement (qNEI, Letham et al., 2019) which is myopic and does not use fantasization (e.g. conditioning). Here, for the SVGPs we used $\min(N, 25)$ inducing points.

Gaussian observations: We use the **Hartmann6** test function, with one black box constraint, maximizing $f(x) = -\sum_{i=1}^{4} \alpha_i \exp\{-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2\}$ subject to the constraint that $c(x) = ||x||_1 \leq 3$ for fixed A, P, α. We use 10 initial points and a batch size of 3 optimizing for 50 iterations, comparing to SVGPs and exact GPs using qNEI. We show the results in Figure 2.8a where SVGPs with qKG match exact GPs with both qNEI and qKG, and outperform SVGPs using qNEI.

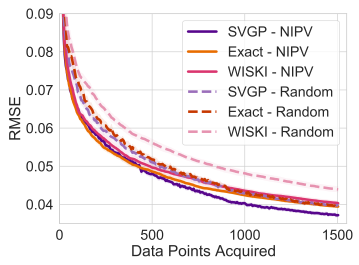
Second, we mimic the **laser tuning** experiment of (McIntire et al., 2016; Duris et al., 2020), demonstrating that SVGPs outperform even weighted online GPs (WOGP), which were designed for this task. Here, we use 100 initial points, with $d = 14$, and and wish to tune a laser's output energy as a function of the magnet settings that produce the beam. Like McIntire et al. (2016) we treat a pretrained GP fit on experimental data as a simulator. We use a batch size of 1, finding that SVGPs + KG outperform WOGP (Figure 2.8b). However, exact GPs outperform the variational approaches (Appendix Fig. B.4b).

Non-Gaussian likelihoods: Next, we extend the knowledge gradient to problems with non-Gaussian likelihoods. First, we take the constrained Hartmann6 test function from the previous section, and use **Poisson** responses, $y \sim \text{Poisson}(\exp\{f(x)\})$, repeating the same settings as for the Gaussian case. Now, the data is non-Gaussian and cannot be well-modelled by a Gaussian likelihood, so we compare to only SVGPs with qNEI. qNEI is outperformed by qKG, as shown in Figure 2.8c.

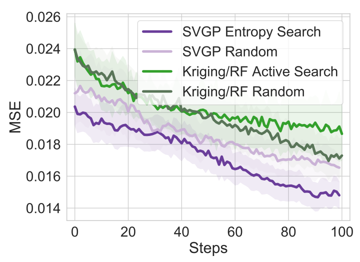
Second, in Figure 2.8d, we consider a **preference learning** problem inspired by Lin et al. (2020). Here, the latent data is described by

$$f(x) = -10^{-1/2} \sum_{i=1}^{10} \sqrt{i} \cos(2\pi x_i) \text{ for } x \in [0, 1]^{10},$$

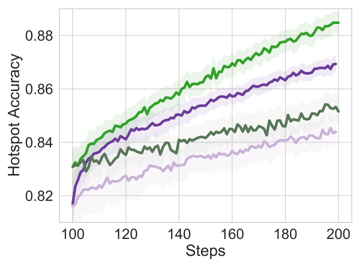
comparing to Laplace approximations (Chu & Ghahramani, 2005). Again, we see that SVGPs with qKG outperform qNEI with both SVGPs and Laplace approximations.



(a) Malaria incidence in Nigeria



(b) Schistomiasis incidence in Cote d'Ivoire



(c) Schistomiasis hotspot prediction

Figure 2.9: **(a)** Active learning of malaria incidence from satellite data. Using qNIPV outperforms randomly selecting points, while the SVGPs slightly outperform both exact GPs and WISKI. **(b,c)** Active learning of schistomiasis incidence in Cote d'Ivoire from Andrade-Pacheco et al. (2020). Comparison is to the random forest based approach using active sampling. While the GP based models are somewhat less accurate at predicting hotspots **(b)**, they are better as a global model of prevalence **(c)**.

2.8.2 Active Learning of Disease Incidence

We next present results for two active learning tasks governing the collection of disease incidence data. In both tasks the acquisition functions again require efficient conditioning on hypothetical data, and the second task has Binomial responses, so exact GPs cannot be applied. In both settings, applying OVC to SVGPs gives strong results competitive with either exact GPs or random forests.

Modelling of Malaria Incidence: We consider data from the Malaria Global Atlas (Weiss et al., 2019) describing the infection rate of a parasite known to cause malaria in 2017. We wish to choose spatial locations to query malaria incidence in order to make the best possible predictions on a withheld test set, the entire country of Nigeria. Following Stanton et al. (2021), we minimize the negative integrated posterior variance (NIPV, Seo et al., 2000), defined as

$$a(\mathbf{x}; \mathcal{D}) := -\int_{\mathcal{X}} \mathbb{E}(\mathbb{V}(f(\mathbf{x})|\mathcal{D}_{+\mathbf{x}})|\mathcal{D}) d\mathbf{x},$$

again with $\mathcal{D}_{+\mathbf{x}} = \mathcal{D} \cup \{(\mathbf{x}, y)\}$. Intuitively, the minimizer of this acquisition will be the batch of data points that when added into the model will most reduce the total posterior uncertainty across the domain, requiring efficient conditioning to do so in a tractable manner. The results are shown for a batch size of $q = 6$ across 15 trials in Figure 2.9a where we see that each method outperforms random baselines. Perhaps due to the optimization freedom, the SVGP outperforms both the exact GP and WISKI (Stanton et al., 2021).

Hotspot Modelling: We follow Andrade-Pacheco et al. (2020) and model the prevalence of schistomiatosis in Côte d'Ivoire using simulated responses from 1500 villages in that country, and taking into account six other demographic variables. We model the responses y (incidence) at locations $\mathbf{x}$ with population $n(\mathbf{x})$ with a Binomial likelihood $p(y|f, \mathbf{x}) \sim \text{Binomial}(n(\mathbf{x}), r(f))$, where $r(f) = (1 + \exp\{-f\})^{-1}$. Letting $\tau \in (0, 1)$ be a threshold on the prevalence for a location to be considered a "hotspot" (Andrade-Pacheco et al., 2020), we compute the entropy:

$$\begin{aligned} h_\tau(\mathbf{x}, \mathcal{D}) &:= \mathbb{E}_{p(f|\mathcal{D})}(\mathbb{H}(\text{Bernoulli}(f > \text{logit}(\tau)))), \\ &\approx \frac{1}{K} \sum_{i=1}^{K} 1_{f > \text{logit}(\tau)} \mathbb{H}(\text{Bernoulli}(f)), \end{aligned}$$

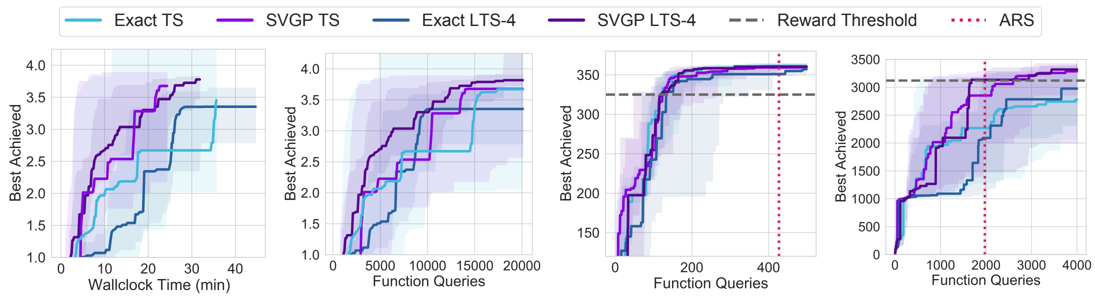
taking the acquisition value to be the reduction in the entropy of the posterior predictive distribution over the incidence under the hotspot-focused likelihood.

$$a_\tau(\mathbf{x}, \mathcal{D}) := \int_{\mathbf{x}' \in \mathcal{X}} \big(h_\tau(\mathbf{x}', \mathcal{D}_{+\mathbf{x}}) - h_\tau(\mathbf{x}', \mathcal{D})\big) d\mathbf{x}'. \tag{2.24}$$

A location is given a high acquisition value if observing the incidence at that location reduces the uncertainty of the model on the predicted set of hotspots. In Figure 2.9c, we compare to Andrade-Pacheco et al. (2020) who use spatial kriging on the residuals of a random forest model. Both their random baseline and their exploration based procedure (a variant of UCB) start off with higher prediction accuracy; however our SVGP models ultimately outperform the kriging approach with random selection. The SVGP is a better predictor of true prevalence, as shown in Figure 2.9b. In both cases, our acquisition function significantly outperforms random selection with a SVGP surrogate.

2.8.3 Rollouts within Thompson Sampling for High Dimensional BO

For our final set of experiments, we solve control problems using trust region Bayesian optimization (TurBO, Eriksson et al., 2019). Inspired by multi-step look-aheads (Jiang et al., 2020a; Bertsekas, 2020), we propose h-step look-ahead Thompson sampling (LTS-h). In BO, Thompson sampling (TS) is often implemented by drawing samples from the posterior at points all over the domain, then selecting the q best to form a query batch (Thompson, 1933; Eriksson et al., 2019). LTS-h extends the idea by conditioning the surrogate independently on each posterior sample in the original TS query batch, Thompson sampling again from the updated posterior with a new set of points, and



(a) Wall-Clock time, rover, $d = 60$

(b) rover, $d = 60$

(c) swimmer, $d = 16$

(d) hopper, $d = 33$

Figure 2.10: Multi-step rollouts with OVC and SVGPs provides sample-complexity and wall-clock time improvements on high dimensional BO problems when using TurBO and LAMCTS (Eriksson et al., 2019; Wang et al., 2020). SVGP rollouts are as time efficient (to 150 iterations) as standard TS **(a)** on lunar rover, $d = 60$. **(c-d)** MuJoCo environments using LAMCTS + TurBO. Also shown is the reward threshold (dashed grey lines) and augmented random search (ARS)'s performance (dotted red lines) (Mania et al., 2018). The median and its 95% confidence interval are shown over 24 trials for rover and 10 trials for swimmer and hopper.

appending the best sample to its predecessor to form a *path*. The process is repeated h times. Finally, we condition the original surrogate jointly on each path, then perform TS again to choose the new query batch. Informally, each path corresponds to a distinct, coherent draw from $p(f|\mathcal{D})$, allowing the inner-loop to refine its guess of the global optimum for different f, and the final round of TS chooses the query batch based on those guesses. See Appendix B.3.5 for a formal description. Like other look-ahead acquisitions, LTS-h is only practical if posterior conditioning and samples are very efficient and numerically stable. LTS-h is conceptually similar to path sampling for look-ahead in Jiang et al. (2020a) and kriging believer (Ginsbourger et al., 2010).

For validation, we consider tuning the 60 dimensional path that a **lunar rover** takes across a field stacked with obstacles (Wang et al., 2018; Eriksson et al., 2019). We use batches of $q = 100$ with 200 initial points, and use trust region Bayesian optimization (TurBO) to split up the space effectively (Eriksson et al., 2019), comparing to TurBO with Thompson sampling (TS) as the acquisition (Thompson, 1933). We show the wall clock times per iteration in Figure 2.10a, where we see that the TurBO + LTS approaches compare well in wallclock time to TurBO+TS, while being more function efficient (Figure 2.10b).

Finally, we consider **MuJoCo** problems using the OpenAI gym (Todorov et al., 2012; Brockman et al., 2016) with LTSs inside of TurBO with trust regions generated by Monte Carlo Tree Search following the procedure of Wang et al. (2020). Following Wang et al. (2020), we learn a linear policy and consider the `swimmer-v2, hopper-v2` environments over 10 trials displaying the median and its 95% confidence band due to high variance. On both problems, SVGP with LTSs tend to be the most sample efficient, with SVGP + TS performing at least as well on `swimmer-v2`. We also show the reward threshold and the performance achieved by augmented random search (ARS), which is a strong baseline reinforcement learning method that uses random search to tune linear controllers (Mania et al., 2018). Our results suggest that LTSs are promising overall; however, more work needs to be done for high dimensional kernels on these problems.

2.9 Discussion

We have shown how to achieve constant-time online updates with Gaussian processes while retaining exact inference. Our approach, WISKI, achieves comparable performance to Gaussian processes with exact kernels, and comparable speed to state-of-the-art streaming Gaussian processes based on variational inference. Despite the present day need for scalable online probabilistic inference, recent research into online Gaussian processes has been relatively scarce. We hope that our work is a step towards making streaming Bayesian inference more widely applicable in cases when both speed and accuracy are crucial for online decision making.

... the main use I would make of [probability theory] ... is normative, to police my own decisions for consistency and ... to make complicated decisions depend on simpler ones.

Leonard J. Savage

3 Bayesian Optimization for Biological Sequence Design

3.1 Motivation

Modern drug development is a very costly endeavor, with estimates ranging from \$310M to \$2.8B for each new drug (Wouters et al., 2020). There are three major phases of drug development: 1) *target discovery*, the proposal of a biological mechanism hypothesized to treat a specific medical condition, 2) *drug design*, the specification of a molecular payload which will interact with the proposed mechanism, and 3) *clinical trials*, the evaluation of the efficacy and safety of the payload *in vivo*. Though each phase contributes to the total cost of development, in this work we focus on the drug design phase. In particular we will optimize real-valued target properties (such as neurotransmitter receptor affinity or protein folding stability) for payloads represented as discrete sequences (e.g. SMILES strings). Since the mappings from sequence to target are often unknown, expensive to observe *in vitro*, and difficult to simulate, we pose biological sequence design as a costly black-box optimization (BBO) problem over a large discrete search space.

3.2 Overview

Recent work applying deep learning to biological tasks has primarily focused on learning from a static, offline dataset (Rao et al., 2019; Jumper et al., 2021; Baek et al., 2021; Meier et al., 2021; Rives et al., 2021). When these models are used to select new sequences to label, they are applied in a one-shot fashion, without accounting for future design rounds (Gligorijevic et al., 2021;

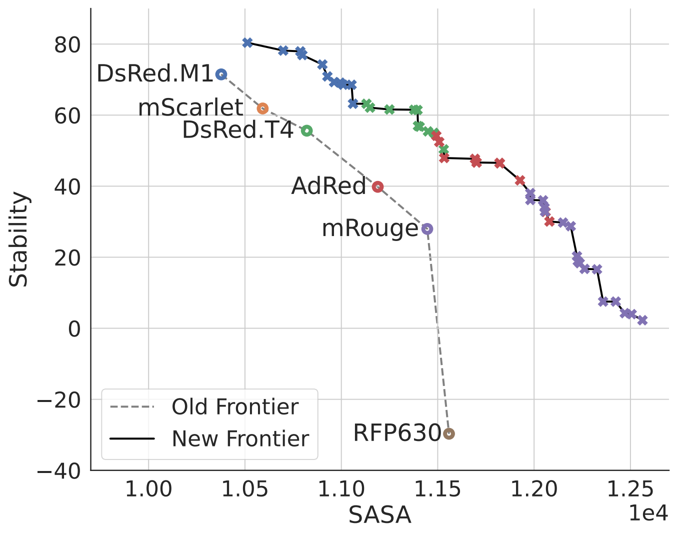


Figure 3.1: BayesOpt can be used to maximize the simulated folding stability (i.e. -dG, or the negative change in Gibbs free energy) and solvent-accessible surface area (SASA) of red-spectrum fluorescent proteins (RFPs). Higher is better for both objectives. The ancestor proteins are shown as colored circles, with corresponding optimized offspring shown as crosses. Stability correlates with protein function (e.g. how long the protein can fluoresce) while SASA is a proxy for fluorescent intensity.

Biswas et al., 2021). Because labels are scarce for many important targets, it is important to account for model uncertainty to manage the explore-exploit tradeoff (O'Donoghue et al., 2018).

Bayesian optimization (BayesOpt) is a powerful class of BBO algorithms, explicitly designed to *coherently* reason about online decision-making based on incomplete information (Brochu et al., 2010). BayesOpt balances the explore-exploit tradeoff in a principled way, relying on a probabilistic discriminative model to prioritize decisions with the highest potential payoff. At each decision point the discriminative model produces a posterior distribution over the hypothesis space of all functions the model can represent. The posterior formally represents the degree to which any particular hypothesis is plausible given the data and model assumptions. To make a decision, BayesOpt selects the best decision as defined by an *acquisition function*, such as expected improvement (EI), with each hypothesis contributing to the acquisition value in proportion to its posterior probability (Jones et al., 1998). BayesOpt is not only philosophically appealing, it is provably a *no-regret* strategy under the right conditions (Srinivas et al., 2010).

Like many Bayesian methods, the barriers hindering widespread adoption of BayesOpt are not conceptual, but *practical.* Drug design in particular is a natural application domain, but introduces multiple challenges. *Discrete, high-dimensional inputs:* antibody therapeutic payloads are often RNA proteins which instruct the patient's immune system to produce the desired antibody. When proteins are represented as a sequence of residues, each identifying one of 20 possible amino acids, even a fairly small 200-residue protein is only one of $20^{200} \approx 1.6 \times 10^{260}$ options. [*] By contrast, conventional BayesOpt works best on problems with 10 or fewer continuous decision variables, due to the properties of standard kernels, and the lack of gradients to maximize the acquisition function. *Batched, multi-objective experiments:* because considerations including efficacy, toxicity, and yield must all be taken into account, drug design is inherently multi-objective. Furthermore sequences are labeled in batches, necessitating the use of more sophisticated acquisition functions than standard workhorses like EI. *Data-scarcity:* wet lab experimental data is expensive and

[*]For a sense of scale, with an estimated 10^{80} atoms in our observable universe, enumerating 10^{260} would be like counting atoms if every atom in our universe was itself a universe, and all the atoms in those universes were also universes.

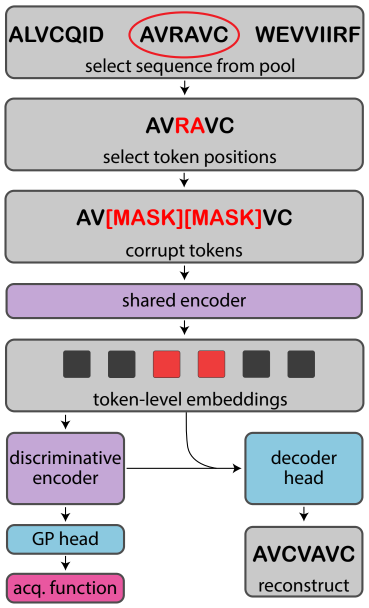


Figure 3.2: LaMBO with a non-autoregressive denoising autoencoder (DAE) architecture: a shared encoder produces continuous token-level embeddings Z from corrupted inputs, which are passed to a discriminative encoder to produce target-specific token-level embeddings Z'. A generative decoder head receives both Z and Z' as input, and a discriminative Gaussian process (GP) head pools Z' to predict the objective values. The GP head allows the use of principled acquisition functions to manage the explore-exploit tradeoff, and the DAE head allows LaMBO to follow the acquisition gradient in latent space when selecting new queries.

challenging to collect, so it is rare to have large-scale datasets with labels for the exact target properties of interest.

No single BayesOpt method has been shown to simultaneously address all of these challenges (see Section 3.3). As a result, previous methods have necessarily only been evaluated on very stylized tasks that fail to capture key aspects of real drug design problems. In this work we present **La**tent **M**ulti-Objective **B**ayes**O**pt (**LaMBO**) to address this deficiency, and propose a novel *in silico* task which emulates protein design tasks more closely than common open-source drug design benchmarks. We preview our results applying LaMBO to this new task in Figure 3.1, maximizing the stability and SASA of RFPs derived from the fpBase dataset (Lambert, 2019), with the initial Pareto frontier in objective space shown as a dashed line. The new Pareto frontier (the solid line) discovered by LaMBO is superior, as it is characterized more densely by new sequences that are Pareto improvements over their ancestors. In short, our contributions, originally published in Stanton et al. (2022), are as follows:

1. We propose the LaMBO architecture, a novel combination of a generative DAE with a discriminative deep kernel learning GP, with a simple joint training procedure and an effective decision optimization routine.

2. We propose a new *in silico* large-molecule task to augment existing open-source drug design benchmarks.

3. We evaluate LaMBO *in silico* on two small-molecule tasks and our new large-molecule task, comparing to both genetic and latent-space BBO baselines, showing improved sample efficiency and solution quality.[*]

4. We present *in vitro* wet lab results using LaMBO to discover brighter, more thermostable red fluorescent proteins.

3.3 Related Work

Discrete Optimization by Sampling: genetic algorithms (GAs) such as NSGA-II (Deb et al., 2002) slowly evolve a good solution by random mutation. GAs are a simple, popular baseline for BBO problems, but are known

[*]Code here: github.com/samuelstanton/lambo.

for being inefficient (Turner et al., 2021). One solution is to continue generating mutations randomly, but screen the proposed queries with a discriminative model before labeling (Nigam et al., 2019; Yang et al., 2019b). Other solutions focus on proposing mutations more intelligently, including RL-based approaches (Angermueller et al., 2020a;b) and generative approaches (Jensen, 2019; Biswas et al., 2021; Zhang et al., 2021). In particular the generative approach described by Lee et al. (2018) and Gligorijevic et al. (2021) inspired the LaMBO architecture. All the approaches just discussed are greedy in the sense that they rely on point estimates of the objective values to select new queries.

Discrete BayesOpt: excluding library-based approaches such as Yang et al. (2019a), discrete BayesOpt methods can be categorized by how they structure the inner loop optimization problem. Some methods use substring kernels (SSKs), optimizing queries directly in sequence space with a GA, and evaluating only on small tasks (Lodhi et al., 2002; Beck & Cohn, 2017; Moss et al., 2020). Khan et al. (2022) is an example of concurrent antibody design work in this vein, exploiting task-specific knowledge of complementarity-determining regions (CDRs) of the antibodies to make the problem tractable. See Appendix C.3.1 for more discussion and an experiment comparing SSK and DKL GPs in the offline regression setting.

Latent-space optimizers learn continuous sequence embeddings Z, which are shared by a generative decoder modeling $p(\mathbf{x}|Z)$ and the discriminative surrogate modeling $p(\mathbf{y}|Z)$ (Deshwal & Doppa, 2021; Grosnit et al., 2021;

Maus et al., 2022). Thus Z can be optimized with gradient-based methods to produce new sequences. LaMBO is most similar to Latent-Space BayesOpt (LSBO) (Gómez-Bombarelli et al., 2018), since both methods model $p(\mathbf{y}|Z)$ as a GP and model $p(\mathbf{x}|Z)$ via an autoencoder. LSBO uses a VAE pretrained on a large dataset to solve single-objective tasks, training the VAE weights and the auxiliary GP head separately. With a specialized architecture proposed by Jin et al. (2018) and a biased VAE objective proposed by Tripp et al. (2020), LSBO has been shown capable of solving simple small-molecule tasks such as maximizing penalized logP. Aside from LaMBO's use of DAEs rather than VAEs, this work improves upon LSBO in multiple ways, such as enabling the use of a general-purpose architecture for both small and large molecules, removing the need to pretrain the autoencoder, providing a reliable procedure for jointly training generative and GP heads with a shared encoder, and the introduction of multi-objective tasks and acquisition functions. See Appendix C.3.2 for a comparison between LSBO and LaMBO in the single-objective BBO setting.

Multi-Objective BayesOpt: Daulton et al. (2020a) proposed a batch version of expected hypervolume improvement (EHVI) (Emmerich, 2005; Emmerich et al., 2011), an extension of EI to multiple objectives. In follow-up work, Daulton et al. (2021a) proposed the noisy expected hypervolume improvement (NEHVI) acquisition function, which extends noisy expected improvement (NEI) (Letham et al., 2019) to multiple objectives. Multi-task GPs (MTGPs) have previously been used as surrogates for multi-objective BayesOpt

(Shah & Ghahramani, 2016), including recent work scaling MTGPs up to thousands of training examples or objectives in the continuous setting (Daulton et al., 2021b; Maddox et al., 2021a). We make use of NEHVI and MTGPs, including an efficient MTGP posterior sampling approach developed in Maddox et al. (2021b).

3.4 LaMBO: Latent-Space Multi-Objective BayesOpt

3.4.1 Discrete Multi-Objective Sequence Design

We first define the input space,

$$\mathcal{X} = \prod_{i=1}^{t} \mathcal{V},$$

where $\mathcal{V}$ is an ordered, discrete vocabulary of v tokens, t is the max sequence length, and $\prod$ is the Cartesian product. $\mathcal{V}$ includes a padding token to accommodate sequences of varying length. Because $|\mathcal{X}| = |\mathcal{V}|^t$ is exponential in t, $\mathcal{X}$ becomes too large to enumerate quickly as the sequence length grows, even if $|\mathcal{V}|$ is relatively small. As a result, sequence optimization usually starts with a library or pool of initial sequences (see Figure 3.2, top), which are modified to produce new candidate sequences. When posed in this way, the optimization problem is restructured into three nested decisions: (1) Choose a base sequence from the pool. (2) Choose which positions on the sequence to change. (3) Choose how the sequence will be changed at those positions.

We represent sequence design as a multi-objective optimization problem

$\max_{\mathbf{x}\in\mathcal{X}}(f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$, where m is the number of objectives and each $f_i : \mathcal{X} \to \mathbb{R}$ is an expensive, black-box function of decision variables $\mathbf{x} \in \mathcal{X}$. Given two feasible solutions $\mathbf{x}$ and $\mathbf{x}'$, $\mathbf{x}$ *dominates* $\mathbf{x}'$ ($\mathbf{x} \prec \mathbf{x}'$) if $f_i(\mathbf{x}) \leq f_i(\mathbf{x}')\ \forall i \in \{1, \dots, m\}$, and $\exists i \in \{1, \dots, m\}$ s.t. $f_i(\mathbf{x}) < f_i(\mathbf{x}')$. In general, there will not be a single dominating solution $\mathbf{x}^*$; instead, we define the set of non-dominated solutions (i.e. the true *Pareto frontier* $\mathcal{P}^*$),

$$\mathcal{P}^* := \{\mathbf{x} \in \mathcal{X} \mid \{\mathbf{x}' \in \mathcal{X} \mid \mathbf{x}' \prec \mathbf{x}, \mathbf{x}' \neq \mathbf{x}\} = \emptyset\}. \tag{3.1}$$

Since $\mathcal{P}^*$ is unknown, we seek a set of candidate solutions $\mathcal{P}$ that are close in objective space to those in $\mathcal{P}^*$. We find these solutions by maximizing the hypervolume bounded by the extremal points in $\mathcal{P} \cup \{\mathbf{x}_{\text{ref}}\}$, where $\mathbf{x}_{\text{ref}}$ is some reference solution. We now describe the key ideas behind LaMBO, summarized in Figure 3.2 and Algorithm 3.

3.4.2 Architecture Overview

We use a non-autoregressive denoising autoencoder to map discrete sequences to and from sequences of continuous token-level embeddings, with a multi-task GP head operating on pooled sequence-level embeddings. Unlike previous work combining GPs with deep generative models, we do not require a pretrained autoencoder, nor do we require the surrogate to be completely reinitialized after receiving new data. Furthermore, we demonstrate that both stochastic variational and exact GP inference can be used, alleviating concerns regarding computational scalability or applicability to noisy objectives with non-Gaussian likelihoods. See Appendix C.2 for more details.

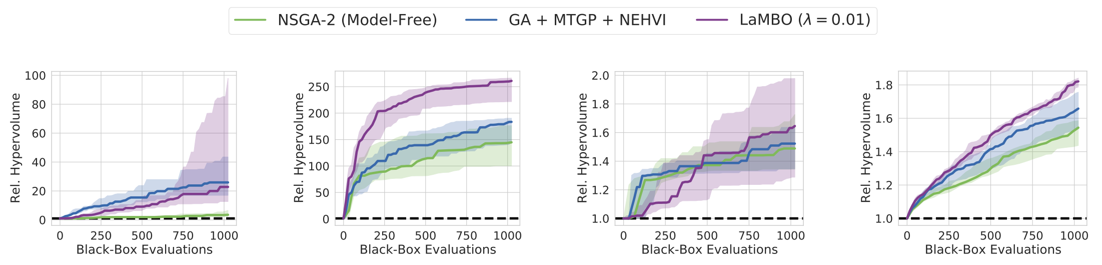
Shared Encoder: we use a non-autoregressive bidirectional encoder $g(\mathbf{x}, \theta_{\text{enc}}) = [\mathbf{z}_1, \dots, \mathbf{z}_t] = Z$, where $\mathbf{z}_i \in \mathbb{R}^d$ are latent token-level embeddings. In particular, our encoder is composed of 1D CNN layers, using standard vocabulary embeddings and sinusoidal position embeddings, padding token masking, skip-connections, and layernorm. A key advantage of using a DAE rather than a VAE is that projects like ChemBERTa, TAPE and ESM have already openly released large DAE models trained on large sequence corpora (Chithrananda et al., 2020; Rao et al., 2019; Rives et al., 2021). As a result, encoders from these models could be used as drop-in replacements for our small CNN encoder.

Discriminative Head: this head takes an encoded sequence Z and outputs a scalar value indicating the utility of selecting that sequence as a query point. We pass Z to a discriminative encoder w to obtain transformed embeddings Z', then pool Z' into low-dimensional sequence-level features. The discriminative encoder is smaller than the shared encoder, for example a single residual CNN block. The pooling operation $(|\mathcal{I}(\mathbf{x})|)^{-1} \sum_{i \in \mathcal{I}(\mathbf{x})} \mathbf{z}'_i$ averages token-level embeddings over a restricted index set $\mathcal{I}(\mathbf{x})$ which excludes positions corresponding to padding tokens. We define a multi-task GP kernel by combining a 5/2 Matérn kernel evaluated on these sequence features with an intrinsic model of coregionalization (ICM) kernel over f_i (Goovaerts et al., 1997; Alvarez et al., 2011; Rasmussen & Williams, 2008). The resulting GP outputs a posterior predictive distribution $p(f|\mathcal{D})$, which is passed as input to the acquisition function. We use the noisy expected hypervolume improvement (NEHVI) acquisition from Daulton et al. (2021a) since some objectives (particularly those involving biological data) are inherently noisy.

Generative Decoder Head: this head (h) maps a sequence of token-level embeddings to a predictive distribution over $\mathcal{X}$ and can either be a simple masked language model (MLM) head (Devlin et al., 2018) or a full seq2seq latent non-autoregressive neural machine translation (LANMT) decoder (Shu et al., 2020). In this work we make use of both. An MLM head is simpler and easier to control than an LANMT decoder, which allows for careful comparisons between LaMBO and discrete genetic optimizers. Despite their complexity, LANMT decoders accommodate insertions and deletions more gracefully than MLM heads by means of a length prediction head and length transform operation, allowing the same latent representation to be decoded to sequences of varying length. In either case, the decoder uses the same layer types as the shared encoder g, and takes both Z and Z' as input.

3.4.3 Initializing the Latent Embeddings

At each outer-loop iteration i, there are many choices of $\mathbf{x}$ we are confident will *not* improve our current solution $\mathcal{P}_i$, corresponding to large flat regions of the inner-loop loss surface. If the inner-loop is not initialized carefully, it is very likely the initial solution will fall in one of these flat regions, severely hampering gradient-based optimization. If $\mathbf{x}$ was continuous, we could use initialization heuristics from previous work to avoid this pitfall (Balandat et al.,



(a) Bigrams **(b) logP + QED** **(c) DRD3 + SA** **(d) Stab. + SASA**

Figure 3.3: On all four tasks (described in Section 3.5.1), LaMBO outperforms genetic algorithm baselines, specifically NSGA-2 and a model-based genetic optimizer with the same surrogate architecture (GA + MTGP + NEHVI). Performance is quantified by the hypervolume bounded by the optimized Pareto frontier. The midpoint, lower, and upper bounds of each curve depict the 50%, 20%, and 80% quantiles, estimated from 10 trials. See Section 3.5.2 for more discussion.

2020a; Daulton et al., 2021a). Instead we now show how the the same corruption process used to train and sample from DAEs can be used as a robust initialization procedure for discrete $\mathbf{x}$ by initializing the inner loop solution in latent-space (Z_0) with corrupted, encoded variants of the current outer-loop solution $\mathcal{P}_i$.

Selecting base sequences: we begin with a set of seed sequences X_{pool} and select a subset, $X_{\text{base}} \subset X_{\text{pool}}$. After each optimization round, the latest query sequences are added to X_{pool} and can serve as future base sequences. Choosing X_{base} well is critical for fast convergence. If too many low quality sequences are added, too much computation is spent optimizing them. Conversely, only selecting the current Pareto sequences could hinder exploration of sequences with potential for improvement. The interaction between the online queries and the decoder head must also be considered, since we want to prevent the generative samples from collapsing to a couple sequences.

We populate X_{base} first with the current Pareto sequences $\mathcal{P}_i$, then with random sequences (without replacement) that were on the Pareto frontier in previous optimization rounds ($\mathcal{P}_{<i}$), and finally fill any remaining space in the base set with random sequences from the entire optimization history. In practice, we took the size of the base set to be the same as the query batch size b. We perform multiple restarts during the inner loop, and each restart samples b sequences from X_{base}, with replacement. We do not sample any base sequences uniformly at random, but according to a weighted distribution $\Delta(X_{\text{pool}})$ to ensure that high-scoring sequences are more likely to be optimized than low-scoring ones.

Let $r_i(\mathbf{x}_j, X)$ be the rank of $\mathbf{x}_j \in X$ w.r.t. the 0-indexed dense ranking of X according to f_i, and let $r_{\max}(\mathbf{x}_j, X) = \max_i r_i(\mathbf{x}_j, X)$. The sampling weight w_j of $\mathbf{x}_j \in X_{\text{pool}}$ is

$$\begin{aligned} w_j &= s_j(-\log(1+\mathbf{r})/\tau), \\ \mathbf{r} &= [r_{\max}(\mathbf{x}_1, X), \ldots, r_{\max}(\mathbf{x}_p, X)], \end{aligned} \tag{3.2}$$

where $s_j(\mathbf{v}) = \exp(v_j)/\sum_i \exp(v_i)$ is the softmax function and $\tau \in (0, +\infty)$ is the softmax temperature. In other words, we choose the least favorable ranking across all objectives for each $\mathbf{x}$ to determine its weight. This type of weighting is similar in spirit to a procedure used by Tripp et al. (2020) to bias a VAE loss in favor of high-scoring sequences.

Selecting base sequence positions: after obtaining X_{base} we apply a

Algorithm 3: The LaMBO inner loop. For clarity, steps where LaMBO differs from LSBO are shown in blue.

```
Inputs: acquisition $a$, corruption $c$, shared encoder $g$, discriminative
  encoder $w$, decoder $h$, base sequences $X_{\text{base}}$, and batch size $b$.
$v^* \leftarrow +\infty$
$Z_0 = \{g \circ c(\mathbf{x}_0), \dots, g \circ c(\mathbf{x}_b)\},\ \mathbf{x}_m \in X_{\text{base}}$
for $j = 0, \dots, j_{\max}$ do
    $Z'_j = w(Z_j)$
    $Z_{j+1} = Z_j - \eta \nabla_Z \left[a(Z'_j) - \lambda \mathbb{H}\left[h(Z_j, Z'_j)\right]\right]$
    $X_{\text{cand}} \leftarrow \{\mathbf{x}'_1, \dots, \mathbf{x}'_b\} \sim h(Z_j, Z'_j)$
    $Z_{\text{cand}} \leftarrow [g(\mathbf{x}'_1), \dots, g(\mathbf{x}'_b)]^\top$
    $v_j = a(w(Z_{\text{cand}}))$
    if $v_j < v^*$ then
        $v^*, \mathcal{X}^* \leftarrow v_j, X_{\text{cand}}$
    end
end
return $X^*$
```

corruption function c to each element before passing them as input to the encoder, similar to the procedure proposed in Gligorijevic et al. (2021). The corruption function first selects positions in each sequence to modify, then selects a modification (substitution, insertion, deletion) at those positions. We uniformly sample positions not occupied by utility tokens, such as padding tokens.

Selecting corruption operations: once sequence positions have been selected, the corruption function chooses corruption operations to apply at those positions. For LaMBO, the corruption procedure differs depending on whether the decoder is an MLM head or a LANMT head. If the decoder is an MLM head then all operations are substitution operations, and the replacement tokens are all masking tokens. If the decoder is an LANMT head then the operation type is chosen randomly and replacement tokens are sampled uniformly from $\mathcal{V}$. Note we use a similar corruption procedure to train the decoder head.

3.4.4 Sequence Optimization

At a high level, we solve the inner-loop by treating the output of the decoder head h as a proposal distribution. We iteratively refine the proposal distribution by following a regularized acquisition gradient in latent space, drawing and scoring batches of sequences along the way (Algorithm 3).

More precisely, once we have selected and corrupted the base sequences, we pass them through the encoder to produce latent embeddings Z_0 that serve as the initial solution for the inner-loop optimization problem. Then for each optimization step $0 \leq j < j_{\max}$, we take $Z_{j+1} = Z_j - \eta \nabla_Z[\ell_{\text{query}}(Z_j)]$, where $Z'_j = w(Z_j)$ (i.e. the output of the discriminative encoder w),

$$\ell_{\text{query}}(Z_j) = a(Z'_j) + \lambda \mathbb{H}[h(Z_j, Z'_j)], \tag{3.3}$$

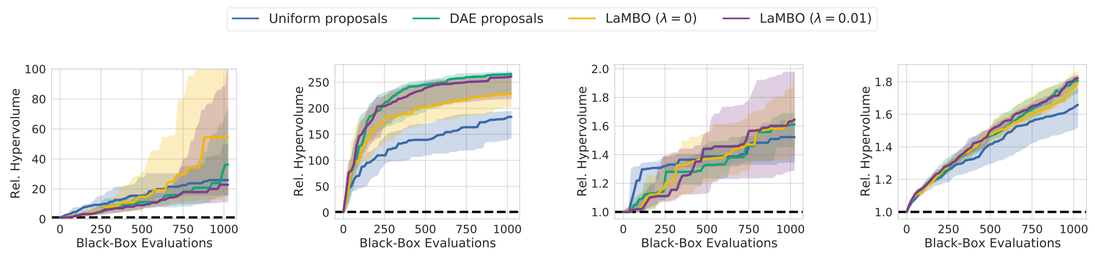
$\mathbb{H}$ is the Shannon entropy, and η, λ are hyperparameters controlling the step-size and regularization strength. We sample $\mathcal{X}_{\text{batch}} = \{\mathbf{x}'_0, \ldots, \mathbf{x}'_b\} \sim h(Z_j, Z'_j)$, and score $\mathcal{X}_{\text{batch}}$ by passing it *uncorrupted* through the shared encoder and discriminative head. If we see that X_{batch} has the best acquisition value so far, we store it and continue optimizing. Note that this procedure produces a different result than simply optimizing Z and decoding once at the end. Recall that the decoder is stochastic, and the ultimate goal of the inner loop is to produce *sequences* with high acquisition value, not just high-scoring latent embeddings.

We observed that following the unregularized acquisition gradient caused the decoder entropy to quickly increase, resulting in very uniform proposal distributions. When the proposal distributions are uniform, LaMBO essentially performs a variant of random search. However, because Z_0 is produced in the same way the decoder head is trained (i.e. the same corruption process), we expect that it will be close to other latent embeddings seen during training, and the decoder entropy will be relatively low. These observations motivated us to include the proposal entropy penalty in Eq. (3.3), to encourage Z_j to stay in a region of latent space where the decoder has non-uniform predictive distributions.

We also considered choosing a small step-size η to implicitly confine Z_j to a small region around Z_0, but we found it difficult to choose η both large enough to improve the ℓ_{query} and small enough to prevent uniform proposals. The proposal entropy penalty also has the added benefit of smoothing the query loss surface when using acquisitions like NEHVI, which helps make the inner loop less dependent on a good initialization. Adding a regularization term for improved control of the acquisition function has been previously studied in the continuous setting by Shahriari et al. (2016).

3.5 Empirical Evaluation

We now evaluate LaMBO on a suite of small-molecule and large-molecule sequence design tasks. In Section 3.5.1 describe our suite of *in silico* tasks, including a new multi-objective large-molecule task which in which we maximize



(a) Bigrams **(b) logP + QED** **(c) DRD3 + SA** **(d) Stab. + SASA**

Figure 3.4: An ablation of LaMBO's main components. Starting from our model-based genetic algorithm baseline (uniform proposals), we cumulatively add the elements described in Section 3.4.4: (1) DAE-generated proposals, (2) DAE proposal optimization following $\nabla_Z[\ell_{\text{query}}]$ with $\lambda = 0$. (see Eq. (3.3)), and (3) DAE proposal optimization with $\lambda = 0.01$. DAE proposals improve performance on all tasks. Proposal optimization is most helpful on **Bigrams** where the starting sequence distribution is very unlikely to produce high-scoring queries. The entropy penalty is detrimental when working with random sequences **(a)**, but is helpful when working with biological sequences **(b-d)**. The midpoint, lower, and upper bounds of each curve depict the 50%, 20%, and 80% quantiles, estimated from 10 trials.

the folding stability and SASA of RFPs. See Appendix C.3.1 for an experiment comparing SSK GPs and DKL GPs, and Appendix C.3.2 for an experiment comparing LSBO and LaMBO. In Section 3.5.2 we show that LaMBO outperforms strong genetic algorithm baselines in a carefully controlled comparison, followed by two investigative experiments in Section 3.5.3 which give insight into the design choices behind LaMBO. Finally in Section 3.5.4 we analyze molecules designed by LaMBO, including *in vitro* wet lab results showing the discovery of improved RFP variants.

3.5.1 Evaluation procedure

We consider the following *in silico* evaluation tasks, with full descriptions in the appendix:

- **Bigrams**: optimize short strings (around 32 tokens) uniformly sampled from $\mathcal{X}$ to maximize the counts of three predetermined bigrams (Appendix C.1.1).

- **logP + QED:** optimize SELFIES-encoded small molecules (50-100 tokens) w.r.t. logP and QED (Appendix C.1.2).

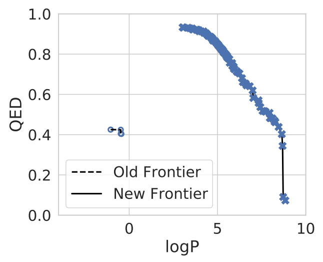
- **DRD3 docking + SA:** optimize SELFIES-encoded small molecules (50-100 tokens) w.r.t. DRD3 docking and synthetic accessibility (SA) (Appendix C.1.3).

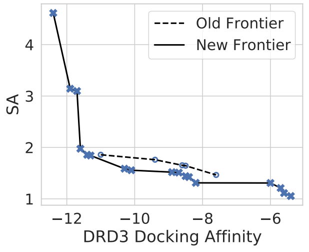
- **Stability + SASA:** optimize large-molecule RFPs (around 200 tokens) w.r.t. folding stability and SASA. Both objectives require the 3D protein structure to compute, however we treat the folding simulator as part of the black-box objective (Appendix C.1.4).

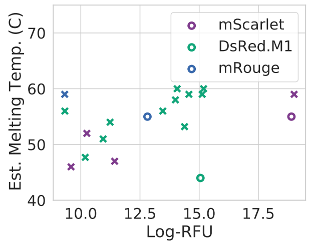
Unless otherwise noted, each task begins with 512 examples in its start pool, and collects a total of 1024 online queries in batches of 16. No additional pretraining data is used. Each method uses the same architecture and hyperparameters for all tasks (Appendix C.2). We evaluate all methods by comparing the relative improvement of the hypervolume bounded by the Pareto front after x black-box function calls compared to the starting hypervolume.

3.5.2 Comparing to Multi-Objective Genetic Optimizers

In Figure 3.3 we compare LaMBO with a MLM decoder head against two genetic algorithm (GA) baselines. For this experiment we set the entropy







(a) logP + QED **(b) DRD3 + SA** **(c) RFP Wetlab Results**

Figure 3.5: Here we show the old and new Pareto fronts in objective space discovered by LaMBO in Section 3.5.2. For **(a)** higher is better for both objectives, and for **(b)** lower is better. For all tasks every point in the old frontier is strictly dominated by at least one point in the new frontier, and solutions are evenly spread across the new frontiers. In **(c)** we provide wet lab measurements of brightness (x axis) and thermostability (y axis) for proteins optimized for **Stability + SASA**. Higher is better for both objectives. We discovered new, non-dominated protein variants for three of the five ancestor proteins in our evaluation set.

penalty at $\lambda = 0.01$. The simplest baseline is NSGA-2, a robust model-free multi-objective GA, which effectively simply randomly mutates solutions along the Pareto frontier. The other baseline is a model-based GA which also randomly mutates solutions but also screens new queries with a discriminative model, for which we use the same architecture and acquisition function (NEHVI) as LaMBO. Both GA baselines use a uniform mutation proposal distribution. In this experiment all optimizers are only allowed to change a single token per optimization round, and each optimizer selects base sequences and token positions in the same way. The model-based GA differs from LaMBO primarily in two respects: (1) the encoder is trained only through the supervised loss, and (2) the proposal distribution is uniform rather than generated by a DAE. In fact LaMBO can be viewed as a generalization of the model-based GA, where the proposal distribution is learned and optimized, rather

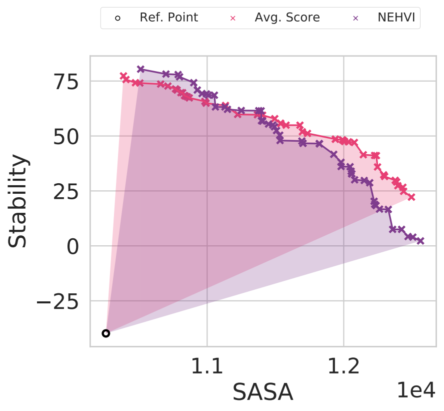


Figure 3.6: Pareto fronts of LaMBO and a variant that optimizes the expected average objective value on the RFP task. NEHVI incentivizes more exploration of the frontier extremities, resulting in solutions representing more diverse functional tradeoffs and slightly higher hypervolume.

than fixed *a priori*. The effect of (2) is most strongly seen in **logP + QED**, since the task requires very little exploration and the SELFIES vocabulary for small molecules is significantly larger than the amino acid vocabulary for proteins. LaMBO performs well on all four tasks, particularly those involving natural sequences **(b-d)**. In contrast the starting distribution over sequences in the **Bigrams** task has the highest possible entropy, so LaMBO learns a good sampling distribution more slowly.

3.5.3 Analyzing LaMBO

Having demonstrated that LaMBO compares favorably to genetic optimizers, we now examine the different components of LaMBO and how each contributes to performance. We first disentangle the effect of replacing a uniform proposal distribution with a DAE-generated one, and the effect of optimizing the proposal distribution by gradient descent on ℓ_{query}. In Figure 3.4 we interpolate between the model-based GA baseline in Figure 3.3 and LaMBO by cumulatively adding DAE-generated proposals, proposal optimization, and the proposal entropy penalty. We find that unoptimized DAE proposals with NEHVI screening is a strong baseline on all tasks. Proposal optimization is the most useful in **Bigrams**, where the true non-dominated solutions $\mathcal{P}^*$ lie far outside the starting sequence distribution, requiring more exploration. For the same reason the entropy penalty is somewhat detrimental specifically in **Bigrams**, since it keeps new queries near those previously seen. See Figure C.4 in Appendix C.3 for more discussion.

In Figure 3.6, we evaluate the sensitivity of LaMBO to the choice of acquisition function on the RFP task. We compare the final Pareto frontier obtained with a simple multi-objective scalarization (averaging normalized scores) to the frontier obtained with NEHVI. Score averaging focuses optimization on solutions with similar tradeoffs, pushing the interior of the frontier out quickly. This behavior leads to less exploration than NEHVI and thus a slightly lower hypervolume of 1.51 as compared to NEHVI's hypervolume of 1.57. Our findings corroborate similar results in previous work comparing scalarization and hypervolume-based acquisitions in different problem settings (Emmerich, 2005; Emmerich et al., 2011; Daulton et al., 2020a).

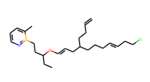
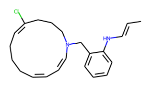
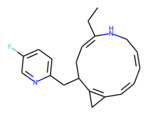
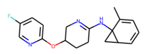
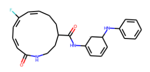
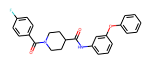
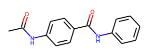
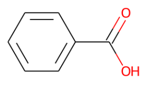
← better logP		better QED →	
			
7a.1: (8.84, 0.07)	7a.2: (6.59, 0.65)	7a-3: (4.48, 0.89)	7a.4: (3.02, 0.93)
← better docking		better SA →	
			
7b.1: (-12.4, 4.61)	7b.2: (-11.4, 1.86)	7b.3: (-8.2, 1.31)	7b.4: (-5.4, 1.05)

Figure 3.7: Example molecules at varying points on the new Pareto fronter discovered by LaMBO in Section 3.5.2 for **logP + QED** (top) and **DRD3 docking + SA** (bottom). Under each molecule we show the objective values as index : (y_1, y_2). From left to right y_1 worsens and y_2 improves. The Pareto frontier is a rich, low-dimensional space that can be analyzed much more easily than the whole search space.

3.5.4 Analyzing Sequence Designs

Though metrics like hypervolume are useful for conducting baseline comparisons and ablation studies, such metrics do not give much insight into more qualitative aspects of the sequences we are designing. In Figure 3.5(a-b) we show the Pareto fronts found by LaMBO for our two small-molecule tasks, as we did in Figure 3.1 for the large-molecule task. [*] For every task every sequence on the old frontier is strictly dominated by at least one sequence on the new frontier.

So far we have shown that LaMBO can optimize *in silico* objectives effectively, and argued in Appendix C.1.4 that the **Stability + SASA** objectives

[*]We do not visualize the Pareto frontier for Bigrams because the task has more than two objectives.

are likely correlated with properties of RFPs we actually wish to optimize in the real world, specifically brightness and thermostability. In Figure 3.5(c) we evaluate *in vitro* protein sequences designed by LaMBO to optimize **Stability + SASA**, reporting brightness (log relative fluorescence units, log-RFU) and thermostability (protein melting temperature). We discovered non-dominated variants of three of the five ancestor sequences in our evaluation set, including strictly dominant variants of mScarlet and DsRed.M1, with inconclusive results for the other two ancestor sequences. See Appendix C.1.5 for more details about our experimental procedure and results. Our results indicate that **Stability + SASA** are good *in silico* proxy objectives for a real-world protein design task.

In Figure 3.7 we visualize solutions at different points on the optimized Pareto frontiers found by LaMBO for our two small-molecule tasks. Intriguingly, we find that although we did not specifically optimize for solution diversity in sequence space, the sequences change significantly as we move along the frontiers. In contrast, many generative-only approaches use heuristics to explicitly encourage diversity in sequence space and prevent solution collapse.

Some of the proposed molecules shown have features such as large macrocycles in 7a.2 and 7b.1, or three-carbon rings in 7a.3 and 7a.4, that are likely not synthetically accessible (Gao et al., 2021). However, we note that **logP + QED** does not explicitly incentivize accessibility, and remarkably one can deduce that molecules with large macrocycles are difficult to synthesize even with little knowledge of chemistry by simply comparing molecules 7(b)-1 and

7(b)-2. Our results highlight the multi-objective nature of drug design, the need for careful objective selection (i.e. target discovery), and the human-interpretable insights that can be gained by studying the differences between non-dominated solutions found by machine learning methods.

3.6 Discussion

Drug design is quickly emerging as an extremely important application of machine learning. BayesOpt has extraordinary potential for this domain, but existing approaches struggle to extend to high-dimensional, discrete, multi-objective design tasks. We have shown how deep generative models can be integrated with BayesOpt to address these challenges, achieving good sample efficiency and solution quality across a range of design tasks. Moreover, we introduced a new large-molecule task, which provides a challenging and realistic benchmark with which to evaluate new methods for biological sequence design. Finally we successfully optimized red fluorescent proteins *in vitro*, and showed how characterizing the Pareto frontier can lead to useful, interpretable scientific insights.

In the short term, we are excited to combine LaMBO with large specialized pretrained generative models for antibodies (Ruffolo et al., 2021). The selection of mutation sites in the initialization procedure in Section 3.4.3 could also be improved beyond uniform random sampling. In the longer term, there are also many promising developments in BayesOpt methodology that have yet to be explored for sequence design, such as non-myopic acquisition functions

(Jiang et al., 2020b), multi-fidelity acquisition functions to determine the type of experimental assay and number of replications (Kandasamy et al., 2017; Wu et al., 2019), rigorous treatment of design constraints (Eriksson et al., 2019), and the coordination of many parallel drug development campaigns by optimizing risk measures across a whole compound portfolio (Cakmak et al., 2020). BayesOpt with multi-modal inputs is a particularly exciting direction, allowing scientists to combine many different sources of experimental data, including 3D structure and raw instrument output (Jin et al., 2021). BayesOpt itself can also be developed for greater resilience to model misspecification, miscalibration, and covariate shift, all of which are important in drug design tasks. Finally, our work also the highlights the need for better benchmarks to evaluate drug design methods.

While there are still many challenges to overcome, there is a path for Bayesian optimization to revolutionize drug design, profoundly improving our lives, and changing the way we approach scientific discovery.

... all the theoretical constructions ... which are used in the various branches of physics are only imperfect instruments to enable the world of empirical fact to be reconstructed in our minds.

Richard von Mises

4 Conformal Bayesian Optimization

4.1 Motivation

In the last chapter we saw how BayesOpt can be used to define a coherent strategy for active data collection in contexts like drug design. To continue this scenario, imagine this: you are a computational specialist working with an interdisciplinary team to design an antibody therapy targeting a novel antigen. Your job is to crunch data coming in from a wetlab team and return as output the specific batch of antibody sequences that should be synthesized and tested next. You are confident in your surrogate model, which scores well on random holdout validation sets, and you are confident that you are able to find sequences that maximize your expected utility.

In the past your proposed sequences have not performed well, but you have repeatedly reassured the wetlab team that the surrogate simply needs more training data. Your latest batch also performs poorly. Even worse, it is clear that the prediction sets generated by the surrogate are untrustworthy, with measurements frequently falling outside their predicted range. In the post-mortem analysis of the last batch the wetlab team wonders, understandably, why they are taking recommendations from an unreliable model. Excited to explain Bayesian decision theory, you begin, “You see, if we assume the true data generation process is in fact one of the hypotheses in our hypothesis class...” To your surprise your colleagues are not convinced. In the first place it is very clear that the true data generation process is much more complex than any of the available models. In the second place your notion of utility does not seem to place any value on reliable, observable outcomes, focusing instead on properties of an abstract model. The wetlab team calls for a halt until you demonstrate the your recommendations are based on prediction sets that reliably include actual measurements. Chastened, you end the meeting and search for a solution.

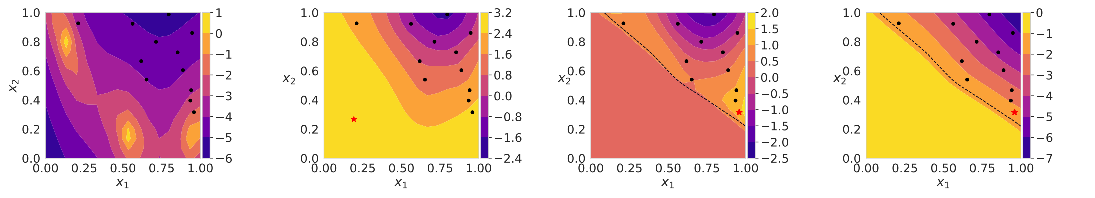
This chapter describes one such solution.

4.2 Overview

As BayesOpt agents are called upon to help guide decision-making processes with significant impact and consequences, it is critically important to guarantee that their recommendations are not just optimal under ideal conditions, but also robust and reliable under adverse conditions. Specifically, we want to ensure that our agents can methodically search high-dimensional spaces, and we want to be sure that when an outcome is predicted with high confidence, that outcome usually occurs.

4.2.1 Internal Coherence Is Not Enough

There are two main factors preventing BayesOpt agents from exhibiting these two traits. First the notion of utility encoded in commonly-used acquisition functions like upper confidence bound (UCB) causes our agents to prioritize points far from the training data, which can lead to erratic recommendations when the search space is so high-dimensional that the space of distant, unobserved points is nearly inexhaustible (Siivola et al., 2018; Wang et al., 2018; Eriksson et al., 2019). Second, our agents usually predict outcomes with Bayes β-credible sets, but in general there is no direct relationship between the subjective credibility of an outcome and the objective frequency of that outcome, a gap that is exacerbated by erroneous modeling assumptions (Grünwald & Van Ommen, 2017). Unlike credible sets, frequentist prediction sets can be designed such that the prediction confidence directly corresponds to the expected coverage, i.e. the frequency that the outcome falls within the predic-



(a) Objective fn. **(b)** UCB. **(c)** CUCB. **(d)** Log imp. weights

Figure 4.1: A motivating example of feedback covariate shift. We want $\mathbf{x}^* \in [0,1]^2$ which maximizes the Branin objective **(a)**, starting from 12 examples in the upper right (the black dots). An acquisition function like UCB **(b)** selects the next query (the red star) far from any training data, where we cannot guarantee reliable predictions. Given a miscoverage tolerance α, conformal UCB **(c)** directs the search to the region where conformal predictions are guaranteed coverage of at least $(1-\alpha)$. The boundary of the search space (the dashed black line) is the set of $\mathbf{x}$ with importance weights **(d)** exactly equal to α. For each $\mathbf{x}$ the importance weight is proportional to the ratio of the likelihood of selecting $\mathbf{x}$ as the next query point to the likelihood of drawing $\mathbf{x}$ from the train distribution.

tion set (Wasserman, 2008).

4.2.2 Bayesian and Frequentist Methods Are Complementary

Conformal prediction sets in particular come with coverage guarantees that hold even if the underlying model is misspecified (Vovk et al., 2005). Furthermore conformal prediction has a simple mechanism to correct for covariate shift, which occurs when we systematically bias our queries (e.g. towards regions of search space with high acquisition value). Once we recognize the tendency of BayesOpt agents to push test points away from train data as a phenomenon known as feedback covariate shift (FCS), we can see that conformal prediction offers an elegant solution (Fannjiang et al., 2022). The biggest drawback of conformal prediction is it cannot easily be used to reason about epistemic uncertainty, which is crucial to manage explore-exploit tradeoffs.

4.2.3 Key Ideas and Contributions

We now present conformal Bayesian optimization, a method that is both Bayesian and frequentist, with a motivating example in Figure 4.1. Conformal BayesOpt adjusts how far new queries want to move from the train data simply by choosing an acceptable miscoverage tolerance $\alpha \in (0, 1]$. If $\alpha = 1$ then we recover conventional BayesOpt, but if $\alpha < 1$ then the search will be directed to the region where conformal predictions are guaranteed coverage of at least $(1 - \alpha)$. Additionally conformal BayesOpt predicts outcomes via conformal Bayes prediction sets with coverage guarantees that are robust to model misspecification. In summary the primary contributions of this chapter are as follows:

- Conformal versions of common BayesOpt acquisition functions to direct queries towards regions of the search space where the surrogate predictions are valid.
- An efficient, differentiable implementation of full conformal Bayes for GP regression models allowing queries to be optimized with gradient-based methods, and a practical estimation procedure for the resulting implicit density ratio.
- Demonstrations on synthetic black-box optimization tasks and real ranking tasks that conformal BayesOpt has comparable sample-efficiency to baselines, while improving query coverage significantly.

4.3 Limitations of Bayesian Inference

As discussed in Chapter 1.4 we consider optimization problems of the form $\max_{\mathbf{x} \in \mathcal{X}}(f_1^*(\mathbf{x}), \ldots, f_m^*(\mathbf{x}))$, where each $f_i^* : \mathcal{X} \to \mathbb{R}$ is an expensive, black-box function of decision variables $\mathbf{x} \in \mathcal{X}$, and m is the number of objectives. We typically assume that we do not observe f^* directly, but instead receive noisy labels $y \in \mathcal{Y}$ according to some likelihood $p^*(y|f)$.

4.3.1 Common Examples of Model Misspecification

In order to do Bayesian inference at all, we must assume that our prior $p(f)$ actually supports f^*. For example, in BayesOpt it is very common to choose a stationary kernel κ (such as a Matérn kernel) and assume $f^* \sim GP(0, \kappa)$. Stationary kernels are translation-invariant and encode a strong bias towards uniformly smooth functions, but in practice we may encounter objectives that are smooth in some regions of $\mathcal{X}$, and rough in others. Furthermore, we typically do not know the true likelihood $p^*(y|f)$, and often choose $p(y|f) = \mathcal{N}(f, \sigma^2 I_m)$ for convenience. In reality the true noise process may be correlated with $\mathbf{x}$, across objectives, or may not be Gaussian at all (Assael et al., 2014; Griffiths et al., 2019; Makarova et al., 2021). Circumstances like these can cause Bayes β-credible prediction sets (defined in Chapter 1.2) to correspond very poorly with the external world.

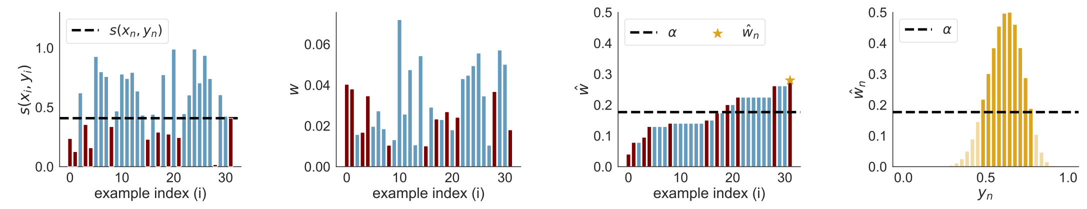
In particular, β-credible sets may exhibit poor coverage, meaning the frequency of "implausible" events outside the set happening is much more than

$1 - \beta$ (Bachoc, 2013).[*] Nevertheless, incorrect modeling assumptions do not make Bayesian inference useless. Bayesian decision theory is very practical compared to frequentist ideas like u-admissable decision rules, which are also affected by modeling assumptions and can involve odd reasoning about data that we could have seen but did not (Berger, 2013).

4.3.2 Ending the Search for the Perfect Model

Not only are incorrect modeling assumptions virtually inevitable, they often have significant practical benefits. Continuing our previous examples, f^* may be just smooth enough that a BayesOpt agent using a smooth stationary kernel (e.g. RBF) performs better than one using a kernel with weaker inductive biases. Indeed, theoretical convergence rates for acquisition functions like UCB suggest that for BayesOpt we want to use the *smoothest possible* model, subject to the constraint that we can still model f^* sufficiently well (Srinivas et al., 2010). Similarly isotropic Gaussian likelihoods have significant computational advantages, and there is no guarantee that constructing a task-specific likelihood for every optimization problem would be especially useful. The solution is not to blind ourselves to the error in our assumptions, nor is it to paralyze ourselves in pursuit of a perfect model. Instead we should develop methods that can gracefully accommodate imperfect models, balancing internal coherence with external validity. Conformal prediction is particularly appealing in this regard, since it can reliably produce externally valid predictions, even

[*]We emphasize that this does not necessarily imply that either $p(f|D)$ or the credible set were computed incorrectly, it may simply indicate a poor modeling assumption.



(a) conformal scores **(b)** imp. weights **(c)** IW partial sums **(d)** prediction set

Figure 4.2: Constructing a conformal prediction set $C_\alpha(\mathbf{x}_n)$ in the regression setting. First, **(a)** we choose some $\mathbf{y}' \in \mathcal{Y}$ and guess $\mathbf{y}_n = \mathbf{y}'$, computing conformal scores on the train examples $(X_{\text{seen}}, Y_{\text{seen}})$ and the guessed example $(\mathbf{x}_n, \mathbf{y}_n)$, **(b)** we note which examples score better than our guess (shown in blue), and mask out the corresponding importance weights. **(c)** we compute the partial sums $\hat{\mathbf{w}}_i$ of the masked importance weights, adding $\mathbf{y}'$ to $C_\alpha(\mathbf{x}_n)$ if $\hat{\mathbf{w}}_n > \alpha$, **(d)** repeat steps **(a - c)** for many guesses of $\mathbf{y}_n$. Accepted and rejected guesses are shaded dark and light, respectively.

with a misspecified model.

4.4 Conformal Prediction

See Shafer & Vovk (2008) for a complete tutorial on conformal prediction, or Angelopoulos & Bates (2021) for a modern, accessible introduction. Vovk et al. (2005) is also an excellent reference.

4.4.1 What is Conformal Prediction?

Informally, a conformal prediction set $C_\alpha(\mathbf{x}) \subset \mathcal{Y}$ is a set of possible labels for a test point $\mathbf{x}_n$. Candidate labels $\mathbf{y}'$ are included in $C_\alpha(\mathbf{x})$ if the resulting pair $(\mathbf{x}_n, \mathbf{y}')$ is sufficiently similar to actual examples seen in the past. The degree of similarity is measured by a score function s and importance weights (IWs) $\mathbf{w}$, and the similarity threshold is determined by the miscoverage tolerance α. In Figure 4.2 we visualize the process of constructing $C_\alpha(\mathbf{x})$. More formally, Definition 4.4.1. Let $\alpha \in (0, 1]$, $(\mathbf{x}_0, \mathbf{y}_0), \dots, (\mathbf{x}_{n-1}, \mathbf{y}_{n-1}) \sim p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$ and $(\mathbf{x}_n, \mathbf{y}_n) \sim p'(\mathbf{x}, \mathbf{y}) = p'(\mathbf{x})p(\mathbf{y}|\mathbf{x})$, with $\mathbf{w}_i \propto p'(\mathbf{x}_i)/p(\mathbf{x}_i)$ s.t. $\sum_k \mathbf{w}_k = 1$. The conformal prediction set corresponding to the score function $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is defined as

$$\mathcal{C}_\alpha(\mathbf{x}_n) := \left\{ \mathbf{y}' \in \mathcal{Y} \;\middle|\; \sum_{i=0}^{n} \mathbb{1}\left\{ s(\mathbf{x}_i, \mathbf{y}_i) \leq s(\mathbf{x}_n, \mathbf{y}') \right\} \mathbf{w}_i > \alpha \right\}. \tag{4.1}$$

In the special case where $(\mathbf{x}_0, \mathbf{y}_0), \dots, (\mathbf{x}_n, \mathbf{y}_n)$ are fully exchangeable (e.g. IID), then $\mathbf{w}_i = 1/(n+1), \ \forall i$. We use importance weighting in order to correct for covariate shift (Tibshirani et al., 2019). Conformal prediction enjoys a frequentist marginal coverage guarantee on $\mathcal{C}_\alpha(\mathbf{x}_n)$ with respect to the joint distribution over $(\mathbf{x}_0, \mathbf{y}_0), \dots, (\mathbf{x}_n, \mathbf{y}_n)$,

$$\mathbb{P}[\mathbf{y}_n \in \mathcal{C}_\alpha(\mathbf{x}_n)] \geq 1 - \alpha, \tag{4.2}$$

meaning if we repeatedly draw n training examples from $p(\mathbf{x}, \mathbf{y})$, and draw a single test example $(\mathbf{x}_n, \mathbf{y}_n)$ from $p'(\mathbf{x}, \mathbf{y})$, $\mathcal{C}_\alpha(\mathbf{x}_n)$ will contain the observed label $\mathbf{y}_n$ at least $(100 \times (1 - \alpha))\%$ of the time. A prediction set with a coverage guarantee like Eq. (4.2) is said to be *conservatively valid* at the $1 - \alpha$ level.

4.4.2 What Assumptions Does the Coverage Guarantee Require?

The assumptions underlying the coverage guarantee for conformal prediction are strikingly mild. Specifically we must assume $(\mathbf{x}_0, \mathbf{y}_0), \dots, (\mathbf{x}_n, \mathbf{y}_n)$ are pseudo-exchangeable. Note that although every IID sequence of random variables is exchangeable, not every exchangeable sequence is IID. Similarly pseudo-exchangeability is not "IID-lite", where every element of the sequence except for the last is IID. A sequence is pseudo-exchangeable if the joint density can be factored into terms that only depend on the values of the sequence, not the ordering (Fannjiang et al., 2022). Informally, we can see that BayesOpt satisfies pseudo-exchangeability because the distribution over training data is just the mixture of all the previous query distributions. Furthermore once queries are labeled and added to the training data, they can be shuffled at will without changing the next query distribution.

Any real-valued, measurable score function will produce a valid prediction set (Vovk et al., 2005). There are trivial examples that produce trivially valid prediction sets $\mathcal{C}_\alpha(\mathbf{x}) = \mathcal{Y},\ \forall \mathbf{x},\ \forall \alpha$. In general with conformal prediction if we choose s poorly we pay a price in terms of *efficiency* (i.e. the volume of the prediction sets), but validity is still maintained. In other words, the conformal prediction coverage guarantee *does not* assume the surrogate model is correctly specified, nor does it require strong smoothness assumptions on f^*.

4.4.3 Full Conformal Bayes

Choosing the following score function:

$$s(\mathbf{x}_i, \mathbf{y}_i) = \log p(\mathbf{y}_i | \mathbf{x}_i, D \cup \{(\mathbf{x}_n, \mathbf{y}')\}),$$

corresponds to full conformal Bayes prediction (Fong & Holmes, 2021). Conditioning an existing posterior $p(\mathbf{y}|\mathbf{x}_i, D)$ on an additional observation $(\mathbf{x}_n, \mathbf{y}_j)$ is commonly referred to as "retraining" in the conformal prediction literature. We discuss efficient computation of full conformal Bayes in Section 4.6.2. *If* the surrogate just so happens to be correctly specified, then full conformal Bayes is the optimal choice of score function, meaning it provides the most efficient prediction sets (i.e. smallest by expected volume w.r.t. the prior $p(f)$) among all prediction sets that are valid at the $1 - \alpha$ level (Hoff, 2021). In the typical situation where we think the surrogate is plausible but do not really believe it, full conformal Bayes rewards us if the surrogate turns out to be right, yet in all cases it produces valid predictions — even if the surrogate is wrong.

4.5 Related Work

4.5.1 Bayesian Inference and Model Misspecification

Van Der Vaart & Van Zanten (2011) and Wynne et al. (2021) give bounds on the estimation of the GP's posterior mean and variances in the presence of misspecification, while Zaytsev et al. (2018) and others (Beckers et al., 2018; Fiedler et al., 2021) upper bound the estimation error between f^* and the GP predictive mean when the prior covariance is incorrect. Neiswanger & Ramdas (2021) proposed frequentist confidence intervals guaranteed to contain f^* even under a misspecified GP prior. Our work also combines Bayesian and frequentist methods, however we focus more on the decision-making aspect of

BayesOpt, rather than the inference subproblem.

4.5.2 Robust BayesOpt

We do not consider corruptions to the queries during execution, see Kirschner et al. (2020); Fröhlich et al. (2020); Daulton et al. (2022) for work in that vein. Bogunovic & Krause (2021) is more closely related, proposing a variant of UCB which can adjust to a misspecified GP prior. Similarly Makarova et al. (2021) propose a risk-averse variant of UCB to accomodate a misspecified likelihood. In contrast, we do not make any assumptions regarding exactly how our model is misspecified, and we show how to apply our solution to many different acquisition functions.

4.5.3 Conformal Prediction

Our work is most closely related to Fannjiang et al. (2022), who propose a black-box optimization method based on conformal prediction specifically to address feedback covariate shift. However, because they assume new queries are drawn from a known, closed-form density, and because exact conformal prediction is not differentiable, their approach cannot be easily extended to most BayesOpt methods.[*] Bai et al. (2022) propose a differentiable approximation of conformal prediction, but it requires solving a minimax optimization subproblem. Stutz et al. (2021) independently proposed a continuous relaxation of conformal prediction, like our work, but only for fully exchangeable

[*]For example, BoTorch relies heavily on L-BFGS-B to optimize acquisition functions (Balandat et al., 2020a).

data. We propose a more general form that allows for covariate shift, and we also show how to estimate the importance weights when the queries are drawn from an implicit density.

4.6 Conformal Bayesian Optimization

We now share the key ideas behind conformal BayesOpt. First in Section 4.6.1 we describe how conformal prediction sets can be incorporated into common BayesOpt acquisition functions such as EI, next in Section 4.6.2 we discuss how to compute the required integrals efficiently and differentiably, and finally in Section 4.6.3 we show how to correct for covariate shift. In Appendix D.3.1 we provide a detailed overview explicitly showing how all the key pieces fit together.

4.6.1 Conformal acquisition functions

By the sum rule of probability, we can rewrite the Bayes posterior $p(f(\mathbf{x})|D)$ as an integral over all possible outcomes $\mathbf{y}|\mathbf{x}$,

$$p(f(\mathbf{x})|D) = \int_{\mathbf{y}\in\mathcal{Y}} p(f(\mathbf{x})|D \cup \{(\mathbf{x}, \mathbf{y})\})p(\mathbf{y}|\mathbf{x}, D)d\mathbf{y}. \tag{4.3}$$

Eq. (4.3) states that we can obtain $p(f(\mathbf{x})|D)$ by marginalizing the joint posterior $p(f(\mathbf{x}), \mathbf{y}|\mathbf{x}, D)$ w.r.t. $\mathbf{y}$. $p(f(\mathbf{x})|D)$ can also be seen as a Bayesian model average, where we condition each component model on a different potential observation $(\mathbf{x}, \mathbf{y})$, and weight the components by $p(\mathbf{y}|\mathbf{x}, D)$.

Note that we are free to change the component weights to any other valid distribution over $\mathbf{y}$ we like. Choosing the Bayes posterior $p(\mathbf{y}|\mathbf{x}, D)$ will result in a Bayes-optimal policy w.r.t. our utility function u, but the value of Bayes-optimality is questionable if we do not really believe the surrogate. If the surrogate is misspecified, peaks in $p(\mathbf{y}|\mathbf{x}, D)$ may not correspond to the most likely outcome at all, degrading the quality of our queries. Here we introduce the conformal Bayes predictive posterior $p_\alpha(\mathbf{y}|\mathbf{x}, D)$, defined as

$$p_\alpha(\mathbf{y}|\mathbf{x}, D) := \begin{cases} (1-\alpha)/Z_1 & \text{if } \mathbf{y} \in C_\alpha(\mathbf{x}), \\ \alpha p(\mathbf{y}|\mathbf{x}, D)/Z_2 & \text{else,} \end{cases}$$

where Z_1, Z_2 are normalization constants.

We are partitioning the outcome space into two events, either $\mathbf{y} \in C_\alpha(\mathbf{x})$ or it is not. We know that $\mathbf{y} \in C_\alpha(\mathbf{x})$ with frequency $(1-\alpha)$, since $C_\alpha(\mathbf{x})$ is a valid prediction set, and we do not consider any particular $\mathbf{y} \in C_\alpha(\mathbf{x})$ to be more likely than another, since the coverage guarantee holds for $C_\alpha(\mathbf{x})$ as a whole. [*] We also expect that $\mathbf{y} \notin C_\alpha(\mathbf{x})$ with frequency α, and we weight each $\mathbf{y} \notin C_\alpha(\mathbf{x})$ by $p(\mathbf{y}|\mathbf{x}, D)$ to form a proper density (i.e. a density that integrates to 1). See

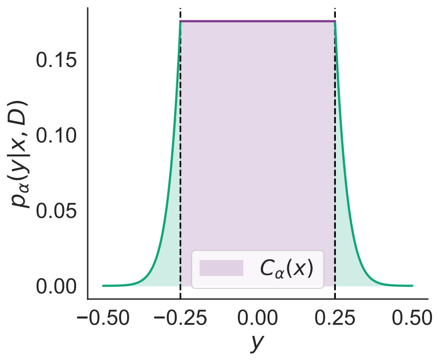


Figure 4.3: An illustration of $p_\alpha(\mathbf{y}|\mathbf{x}, D)$.

[*]We could instead use a conformal predictive density here (Vovk et al., 2017; Marx et al., 2022)

Figure 4.3 for an illustration.

If we had noiseless observations (i.e. $\mathbf{y}_i = f(\mathbf{x}_i)$), we could use $p_\alpha(\mathbf{y}|\mathbf{x}, D)$ directly when computing the acquisition value of new queries. However managing the explore-exploit tradeoff with noisy outcomes requires us to distinguish between reducible and irreducible (i.e. epistemic and aleatoric) uncertainty. If we do not, optimistic acquisition functions like UCB may direct us towards queries whose outcomes are very noisy.

Substituting $p_\alpha(\mathbf{y}|\mathbf{x}, D)$ for $p(\mathbf{y}|\mathbf{x}, D)$ in Eq. (4.3) results in the conformal Bayes posterior $p_\alpha(f(\mathbf{x})|D)$,

$$\begin{aligned} p_\alpha(f(\mathbf{x})|D) := \frac{1-\alpha}{Z_1} &\int_{C_\alpha(\mathbf{x})} p(f(\mathbf{x})|D \cup \{(\mathbf{x}, \mathbf{y})\}) d\mathbf{y} \\ + \frac{\alpha}{Z_2} &\int_{\mathcal{Y} - C_\alpha(\mathbf{x})} p(f(\mathbf{x})|D \cup \{(\mathbf{x}, \mathbf{y})\}) p(\mathbf{y}|\mathbf{x}, D) d\mathbf{y}. \end{aligned} \tag{4.4}$$

In Appendix D.2.1 we show that $p_\alpha(f|D)$ converges pointwise to $p(f|D)$ as $\alpha \to 1$.

Now that we have $p_\alpha(f(\mathbf{x})|D)$, we can "conformalize" any acquisition function written in the form of Eq. (1.5) by substituting $p_\alpha(f(\mathbf{x})|D)$ for $p(f(\mathbf{x})|D)$. For example, as noted in Section 1.4, we can take $u(f(\mathbf{x}), D) = [f(\mathbf{x}) - \max_{\mathbf{y}_i \in D} \mathbf{y}_i]_+$ and computed the expectation w.r.t. $p(f(\mathbf{x})|D)$ to obtain the expected improvement (EI) acquisition function. Similarly we write conformal EI (CEI) as,

$$\mathrm{CEI}_\alpha(\mathbf{x}) = \int [f(\mathbf{x}) - \max_{\mathbf{y}_i \in D} \mathbf{y}_i]_+ p_\alpha(f(\mathbf{x})|D) df(\mathbf{x}). \tag{4.5}$$

In Appendix D.2.3 we derive conformal variants of other popular acquisition functions, including ones for batched queries and multiple objectives.

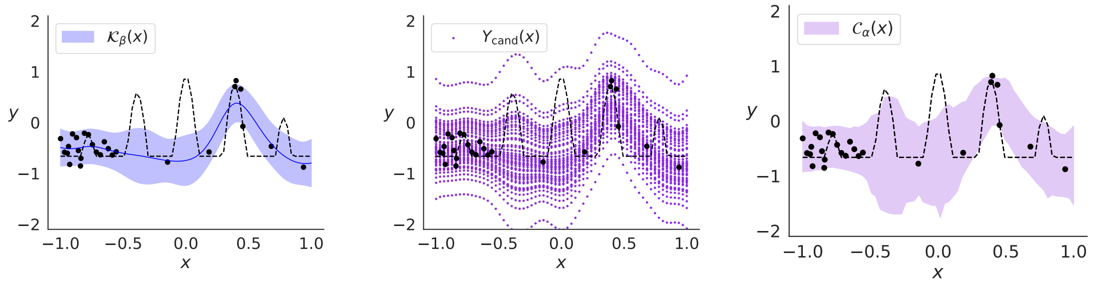
4.6.2 Efficient differentiable full conformal prediction with GPs

Efficient retraining: in the regression setting, we must compute

$$s(\mathbf{x}_i, \mathbf{y}_i) = \log p(\mathbf{y}_i | \mathbf{x}_i, D \cup \{(\mathbf{x}_n, \mathbf{y}_j)\}),\ \forall i \in \{0, \dots, n\},\ \mathbf{y}_j \in Y_{\text{cand}},$$

where Y_{cand} is some discretization of $\mathcal{Y}$. As we discussed at length in Chapter 2, this sort of incremental posterior update can be done very efficiently if the surrogate is a GP regression model, and we will later reuse the conditioned posteriors to estimate expectations w.r.t. $p_\alpha(f(\mathbf{x})|D)$. Note that computing the GP posterior likelihood of training data can be numerically unstable, see Appendix D.3 for a stable approach. By contrast, other Bayesian predictive posteriors (e.g. from Bayesian neural networks) are conditioned on training data via iterative methods such as gradient descent, making computing the full conformal scores very expensive.

Efficient discretization of $\mathcal{Y}$: since we can efficiently compute the full conformal Bayes score function, we now consider how to choose Y_{cand}. In the simple case of a single objective with small query batches ($q \leq 2$), then Y_{cand} can be a regularly sampled dense grid, and y can be integrated out with numerical quadrature. However dense grids are inefficient since they must be wide enough to capture all possible values of $\mathbf{y}$ and dense enough to pinpoint the boundary of $\mathcal{C}_\alpha(\mathbf{x})$ with reasonable accuracy. Here we observe that even if



(a) **(b)** **(c)**

Figure 4.4: Here we should how to construct full conformal Bayes prediction sets starting with a Bayes posterior $p(\mathbf{y}|\mathbf{x}, D)$. In this example D is composed of $n = 27$ noisy observations (shown as black dots) of the true objective (shown as a black dashed line) and $\alpha = \beta \approx 0.19$. In panel **(a)** we show $\mathcal{K}_\beta(\mathbf{x})$, the β-credible prediction set. In panel **(b)** we show Y_{cand} populated by samples from $p(\mathbf{y}|\mathbf{x}, D)$. In panel **(c)** we show $\mathcal{C}_\alpha(\mathbf{x})$, the conformal prediction set. The coverage of $\mathcal{C}_\alpha(\mathbf{x})$ is noticeably better than $\mathcal{K}_\beta(\mathbf{x})$ in regions where there is little training data, even though the nominal level of confidence is the same.

we do not fully believe $p(\mathbf{y}|\mathbf{x}, D)$, it is still our best guess of where $\mathbf{y}|\mathbf{x}$ should be, so instead of a dense grid we populate Y_{cand} with proposals $\mathbf{y}_j \sim p(\mathbf{y}|\mathbf{x}, D)$ (Figure 4.4).* This approach not only reduces computational effort for low-dimensional outcomes, but also allows us to extend to multi-objective tasks.

Instead of numerical quadrature, we can use importance-weighted Monte Carlo estimation to approximate integrals involving $\mathbf{y}$ as follows:

$$\int g(\mathbf{y})d\mathbf{y} \approx \frac{1}{|Y_{\text{cand}}|} \sum_{\mathbf{y}_j \in Y_{\text{cand}}} \frac{1}{p(\mathbf{y}_j|\mathbf{x}, D)} g(\mathbf{y}_j). \tag{4.6}$$

Differentiable conformal prediction: Next, we consider how to approximately compute full conformal Bayes prediction sets differentiably. First, the definition of $\mathcal{C}_\alpha(\mathbf{x}_n)$ in Eq. (4.1) can be broken down into a sequence of simple

*For small values of α we have found that doubling the covariance $p(\mathbf{y}|\mathbf{x}_n, D)$ can help ensure that the grid is sufficiently wide.

Algorithm 4: Computing differentiable conformal prediction masks

```
Data: train data D = $\{(\mathbf{x}_i, y_i)\}_{i=0}^{n-1}$, test point $\mathbf{x}_n$, importance weights $\mathbf{w}$,
      label candidates $Y_{\text{cand}}$, score function $s$, miscoverage tolerance $\alpha$,
      relaxation strength $\tau$.

$\mathbf{m}_j = 0$, $\forall j \in \{0, \ldots, |Y_{\text{cand}}| - 1\}$.      (initialize conformal prediction mask)
for $y_j \in Y_{\text{cand}}$ do
    $\mathbf{s}_j = [s(\mathbf{x}_0, y_0) \ \cdots \ s(\mathbf{x}_n, y_j)]^\top$.                (compute conformal scores)
    $\mathbf{h}_j = \texttt{sigmoid}((\mathbf{s}_j - \mathbf{s}_{jn})/\tau)$.                  (compute weight mask)
    $\hat{\mathbf{w}}_j = \mathbf{1}^\top(\mathbf{h}_j \odot \mathbf{w})$.                      (sum masked importance weights)
    $\mathbf{m}_j \leftarrow \texttt{sigmoid}((\hat{\mathbf{w}}_j - \alpha)/\tau)$.              (update prediction mask)
end

Result: $\mathbf{m}$
```

vector operations interspersed with Heaviside functions. The Heaviside function is piecewise constant, with ill-defined derivatives, so we replace it with its continuous relaxation the sigmoid function. Our procedure is summarized in Algorithm 4. Informally, the output $\mathbf{m}_j$ of the final sigmoid can be interpreted as the *probabilility* of accepting some $\mathbf{y}_j$ into $\mathcal{C}_\alpha(\mathbf{x}_n)$. The smoothness of the relaxation is controlled by a single hyperparameter $\tau \in (0, +\infty)$. As $\tau \to 0$ the relaxation becomes exact but the gradients become very poorly behaved.

Once we have the prediction mask $\mathbf{m}$ we estimate unweighted integrals of g over $\mathcal{C}_\alpha(\mathbf{x})$ as

$$\int_{\mathcal{C}_\alpha(\mathbf{x})} g(\mathbf{y})d\mathbf{y} \approx \frac{1}{\mathbf{1}^\top \mathbf{m}} \sum_{j=0}^{k-1} \frac{\mathbf{m}_j}{p(\mathbf{y}_j|\mathbf{x}, D)} g(\mathbf{y}_j),$$

and estimate weighted integrals over the complement $\mathcal{Y} - \mathcal{C}_\alpha(\mathbf{x})$ as

$$\int_{\mathcal{Y}-\mathcal{C}_\alpha(\mathbf{x})} g(\mathbf{y}) p(\mathbf{y}|\mathbf{x}, D) d\mathbf{y} \approx \frac{1}{\mathbf{1}^\top(\mathbf{1} - \mathbf{m})} \sum_{j=0}^{k-1} (1 - \mathbf{m}_j) g(\mathbf{y}_j).$$

Bringing everything together, we have differentiable Monte Carlo estimates of $\mathrm{CEI}_\alpha(\mathbf{x})$,

$$\mathrm{CEI}_\alpha(\mathbf{x}) \approx (1-\alpha)\mathbf{v}_0^\top \mathbf{u} + \alpha \mathbf{v}_1^\top \mathbf{u}, \qquad \mathbf{u} = [u(\mathbf{b}_0, D), \ldots, u(\mathbf{b}_{k-1}, D)]^\top,$$

$$(\mathbf{v}_0)_i = \frac{\mathbf{m}_i}{p(\mathbf{y}_i|\mathbf{x})} \left(\sum_j \frac{\mathbf{m}_j}{p(\mathbf{y}_j|\mathbf{x})} \right)^{-1}, \quad (\mathbf{v}_1)_i = (1 - \mathbf{m}_i) \left(\sum_j (1 - \mathbf{m}_j) \right)^{-1}, \tag{4.7}$$

where $\mathbf{b}_j \sim p(f(\mathbf{x})|D \cup \{(\mathbf{x}, \mathbf{y}_j)\})$. See Appendix D.2.2 for more details.

4.6.3 Accounting for Feedback Covariate Shift

If we were merely ranking queries that were exchangeable with D, then there would be no need to correct for covariate shift. However, the explicit goal of applications like drug discovery is to discover *novel* candidates with exceptional attributes (i.e. out of distributions observations relative to the training data). In that context, the more successfully we identify unexplored, promising regions of $\mathcal{X}$, the more severe we can expect the covariate shift between our training data and our "test" data to be.

Density ratio estimation: as we saw in Section 4.4, adapting $\mathcal{C}_\alpha(\mathbf{x})$ to covariate shift requires estimating importance weights $\mathbf{w}_i \propto r(\mathbf{x}_i) = p'(\mathbf{x}_i)/p(\mathbf{x}_i)$, where $p'(\mathbf{x})$ is the "test" distribution from which we draw candidate query points. If we know the densities $p(\mathbf{x})$ and $p(\mathbf{x}')$ the we can compute $r(\mathbf{x})$ very easily, but in general we only have samples from $p(\mathbf{x})$. Furthermore if we wish to optimize queries with gradient based methods then $p'(\mathbf{x})$ is implicitly defined as the distribution over $\mathbf{x}_t$ iterates induced by gradient ascent on $a(\mathbf{x})$ from some initial distribution on $\mathbf{x}_0$. Fortunately we can still obtain samples from $p'(\mathbf{x})$ by sampling from the energy distribution, $p'(\mathbf{x}) \propto \exp\{a(\mathbf{x})\}$ via stochastic gradient Langevin dynamics (SGLD) (Welling & Teh, 2011).

Once we have samples from $p(\mathbf{x})$ (which are already in D) and $p'(\mathbf{x})$, we estimate $r(\mathbf{x})$ with a probabilistic classifier (Sugiyama et al., 2012). We assign labels 1 to samples from $p'(\mathbf{x})$ and 0 to samples from $p(\mathbf{x})$, defining new conditional distributions

$$p'(\mathbf{x}) = p(\mathbf{x} \mid z = 1) \text{ and } p(\mathbf{x}) = p(\mathbf{x} \mid z = 0).$$

By Bayes theorem, we rewrite $r(\mathbf{x})$,

$$r(\mathbf{x}) = \frac{p(z=0)}{p(z=1)} \frac{p(z=1 \mid \mathbf{x})}{p(z=0 \mid \mathbf{x})}, \tag{4.8}$$

such that we need only train a probabilistic classifier $\hat{p}(z \mid \mathbf{x})$ to discriminate the sample labels. We estimate the prior ratio $p(z=0)/p(z=1)$ empirically.

Which comes first, the acquisition function or the ratio estimator? To estimate r as just described we clearly must be able compute gradients of a to draw the required samples from $p'(\mathbf{x}) \propto \exp\{a(\mathbf{x})\}$. However taking $a = \mathrm{CEI}_\alpha$ (or another conformal acquisition) introduces a second and more serious issue, since a itself then depends on r. We need an estimator $\hat{r}$ that simultaneously induces $p'(\mathbf{x}) \propto \exp\{\mathrm{CEI}_\alpha(\mathbf{x})\}$ *and* accurately estimates $p'(\mathbf{x})/p(\mathbf{x})$. For example, we could assume $\hat{r}(\mathbf{x}) = 1, \forall \mathbf{x}$, but the induced p' likely does not satisfy $p'(\mathbf{x})/p(\mathbf{x}) = 1, \forall \mathbf{x}$.

To solve these issues, we begin with an initial estimator $\hat{r}_0(\mathbf{x}) = 1, \forall \mathbf{x}$, and for $t \geq 0$ we sample from $p'(\mathbf{x}) \propto \exp\{\mathrm{CEI}_\alpha\}$ via SGLD using the current estimator $\hat{r}_t$, then update the classifier on those new samples to produce an updated estimator $\hat{r}_{t+1}$ for the next iteration. To keep the acquisition surface from changing too rapidly (potentially destabilizing our SGLD chain), we compute an exponential moving average of the classifier weights, and the averaged weights are used when computing gradients of $a(\mathbf{x})$. Our approach is analogous to (and directly inspired by) bootstrapped Q-learning (Mnih et al., 2013).

4.7 Empirical Results

In this section we empirically evaluate conformal BayesOpt. In Section 4.7.2 we investigate robustness to misspecified likelihoods. In Section 4.7.3 we evaluate conformal variants of EI, noisy expected improvement (NEI) and UCB on single-objective tasks, and compare the empirical coverage of credible and conformal prediction sets. In Section 4.7.4 we present additional results on multi-objective tasks. See Appendix D.6 for experimental details.

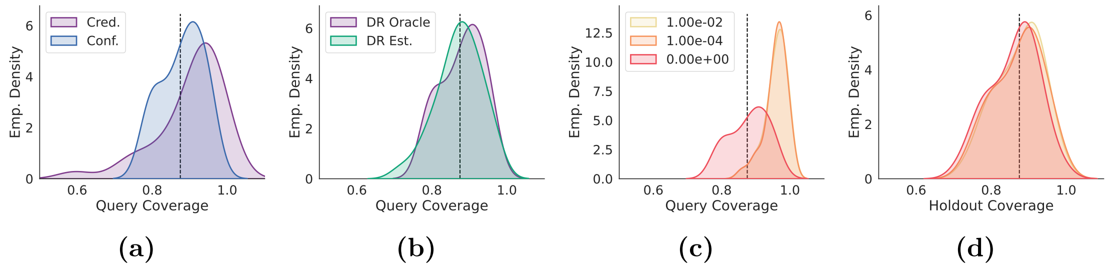


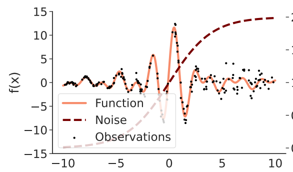
Figure 4.5: Here we quantify our approximation error in a simplified problem. The shaded regions in each panel depict a KDE estimate of the distribution of coverage when $n = 64$ and $\alpha = 0.125$, estimated from 32 independent trials. The black dashed line indicates $1 - \alpha$. In panel **(a)** we compare the coverage of Bayesian credible and randomized conformal prediction sets, if $\tau = 0$ and that we have access to a density ratio oracle. Conformal prediction provides much more consistent coverage. Next in panel **(b)** we show replacing the density ratio oracle with learned density ratio estimates has fairly minimal effect on the coverage of the resulting conformal prediction sets. In panel **(c)** we investigate the effect of the sigmoid temperature τ when $p(\mathbf{x}) \neq p(\mathbf{x})$. Similarly in panel **(d)** we investigate the effect of τ in the IID case when $p(\mathbf{x}) = p'(\mathbf{x})$. In general increasing τ tends to make the prediction sets more conservative.

4.7.1 Investigating Approximation Error

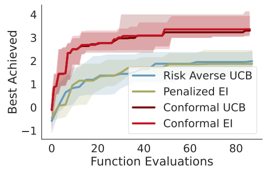
We begin with a carefully controlled experiment to investigate the effect of the approximations we introduced in Section 4.6. We consider a simplified setting where $p(\mathbf{x})$ and $p'(\mathbf{x})$ are known 3D spherical Gaussian distributions, f is the 3D Hartmann function, and $\varepsilon \sim \mathcal{N}(0, 0.05)$.

4.7.2 Conformal BayesOpt with a Misspecified Likelihood

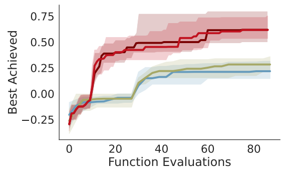
Recall from Section 4.3 that we often assume a simple homoscedastic likelihood $p(y|f) = \mathcal{N}(f, \sigma^2)$, where σ^2 is a learned constant. In Figure 4.6a we show a modified 1D sinc function, where $f^*(x) = (10 \sin(x) + 1) \sin(3x)/x$, with input-dependent noise $\varepsilon(x) \sim \mathcal{N}(0, 2/(1 + e^{x/2}))$, and $\mathcal{X} = [-10, 10]$. In Figure 4.6b we compare conformal BayesOpt with $p(y|f) = \mathcal{N}(f, \sigma^2)$ to



(a) sinc objective



(b) sinc, best obj. value



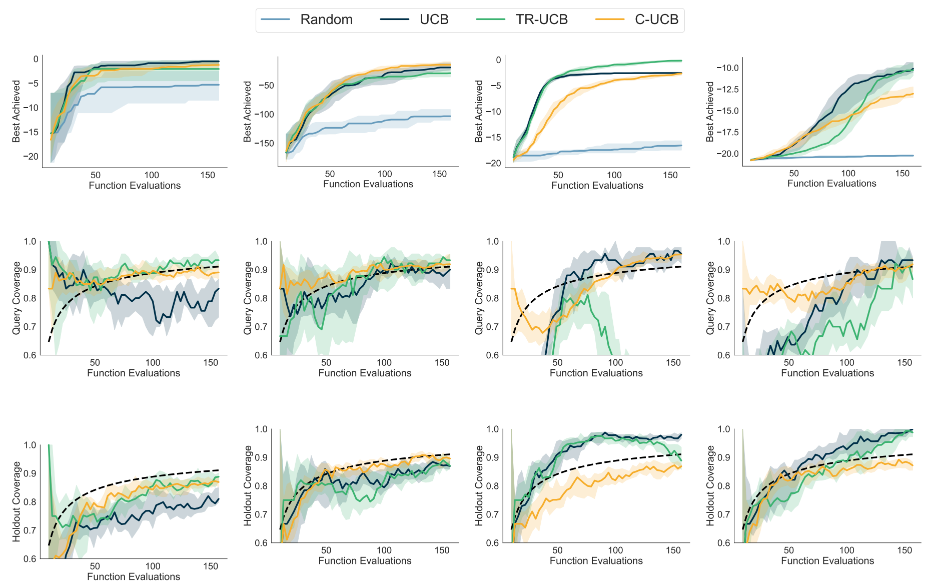
(c) double-knot, best obj. value

Figure 4.6: BayesOpt results on heteroscedastic, single-objective tasks sinc and double-knot (reporting median and its 95% conf. interval, estimated from 16 trials). **(a)** sinc function (left y axis) and noise process (right y axis). **(b)** sinc task, best objective value found by conformal BayesOpt with homoscedastic likelihoods compared to baselines, risk-averse UCB and penalized EI, both with heteroscedastic likelihoods. **(c)** double-knot task, same experiment as in panel (b). Conformal BayesOpt outperforms the specialized baselines on both tasks, despite being mis-specified.

two baselines specifically designed for this kind of task, specifically risk-averse UCB (Makarova et al., 2021) and penalized EI (Griffiths et al., 2019), which both use heteroscedastic likelihoods and custom acquisition functions. Both baselines require multiple replicates of each query to update their likelihoods, which significantly reduces sample efficiency. In Figure 4.6c, we repeat the same experiment on a second heteroscedastic task involving a double-knot function (Gramacy, 2005), where $f^*(\mathbf{x}) = -\mathbf{x}_1 e^{-\mathbf{x}_1 - \mathbf{x}_2}$, $\varepsilon(\mathbf{x}) \sim \mathcal{N}(0, ||\mathbf{x}||_2)$, and $\mathcal{X} = [-2, 6]^2$. Despite having a simpler, misspecified noise model, conformal BayesOpt outperforms its more specialized counterparts in terms of the final objective value and in terms of sample efficiency.

4.7.3 Evaluating Single-Objective Conformal Acquisitions

We now investigate the behavior of conformal BayesOpt more closely, evaluating the on synthetic Levy and Ackley functions with homoscedastic noise and varying input dimensionality. Here, we focus on comparing with both a stan-



(a) Levy-5, $q = 3$. **(b)** Levy-20, $q = 3$. **(c)** Ackley-5, $q = 3$. **(d)** Ackley-20, $q = 3$.

Figure 4.7: Top row: Best achieved across **(a)** Levy-5, **(b)** Levy-20, **(c)** Ackley-5. **Middle row:** Query coverage and **Bottom row:** Heldout set coverage. Here, we compare with UCB, TuRBO with a UCB acquisition (TR-UCB), and our conformal approach (C-UCB). All approaches tend to perform similarly in terms of sample complexity, except for C-UCB on Ackley-20 which reaches a slightly worse optimum. However, C-UCB is signifiantly better calibrated in terms of both its queries (middle row) and the heldout set (bottom row) in terms of reaching achievable queries.

dard acquisition function, upper confidence bound (UCB Srinivas et al., 2010), as well as trust region Bayesian optimization using the UCB acqusition (Eriksson et al., 2019). Further experimental results across a range of query batch sizes and acquisitions are in Appendix D.5.

We see that the conformal acquisitions perform strongly across problems in Figure 4.7 (top row), with slightly worse sample complexity on both Ackley dimensionalities than TR-UCB. Although rapid objective improvement relative to standard methods is encouraging, we also contrast the empirical coverage

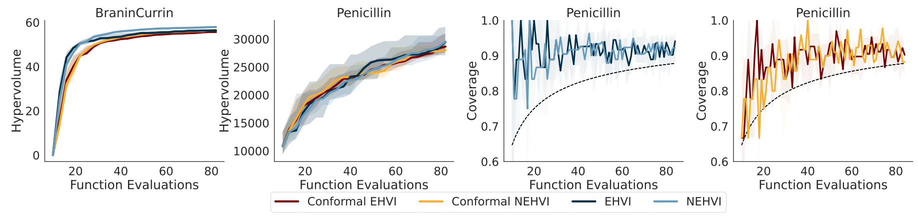


Figure 4.8: BayesOpt results on multi-objective tasks BraninCurrin and Penicillin (reporting median and its 95% conf. interval, estimated from 25 trials). **Left two panels:** Both conformal and standard acquisitions find solution sets with similar hypervolumes. **Right two panels:** Credible and conformal empirical coverage curves. The conformal curves track the target $1 - \alpha$ (black dashed line) better than the credible curves, but both are underconfident.

of credible and conformal prediction sets, which we show in the bottom two rows.

We set $\alpha = 1/\sqrt{n}$ and evaluate the coverage of the $100(1 - \alpha)\%$ credible and conformal prediction sets on the query points that we are optimizing with in the middle row. We follow the same procedure but for heldout examples in the bottom row, again averaged over 50 independent trials. The empirical conformal coverage curve does not perfectly satisfy our theoretical expectations, which would predict the average coverage to coincide very closely with the $1 - \alpha$ target curve. The conformal coverage curve better tracks and more predictably the curve for either Bayesian credible set, despite TurBO being designed specifically for model reliability.

4.7.4 Evaluating Multi-Objective Conformal Acquisitions

Finally, we consider two multi-objective tasks, BraninCurrin with $p = 2$ objectives and Penicillin where $p = 3$ (Liang & Lai, 2021). Here, the goal is not to find a single $\mathbf{x}^*$, but rather to find a set of all non-dominated solutions, the Pareto front. By non-dominated, we mean the set of solutions with the property that the objective value cannot increase in one dimension without decreasing in another. We measure the hypervolume of the current solution set relative to a reference point (Beume et al., 2009). We consider the expected hypervolume improvement (EHVI) (Emmerich, 2005; Emmerich et al., 2011; Daulton et al., 2020b) and noisy expected hypervolume improvement (NEHVI) (Daulton et al., 2021a) acquisition functions, and their conformal analogues. In Figure 4.8, we show that conformal acquisitions remain competitive with standard variants in terms of solution quality. Here, the conformal coverage curves track the target curve better than their credible counterparts, but the impact of discretization error in our estimates of the bounds of C_α plays a larger role as there are more label dimensions. More results are shown in Appendix D.5.

4.8 Discussion

We have proposed the conformal Bayes posterior and demonstrated its utility for reducing the sensitivity of BayesOpt to modeling assumptions, providing a mechanism to correct for the feedback covariate shift induced during optimization and improving the reliability of predictions. Although we focused on GP surrogates in the low-n regime, we expect many of these ideas to transfer to much larger models and datasets by replacing full conformal Bayes with split conformal Bayes, and either augmenting the GP surrogates with deep kernel learning, or by replacing GPs entirely with linear models operating on pretrained features learned by large self-supervised models. Finally we hope that this work spurs further interest into understanding the effects of model mis-specification and covariate shift in the context of BayesOpt specifically and in experimental design more broadly.

An author's most recent word need not be his last word.

Leonard J. Savage

5 Conclusion

We now leave the reader with a few closing thoughts.

Our work has made heavy use of GPs, and it is natural to wonder about the future of such work since current trends in ML research point away from mathematical sophistication towards general-purpose parametric model architectures pretrained on massive data corpora. If current trends continue, a common pattern will likely be to use large pretrained models for automatic feature engineering, then use either Bayesian linear models or GPs as the probabilistic "last layer". In this setting much of the work explored in this thesis can be directly applied by treating the feature space as the input space for the purposes of Bayesian inference.

The rising popularity of distribution-free uncertainty quantification methods like conformal prediction will hopefully cause engineers and scientists to consider more carefully what kind of uncertainty they are trying to measure, and subsequently to choose the appropriate uncertainty quantification method (or combination of methods).

Although specific methods may fall in and out of fashion, the principles of effective decision-making do not change. Decisions should always make use of all available information (up to resource constraints), should reflect coherent beliefs and preferences, and should be based on reliably valid predictions of the outcome.

Appendices

Appendix A: WISKI Supplementary Material

A.1.1 Efficient Computation of GP Predictive Distributions

In this section we provide a brief summary of a major contribution of Pleiss et al. (2018a). Since our cached approach to online inference was partially inspired by the approach of Pleiss et al. (2018a), it is helpful to first understand how predictive means and variances are efficiently computed in the batch setting.

The Lanczos algorithm is a Krylov subspace method that can be used as a subroutine to solve linear systems (i.e. the conjugate gradients algorithm) or to solve large eigenvalue problems (Golub & Van Loan, 2012). Given a square matrix $A \in \mathbb{R}^{n \times n}$ and initial vector $\mathbf{b} \in \mathbb{R}^n$, the d-rank Krylov subspace is defined as $\mathcal{K}_d(A, \mathbf{b}) := \text{span}\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \dots, A^{d-1}\mathbf{b}\}$. The Lanczos algorithm is an iterative method that produces (after d iterations) an orthogonal basis $Q_d \in \mathbb{R}^{n\times d}$ and symmetric tridiagonal matrix $T_d \in \mathbb{R}^{d\times d}$ such that $\mathbf{q}_i \in \mathcal{K}_d(A, \mathbf{b})$ and $T_d = Q_d^\top A Q_d$. [*] If we take $A \approx Q_d T_d Q_d^\top$ and compute the eigendecomposition $T_d = V_d \Lambda_d V_d^\top$, we can write $A \approx SS^\top$, where $S = Q_d V_d \Lambda_d^{1/2}$. Note that if $d = n$ then the decomposition is exact to numerical precision. Hence the computational cost of a root decomposition via Lanczos is $\mathcal{O}(dn^2 + d^2)$ (Trefethen & Bau III, 1997).

The predictive mean caches are straightforward, since they are just the solution $\mathbf{v} = (K_{\mathbf{aa}} + \sigma^2 I)^{-1}\mathbf{y}$ which can be stored regardless of the method used to solve the system (i.e. preconditioned CG, Cholesky factorization, *e.t.c.*). Once computed, in the exact inference setting the predictive mean is given by

$$\mathbf{m}_{\mathbf{b}|\mathcal{D}} = K_{\mathbf{ba}}\mathbf{v}.$$

For inference with SKI we take $\mathbf{v} = K_{\mathbf{uu}} W_{\mathbf{a}}^\top (W_{\mathbf{a}} K_{\mathbf{uu}} W_{\mathbf{a}}^\top + \sigma^2 I)^{-1}\mathbf{y}$ and

$$\mathbf{m}_{\mathbf{b}|\mathcal{D}} = W_{\mathbf{b}}\mathbf{v}.$$

For exact inference, the predictive covariance caching procedure of Pleiss et al. (2018a) begins with the root decomposition

$$K_{\mathbf{aa}} + \sigma^2 I = (Q_d V_d \Lambda_d^{1/2})(\Lambda_d^{1/2} V_d^\top Q_d^\top). \tag{A.1}$$

Since predictive variances require a root decomposition of $(K_{\mathbf{aa}} + \sigma^2 I)^{-1}$, they

[*]Q_d is conventionally used to denote the d-rank Lanczos basis.

store $V = Q_d^\top V_d^\top \Lambda_d^{-1/2}$ and obtain predictive variances as follows:

$$S_{\mathbf{b}|D} = K_{\mathbf{bb}} - K_{\mathbf{ba}} V V^\top K_{\mathbf{ab}}. \tag{A.2}$$

For inference with SKI the procedure is much the same, except the root decomposition in Eq. A.1 is replaced with that of the SKI kernel matrix,

$$W_{\mathbf{a}} K_{\mathbf{uu}} W_{\mathbf{a}}^\top + \sigma^2 I = (Q_d V_d \Lambda_d^{1/2})(\Lambda_d^{1/2} V_d^\top Q_d^\top), \tag{A.3}$$

$V = K_{\mathbf{uu}} W_{\mathbf{a}}^\top Q_d^\top V_d^\top \Lambda_d^{-1/2}$, and Eq. A.2 is modified to

$$S_{\mathbf{b}|D} = K_{\mathbf{bb}} - W_{\mathbf{b}} V V^\top W_{\mathbf{b}}^\top. \tag{A.4}$$

A.1.2 Deriving the WISKI Predictive Mean and Variance

In this section we derive the Woodbury Inverse SKI predictive distributions. In contrast to Pleiss et al. (2018a), the WISKI predictive mean and covariance can be formulated in terms of quantities that can be cached in $\mathcal{O}(m^2)$ space and updated with new observations in constant time. Recall the form of the Woodbury SKI inverse, $M_{\mathbf{a}} := (\sigma^2 K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}}^\top W_{\mathbf{a}})^{-1}$. For both the predictive mean and variance we begin with the standard SKI form, and show how to derive the WISKI form.

Predictive Mean

$$
\begin{aligned}
\mathbf{m}_{\mathbf{b}|D} &= W_{\mathbf{b}} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} (W_{\mathbf{a}} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} + \sigma^2 I)^{-1} \mathbf{y}_{\mathbf{a}}, \\
&= W_{\mathbf{b}} (I + \sigma^{-2} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} W_{\mathbf{a}})^{-1} (\sigma^{-2} K_{\mathbf{uu}}) W_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}}, \\
&= W_{\mathbf{b}} \left((\sigma^{-2} K_{\mathbf{uu}})(\sigma^{2} K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}}^{\top} W_{\mathbf{a}}) \right)^{-1} (\sigma^{-2} K_{\mathbf{uu}}) W_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}}, \\
&= W_{\mathbf{b}} (\sigma^{2} K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}}^{\top} W_{\mathbf{a}})^{-1} K_{\mathbf{uu}}^{-1} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}}, \\
&= W_{\mathbf{b}} M_{\mathbf{a}} W_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}}. \\
&= W_{\mathbf{b}} (\sigma^{-2} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}} - \sigma^{-2} K_{\mathbf{uu}} L (I + \sigma^{-2} L^{\top} K_{\mathbf{uu}} L)^{-1} L^{\top} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} \mathbf{y}_{\mathbf{a}},
\end{aligned}
$$

where $W_{\mathbf{a}}^{\top} W_{\mathbf{a}} = LL^{\top}$. The second line follows from the push-through identity (a special case of the Woodbury matrix identity).

Predictive Covariance The low-rank SKI predictive covariance of a GP is given (elementwise) by

$$
\begin{aligned}
S_{\mathbf{b}|D} &= W_{\mathbf{b}} K_{\mathbf{uu}} W_{\mathbf{b}}^{\top} - W_{\mathbf{b}} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} (W_{\mathbf{a}} K_{\mathbf{uu}} W_{\mathbf{a}}^{\top} + \sigma^2 I)^{-1} W_{\mathbf{a}} K_{\mathbf{uu}} W_{\mathbf{b}}^{\top} \\
&= \sigma^2 W_{\mathbf{b}} \left(\sigma^{-2} K_{\mathbf{uu}} - (\sigma^{-2} K_{\mathbf{uu}}) W_{\mathbf{a}}^{\top} (W_{\mathbf{a}} (\sigma^{-2} K_{\mathbf{uu}}) W_{\mathbf{a}}^{\top} + I)^{-1} W_{\mathbf{a}} (\sigma^{-2} K_{\mathbf{uu}}) \right) W_{\mathbf{b}}^{\top} \\
&= \sigma^2 W_{\mathbf{b}} (\sigma^{2} K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}}^{\top} W_{\mathbf{a}})^{-1} W_{\mathbf{b}}^{\top}, \\
&= \sigma^2 W_{\mathbf{b}} M_{\mathbf{a}} W_{\mathbf{b}}^{\top} \\
&= W_{\mathbf{b}} \left(K_{\mathbf{uu}} - K_{\mathbf{uu}} L (I + \sigma^{-2} L^{\top} K_{\mathbf{uu}} L)^{-1} L^{\top} K_{\mathbf{uu}} \right) W_{\mathbf{b}}^{\top}
\end{aligned}
$$

The third line immediately follows from an application of the Woodbury matrix identity to $(\sigma^{2} K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}}^{\top} W_{\mathbf{a}})^{-1}$. Following Pleiss et al. (2018a), we may compute two root decompositions of the form $BB^\top = K_{\mathbf{uu}}$ and $DD^\top \approx (I + \sigma^{-2} L^\top K_{\mathbf{uu}} L)^{-1}$ to speed up predictive variance computation, as this yields efficient diagonal computation:

$$S_{\mathbf{b}|D} = W_{\mathbf{b}} \left(BB^\top - BB^\top L DD^\top L^\top BB^\top \right) W_{\mathbf{b}}^\top .$$

As we noted in the main text, at time $t+1$, Q_t can be updated with a new observation in constant time via a Sherman-Morrison update, and $(W_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}})_{t+1} = (W_{\mathbf{a}}^\top \mathbf{y}_{\mathbf{a}})_t + W_{\mathbf{b}}^\top \mathbf{y}_{\mathbf{b}}$.

A.1.3 Conditioning on New Observations

Updating the Marginal Likelihood For fixed n, the Woodbury version of the log likelihood in WISKI is constant in n after an initial $\mathcal{O}(n)$ precomputation of $W_{\mathbf{a}}^\top W_{\mathbf{a}}$, as the only terms that ever get updated are the scalar σ^2 and the $m \times m$ matrix $K_{\mathbf{uu}}^{-1}$; this is in and of itself an advance over the computation speeds of other Gaussian process models, including SKI (Wilson & Nickisch, 2015).

Updating $W_{\mathbf{a}}^\top W_{\mathbf{a}}$. To update $W_{\mathbf{a}}^\top W_{\mathbf{a}}$ as we see new data points, we follow the general strategy of Gill et al. (1974) by performing rank-one updates to root decompositions:

$$\begin{aligned} \tilde{A} = A + \mathbf{z}\mathbf{z}^\top &= L(I + \mathbf{p}\mathbf{p}^\top)L^\top, & \mathbf{p} &= L^{-\top}\mathbf{z}, \\ &= LBB^\top L^\top = \tilde{L}\tilde{L}^\top, & BB^\top &= I + \mathbf{p}\mathbf{p}^\top, \end{aligned}$$

where $\tilde{L} = LB$. In our setting, the update is given by

$$(W_{\mathbf{a}}^\top W_{\mathbf{a}})_{t+1} = (W_{\mathbf{a}}^\top W_{\mathbf{a}})_t + W_{\mathbf{b}}^\top W_{\mathbf{b}}.$$

For full generality, we will assume q new points come at once, making $W_{\mathbf{b}} \in \mathbb{R}^{q \times p}$. Recall that $JJ^\top = (W_{\mathbf{a}}^\top W_{\mathbf{a}})^+$, let $\mathbf{p} = J_t^\top W_{\mathbf{b}}^\top$, which is the product of a $r \times p$ matrix and a $p \times q$ matrix, which costs $\mathcal{O}(prq)$. To compute the decomposition $BB^\top = I_r + \mathbf{p}\mathbf{p}^\top$ in a numerically stable fashion, we compute the SVD of $\mathbf{p} = USV^\top$ and use it to update the root decomposition:

$$\begin{aligned}
I_r + \mathbf{p}\mathbf{p}^\top &= I_r + USV^\top VSU^\top \\
&= U\texttt{diag}((S_{ii}^2 + 1); \mathbf{1}_{r-q})U^\top \\
&= U\texttt{diag}(\sqrt{S_{ii}^2 + 1}, \mathbf{1}_{r-q})\texttt{diag}(\sqrt{S_{ii}^2 + 1}, \mathbf{1}_{r-q})U^\top.
\end{aligned}$$

The SVD of this matrix costs $\mathcal{O}(q^2 r)$, assuming $q < r$ ($q = 1$ for most applications). The inner root is $B = U\texttt{diag}(\sqrt{S_{ii}^2 + 1}, \mathbf{1}_{r-q})$, and a final matrix multiplication costing $\mathcal{O}(pr^2)$ obtains the expression for the updated root $L_{t+1} = L_t B$. The updated inverse root is obtained similarly by

$$J_{t+1} = J_t U\texttt{diag}(1./\sqrt{S_{ii}^2 + 1}, \mathbf{1}_{r-q})$$

The overall computation cost is then $\mathcal{O}(prq + q^2 r + pr^2)$.

A.1.4 Online SKI and Deep Kernel Learning

When combining deep kernel learning (DKL) and SKI, the interpolation weight vectors $\mathbf{w}_i = w(\mathbf{x}_i, Z)$ become $\mathbf{w}_i = w(h(\mathbf{x}_i; \phi), Z')$, where $h : \mathbb{R}^d \to \mathbb{R}^{d'}$ is a feature map parameterized by ϕ and $Z' \in \mathbb{R}^{p \times d'}$. One implication of this change is that if the parameters of h change, then the interpolation weights must change as well. In the batch setting, the features and associated interpolation weights can be recomputed after every optimization iteration, since the cost of doing so is negligible compared to the cost of computing the MLL. The online setting does not admit the recomputation of past features and interpolation weights, because doing so would require $\mathcal{O}(n)$ work. Hence at any time t we must consider the previous features and interpolation weights $(\mathbf{h}_1, \mathbf{w}_1), \ldots, (\mathbf{h}_{t-1}, \mathbf{w}_{t-1})$ to be fixed. As a result, when computing the gradient of the MLL w.r.t. ϕ, we need only consider the terms that depend on $\mathbf{w}_t$.

Claim:

$$\nabla_{\mathbf{w}_t} \log p(\mathbf{y}_t | \mathbf{x}_{1:t}, \theta) = \nabla_{\mathbf{w}_t} \left[\frac{1}{2\sigma^2} \left(\mathbf{y}_t^\top W_t M_{t-1} W_t^\top \mathbf{y}_t - \frac{1}{1 + \mathbf{v}_t^\top \mathbf{w}_t} \left(\mathbf{v}_t^\top W_t^\top \mathbf{y}_t \right)^2 \right) - \frac{1}{2} \log(1 + \mathbf{v}_t \mathbf{w}_t) \right], \tag{A.5}$$

where

$$\mathbf{w}_t = w(h(\mathbf{x}_t; \phi), Z'), \qquad \mathbf{v}_t = M_{t-1} \mathbf{w}_t.$$

Proof:

$$\log p(\mathbf{y}_t|\mathbf{x}_{1:t},\theta) = -\frac{1}{2\sigma^2}(\mathbf{y}_t\mathbf{y}_t^\top - \mathbf{y}_t^\top W_t M_t W_t^\top \mathbf{y}_t) \\ - \frac{1}{2}\left(\log|K_{\mathbf{uu}}| - \log|M_t| + (n-m)\log\sigma^2\right) - \frac{t}{2}\log 2\pi.$$

Recalling $M_t := (K_{\mathbf{uu}}^{-1} + W_t^\top W_t)^{-1} = (K_{\mathbf{uu}}^{-1} + W_{t-1}^\top W_{t-1} + \mathbf{w}_t\mathbf{w}_t^\top)^{-1}$, by the Sherman-Morrison identity we have

$$M_t = M_{t-1} - \frac{1}{1+\mathbf{v}_t^\top \mathbf{w}_t}\mathbf{v}_t\mathbf{v}_t^\top. \tag{A.6}$$

Since M_{t-1} is constant w.r.t. $\mathbf{w}_t$, we can substitute Eq. (A.6) into the MLL and differentiate w.r.t. $\mathbf{w}_t$ to obtain

$$\nabla_{\mathbf{w}_t}\log p(\mathbf{y}_t|\mathbf{x}_{1:t},\theta) = \nabla_{\mathbf{w}_t}\frac{1}{2\sigma^2}\mathbf{y}_t^\top W_t(M_{t-1} - \frac{1}{1+\mathbf{v}_t^\top w_t}v_t v_t^\top)W_t^\top \mathbf{y}_t \\ + \frac{1}{2}\log|M_{t-1} - \frac{1}{1+\mathbf{v}_t^\top w_t}\mathbf{v}_t\mathbf{v}_t^\top|.$$

The quadratic term straightforwardly simplifies to the first two terms in Eq. (A.5). The final term results from an application of the matrix-determinant identity, once again dropping any terms with no dependence on $\mathbf{w}_t$,

$$\begin{aligned}
\log|M_{t-1} - \frac{1}{1+\mathbf{v}^\top\mathbf{w}}\mathbf{v}\mathbf{v}^\top| &= \log|M_{t-1}| - \log\left(1 + \frac{1}{1+\mathbf{v}^\top\mathbf{w}}\mathbf{v}^\top M_{t-1}^{-1}\mathbf{v}\right), \\
&= \log|M_{t-1}| - \log\left(1 + \frac{1}{1+\mathbf{v}^\top\mathbf{w}}\mathbf{v}^\top\mathbf{w}\right), \\
&= \log|M_{t-1}| - \log\left(1 + \mathbf{v}^\top\mathbf{w}\right). \qquad \blacksquare
\end{aligned}$$

A.1.5 Heteroscedastic Fixed Gaussian Noise Likelihoods and Dirichlet Classification

For a *fixed* noise term, the Woodbury identity still holds and we can still perform the updates in constant time. For fixed Gaussian noise, the term training covariance becomes

$$\begin{aligned}
K_{\mathbf{aa}} \approx K_{\text{SKI}} &= W_{\mathbf{a}}K_{\mathbf{uu}}W_{\mathbf{a}}^\top + D, \\
K_{SKI}^{-1} &= D^{-1} - D^{-1}W_{\mathbf{a}}(K_{\mathbf{uu}}^{-1} + W_{\mathbf{a}}^\top D^{-1}W_{\mathbf{a}})^{-1}W_{\mathbf{a}}^\top D^{-1}.
\end{aligned}$$

Plugging the second line into Eq. 2.13 tells us immediately that we need to store $\mathbf{y}_{\mathbf{a}}^\top D^{-1}\mathbf{y}_{\mathbf{a}}$ instead of $\mathbf{y}_{\mathbf{a}}^\top\mathbf{y}_{\mathbf{a}}$, $W_{\mathbf{a}}^\top D^{-1}W_{\mathbf{a}}$ instead of $W_{\mathbf{a}}^\top W_{\mathbf{a}}$, and $W_{\mathbf{a}}^\top D^{-1}\mathbf{y}_{\mathbf{a}}$ instead of $W_{\mathbf{a}}^\top\mathbf{y}_{\mathbf{a}}$. The rest of the online algorithm proceeds in the same manner as at each step, we update these caches with new vectors.

The heteroscedastic fixed noise regression approach naturally allows us to perform GP classification as in Milios et al. (2018). Given a one-hot encoding of the class probabilities, e.g. $y = e_c$ where c is the class number, they derive an approximate likelihood so that the transformed regression targets are

$$\tilde{y}_i = \log \alpha_i - \tilde{\sigma}_i^2/2, \qquad \tilde{\sigma}_i^2 = \log(1 + 1/\alpha_i),$$

where $\alpha_i = I_{y_i=1} + \alpha_\epsilon$, where α_ϵ is a tuning parameter. We use $\alpha_\epsilon = 0.01$ in our classification experiments. As there are C classes, we must model each class regression target; Milios et al. (2018) use independent GPs to model each class as we do. The likelihood at each data point over each class target is then $p(\tilde{y}_i|\mathbf{f}) = \mathcal{N}(f_i, \tilde{\sigma}_i^2)$, which is simply a heteroscedastic fixed noise Gaussian likelihood. Posterior predictions are given by computing the `argmax` of the posterior mean, while posterior class probabilities can be computed by sampling over the posterior distribution and using a softmax (Equation 8 of Milios et al. (2018)).

Challenges of Streaming Variational GP Inference

A.2.1 A Closer Look at O-SVGP

In this paper, we focus primarily on the online SVGP objective of Bui et al. (2017a), ignoring for the moment their α-divergence objective that is used in some of their models — which can itself be viewed as a type of generalized variational inference (Knoblauch et al., 2019).

Recalling the online uncollapsed bound of Bui et al. (2017a) and adapting

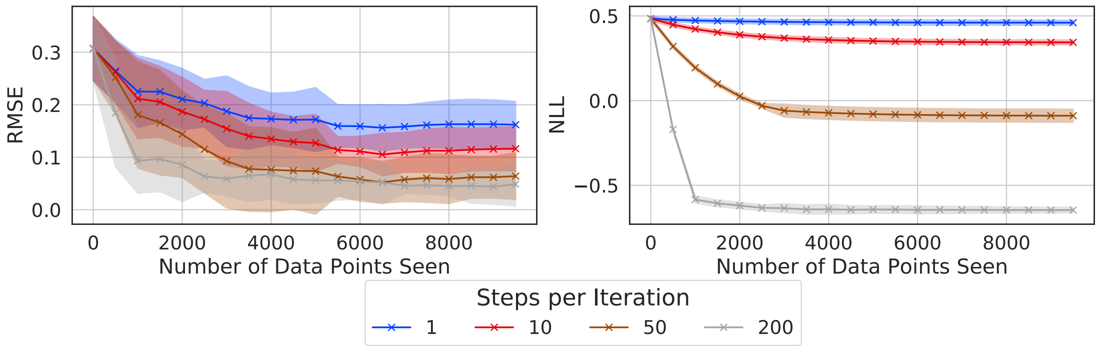


Figure A.1: (**Left:**) MSEs through the course of the dataset stream for up to 10,000 data points coming in batches of 500 data points for online SVGP. We varied the number of optimization steps per batch, finding that at least 10 steps were required to achieve good performance. The data points are drawn from a synthetic sine function corrupted by Gaussian noise. (**Right:**) NLLs over the course of the dataset stream; again, we see that many optimization steps are needed to decrease the NLL on the test set over the course of the stream.

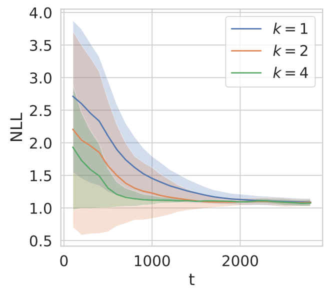
their notation — copying directly from their appendix, the objective becomes

$$\log p(\mathbf{y_b}|\theta) \geq \int \left[\log \frac{p\left(\mathbf{u}'|\theta'\right) q(\mathbf{u})}{p\left(\mathbf{u}|\theta\right) q'(\mathbf{u}') p\left(\mathbf{y_b}|\mathbf{u}\right)}\right] q(\mathbf{u}) d\mathbf{u} \tag{A.7}$$

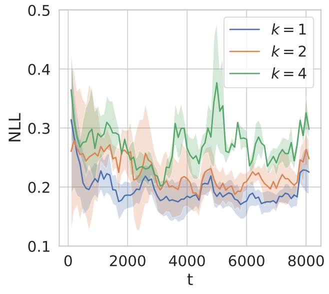
$$= -\mathbb{E}_{q(\mathbf{u})}(\log p\left(\mathbf{y_b}|\mathbf{u}\right)) + \mathrm{KL}\left(q(\mathbf{u}) \| p\left(\mathbf{u}|\theta\right)\right) \tag{A.8}$$

$$+ \mathrm{KL}\left(q(\mathbf{u}') \| q'(\mathbf{u}')\right) - \mathrm{KL}\left(q(\mathbf{u}') \| p\left(\mathbf{u}'|\theta'\right)\right),$$

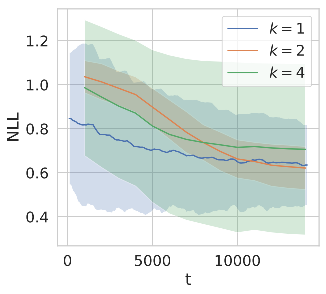
where $\mathbf{u}'$ is the old set of inducing function values, $\mathbf{u}$ is the new set of inducing function values, $q(\cdot)$ is the variational posterior on a set of points, θ are the current kernel hyperparameters to the GP, and θ' are the kernel hyperparameters for the GP at the previous iteration. In Eq. A.7, the first two terms are the standard SVI-GP objective (e.g from (Hensman et al., 2013c)), while the second two terms add to the standard objective allowing SVI to be applied to the streaming setting. Mini-batching can be achieved without knowing the



(a) Skillcraft



(b) Powerplant

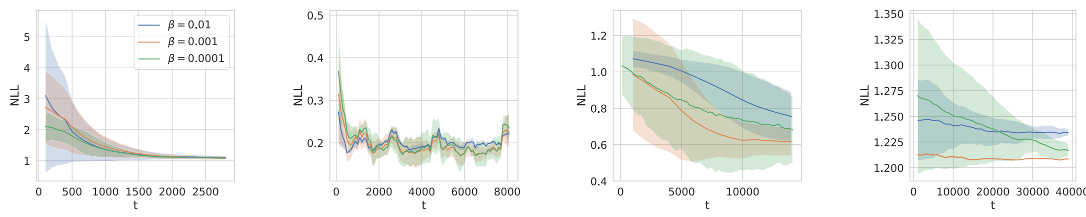


(c) Elevators

Figure A.2: Here we ablate the number of O-SVGP gradient updates per timestep in the context of UCI regression. Notably we see that the value we chose for our comparison in the main text ($k = 1$) performs well in comparison to larger values of k. In contrast to the results in Figure A.1, there is relatively little benefit to increasing the number of gradient update steps per batch when the batch size is very small (in our case, 1).

number of data points a priori; however, this achievement comes at the expense of having to compute two new terms in the loss.

Since the bound in Eq. A.7 is uncollapsed, it must be optimized to a global maximum at every timestep to ensure both the GP hyperparameters and the variational parameters are at their optimal values. As noted in Bui et al. (2017a), this optimization is extremely difficult in the streaming setting for two main reasons. 1) Observations may arrive in a non-iid fashion and violate the assumptions of SVI, and 2) each observation batch is seen once and discarded, preventing multiple passes through the full dataset, as is standard practice for SVI. Even in the batch setting, optimizing the SVI objective to a global maximum is notoriously difficult due to the proliferation of local maxima. This property makes a fair timing comparison with our approach somewhat difficult in that multiple gradient steps per data point will necessarily be slower than WISKI, which does not have any variational parameters to opti-



(a) Skillcraft **(b)** Powerplant **(c)** Elevators **(d)** Protein

Figure A.3: Here we ablate the β hyperparameter in the GVI loss for O-SVGP on the UCI datasets considered in this paper. While there is not a clear winner, we find that β = 1e-3 works well for all datasets.

mize, and thus can learn with fewer optimization steps per observation. We implemented Eq A.7 in GPyTorch (Gardner et al., 2018a), but additionally attempted to use the authors' provided implementation of O-SVGP[*] finding similar results — many gradient steps are required to reduce the loss. As a demonstration, we varied the number of steps of optimization per batch in Figure A.1 with a large batch size 300 (the same as Bui et al. (2017a)'s own experiments) in the online regression setting on synthetic sinusoidal data, finding that at least 10 optimization steps were needed to decrease the RMSE in a reasonable manner even on this simpler problem.

To remedy the convergence issue (and to create a fair comparison with *one* gradient step per batch of data), we down-weighted the KL divergence terms, producing a generalized variational objective (equivalent to taking the likeli-

[*]https://github.com/thangbui/streaming_sparse_gp

hood to a power $1/\beta$) (Knoblauch et al., 2019). Eq A.7 loss becomes:

$$\begin{aligned}\log p(\mathbf{y}_b|\theta) \geq -\mathbb{E}_{q(\mathbf{u})}(\log p\left(\mathbf{y_b}|\mathbf{u}\right)) + \beta \mathrm{KL}\left(q(\mathbf{u}) \| p\left(\mathbf{u}|\theta\right)\right) \\ + \beta \mathrm{KL}\left(q(\mathbf{u}') \| q'(\mathbf{u}')\right) - \beta \mathrm{KL}\left(q(\mathbf{u}') \| p\left(\mathbf{u}'|\theta'\right)\right),\end{aligned} \tag{A.9}$$

where all terms are as before except for $\beta < 1$. We found that $\beta << 1$ produces more reasonable results for a batch size of 1. Ablations for varying this hyperparameter are shown in Figure A.3. Using generalized variational inference does not change the complexity of the streaming objective, which remains $\mathcal{O}(Bp^2 + p^3)$; the $\mathcal{O}(p^3)$ term stays the same due to the log determinant term in the KL objective.

Finally, we vary the number of inducing points in the O-SVGP bound in Figure A.4, finding that O-SVGP is quite sensitive to the number of inducing points.

Experimental Details

A.3.1 Regression and Classification

Algorithm 5 summarizes online learning with WISKI. If an input projection is not learned, then $h(\mathbf{x}; \phi)$ can be taken to be the identity map, and the projection parameter update is consequently a no-op. In the rest of this section we provide additional experimental results in the regression and classification setting. In Figure 2.4 we report the RMSE for each of the UCI regression tasks. Note that the RMSE is computed on the standardized labels. The qualitative

Algorithm 5: Online Learning with WISKI

```
Input: Kernel function $k$, inducing grid $Z$, initial data $D = (X_\mathbf{a}, \mathbf{y}_\mathbf{a})$, GP
       parameters $\theta$, feature map parameters $\phi$, learning rate $\eta$.
Initialize $K_{\mathbf{uu}}, L, W_\mathbf{a}^\top \mathbf{y}_\mathbf{a}, \mathbf{y}_\mathbf{a}^\top \mathbf{y}_\mathbf{a}$.
for $t = n+1, n+2, \dots$ do
    Receive new $X_\mathbf{b}$.
    Predict $\hat{p}(\mathbf{y}_\mathbf{b} | X_\mathbf{b}, D, \theta, \phi)$ (Eq. 2.14).
    Observe new $\mathbf{y}_\mathbf{b}$, compute $W_\mathbf{b}$.
    $\mathbf{y}_\mathbf{a}^\top \mathbf{y}_\mathbf{a} \leftarrow \mathbf{y}_\mathbf{a}^\top \mathbf{y}_\mathbf{a} + \mathbf{y}_\mathbf{b}^\top \mathbf{y}_\mathbf{b}$.
    $W_\mathbf{a}^\top \mathbf{y}_\mathbf{a} \leftarrow W_\mathbf{a}^\top \mathbf{y}_\mathbf{a} + W_\mathbf{b}^\top \mathbf{y}_\mathbf{b}$
    $L \leftarrow (LL + W_\mathbf{b}^\top W_\mathbf{b})^{1/2}$
    $D \leftarrow D \cup (X_\mathbf{b}, \mathbf{y}_\mathbf{b})$
    $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}(\phi)$ (Eq. 2.18).
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$ (Eq. 2.13).
end
```

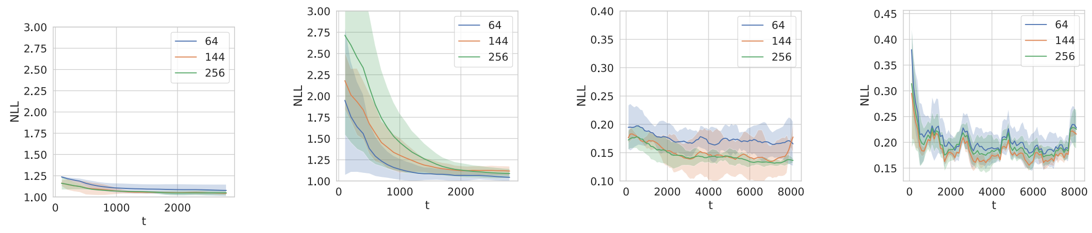
behavior is identical to that of the NLL plots in the main text (Figure 2.4). Figure A.5 is a visualization of a WISKI classifier on non-i.i.d. data. Figures A.4 and A.3 report the results of our ablations on p and β, respectively. This section provides all necessary implementation details to reproduce our results.

Data Preparation For all datasets, we scaled input data to lie in $[-1, 1]^d$. For regression datasets we standardized the targets to have zero mean and unit variance. If the raw dataset did not have a train/test split, we randomly selected 10% of the observations to form a test dataset. From the remainding 90% we removed an additional 5% of the observations for pretraining.

Hyperparameters We pretrained all models for T_{batch} epochs, with learning rates η_{batch}. If we learned a projection of the inputs, we used a lower learning rate for the projection parameters. While a small learning rate worked well

Table A.1: Hyperparameters

Task	m	T_{batch}	$\eta_{\text{batch}}(\theta)$	$\eta_{\text{batch}}(\phi)$	$\eta_{\text{online}}(\theta)$	$\eta_{\text{online}}(\phi)$	β
Banana	256	200	5e-2	-	5e-3	-	1e-3
SVM Guide 1	256	200	5e-2	-	5e-3	-	1e-3
Skillcraft	256	200	5e-2	5e-3	5e-3	5e-4	1e-3
Powerplant	256	200	5e-2	5e-3	5e-3	5e-4	1e-3
Elevators	256	200	1e-2	1e-3	1e-3	1e-4	1e-3
Protein	256	200	1e-2	1e-3	1e-3	1e-4	1e-3
3DRoad	1600	800	1e-2	-	1e-3	-	1e-3



(a) WISKI, Skillcraft

(b) O-SVGP, Skillcraft

(c) WISKI, Powerplant

(d) O-SVGP, Powerplant

Figure A.4: Here we ablate the number of inducing points for both WISKI and O-SVGP. We find that WISKI is not very sensitive to the number of inducing points, but always improves if more inducing points are added. O-SVGP sometimes performs better with fewer inducing points, a phenomenon we attribute to either 1) poor optimization of the GVI objective or 2) overfitting due to the downweighted KL terms in the GVI objective. In theory adding inducing points should only improve the performance of an SVGP. This observation highlights the difficulties O-SVGP often encounters in practice.

for all tasks, for the best performance we used a higher learning rate for easier tasks (Table A.1).

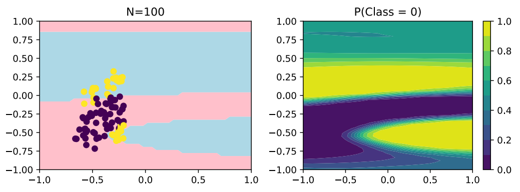
p	r	NLL
256	128	8.2e+6 $\pm$ 9.8e+6
256	192	1.000 $\pm$ 0.010
256	256	1.007 $\pm$ 0.015
1024	256	2.9e+7 $\pm$ 9.2e+7
1024	512	1.050 $\pm$ 0.082
1024	768	0.995 $\pm$ 0.007
1024	1024	1.007 $\pm$ 0.008

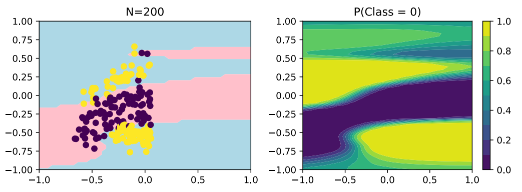
Table A.2: Root decomposition rank (r) sensitivity comparison by NLL across both $p = 256$ and $m = 1024$ inducing points on skillcraft. Too small of a rank fails to converge; however, once r is large enough (about $p/2$), the performance is unchanged.

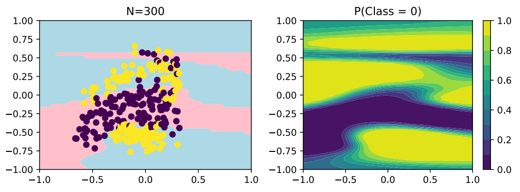
A.3.2 Bayesian Optimization Experimental Details and Further Results

For the Bayesian optimization experiments, we considered noisy three dimensional versions of the Bayesian optimization test functions available from BoTorch[*]. We sed the BoTorch implementation of the test functions with the `qUCB` acquisition function with $q = 3$, randomly choosing five points to initialize with and running 1500 BO steps, so that we end up with 4505 data points acquired from the models. We then followed BoTorch standard optimization of the acquisition functions by optimizing with LBFGS-B with 10 random restarts, 512 samples to initialize the optimization with, a batch limit of 5 and 200 iterations of LBFGS-B. We fit the model to convergence at each iteration as model fits are very important in BO using LBFGS-B for exact and WISKI while using Adam for OSVGP because the variational parameters are much higher dimensional so LBFGS-B is prohibitively slow. The timing results take into account the model re-fitting stage, the acquisition optimization stage, and the

[*]https://botorch.org/api/test_functions.html


(a) Test Accuracy - 70%


(b) Test Accuracy - 70%


(c) Test Accuracy - 77%

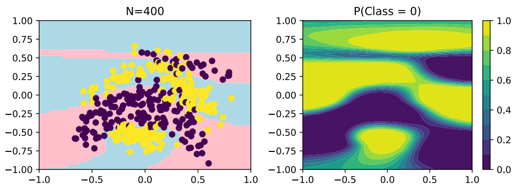

(d) Test Accuracy - 88%

Figure A.5: Online Gaussian Process Dirichlet classification with WISKI on observations from the banana dataset arriving in non-i.i.d. fashion (shown). The WISKI classifier is updated with a single gradient step after each individual observation.

expense of adding a new datapoint into the model. We used a single AWS instance with eight Nvidia Tesla V100s for these experiments, running each experiment four times, except for StyblinskiTang, which we ran three times (as the exact GP ran out of memory during a bayes opt step on one of the seeds). We measure time per iteration by adding both the model fitting time and the acquisition function optimization time. While a single training step is somewhat faster for O-SVGP than for WISKI, we found that it tended to take longer to optimize acquisition functions in BO loops.

Results over time per iteration and maximum achieved value by time for the rest of the test suite are shown in Figure A.6. Overall WISKI performs comparably in terms of maximum achieved value to the exact GP reaches that value in terms of quicker wallclock time. In Figure A.7, we show the maximum value achieved by iteration for each problem, finding that the exact GPs typically converge to their optimum first, while WISKI converges afterwards with OSVGP slightly after that. In Figure A.8, we show the time per iteration for all three methods, finding that WISKI is constant time throughout as is OSVGP, while the exact approach scales broadly quadratically (as expected given that the BoTorch default for sampling uses LOVE predictive variances and sampling). Digging deeper into the results, we found that the speed difference between OSVGP and WISKI is attributable to the increased predictive variance and sampling speed for OSVGP ($\mathcal{O}(p^3)$ compared to $\mathcal{O}(p^2)$).

Levy	Ackley	StyblinskiTang	Rastrigin	Griewank	Michalewicz
10.0	4.0	20.0	10.0	4.0	5.0

Table A.3: Noise scale used for WISKI Bayesian optimization experiments.

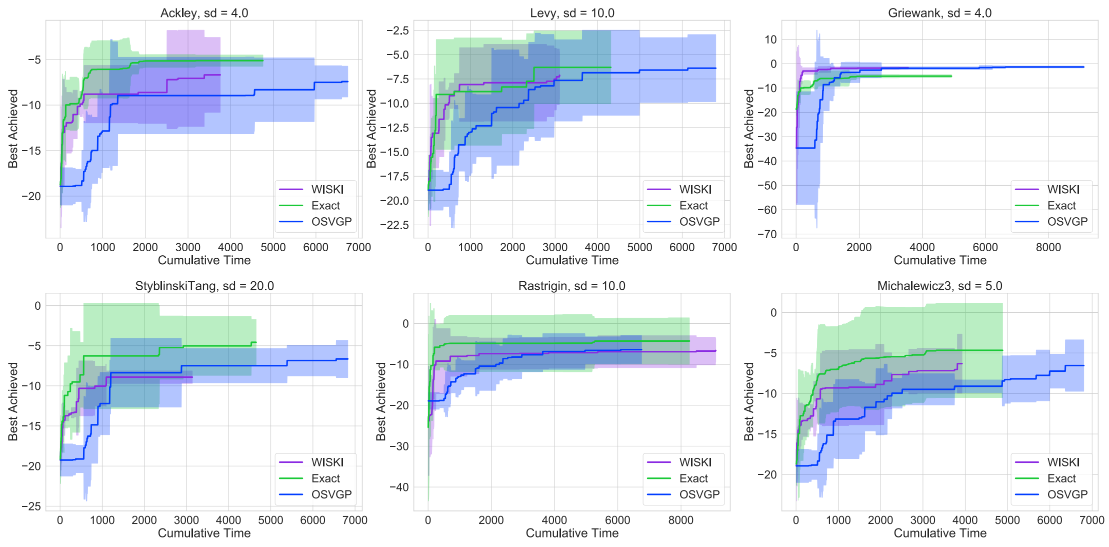


Figure A.6: Bayesian optimization results in terms of total optimization time. Throughout, WISKI is generally the fastest, except on Griewank, while reaching similar optimization performance to the exact GPs across the board. WISKI is somewhat better but significantly faster than the other methods on Griewank, but with similar performance to O-SVGP on StyblinskiTang and Michalewicz.

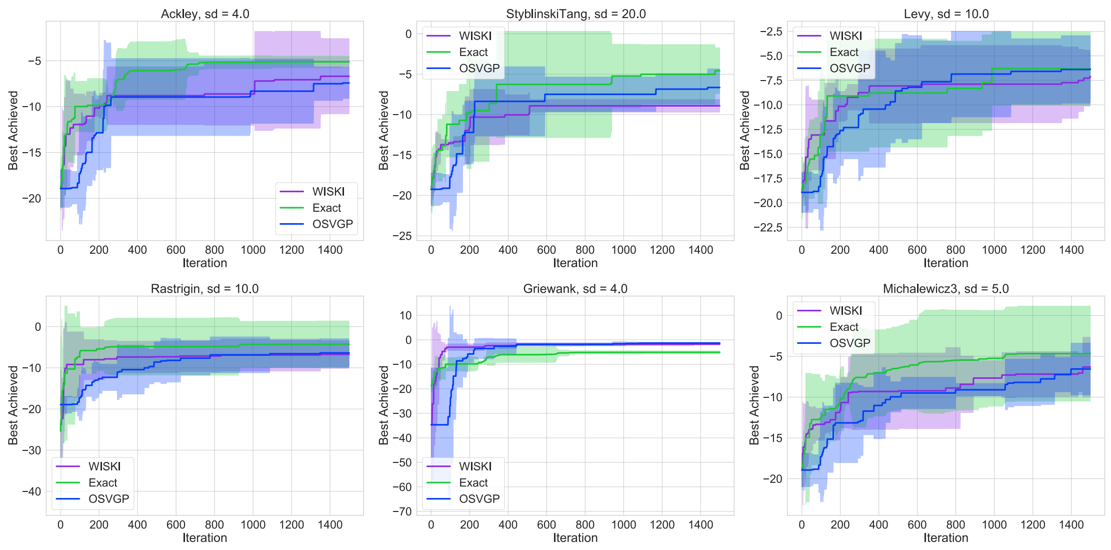


Figure A.7: Bayesian optimization results in terms of iteration complexity for noisy 3D test functions. Throughout, WISKI performs comparably to the exact GP.

A.3.3 Active Learning Experimental Details

For the active learning problem, we used a batch size of 6 for all models, with a base kernel that was a scaled ARD Matern-0.5 kernel, and lengthscale priors of Gamma(3, 6) and outputscale priors of Gamma(2, 0.15). For both OSVGP and WISKI, we used a grid size of 900 (30 per dimension); for OSVGP, we trained the inducing points, finding fixed inducing points did not reduce the RMSE. For O-SVGP, we used $\beta = 0.001$ and a learning rate of 1e-4 with the Adam optimizer, while for WISKI and the exact GPs, we used a learning rate of 0.1. Here, we re-fit the models until the training loss stopped decaying, analogous to the BO experiments. The dataset can be downloaded at `https://wjmaddox.github.io/assets/data/malaria_df.hdf5`.

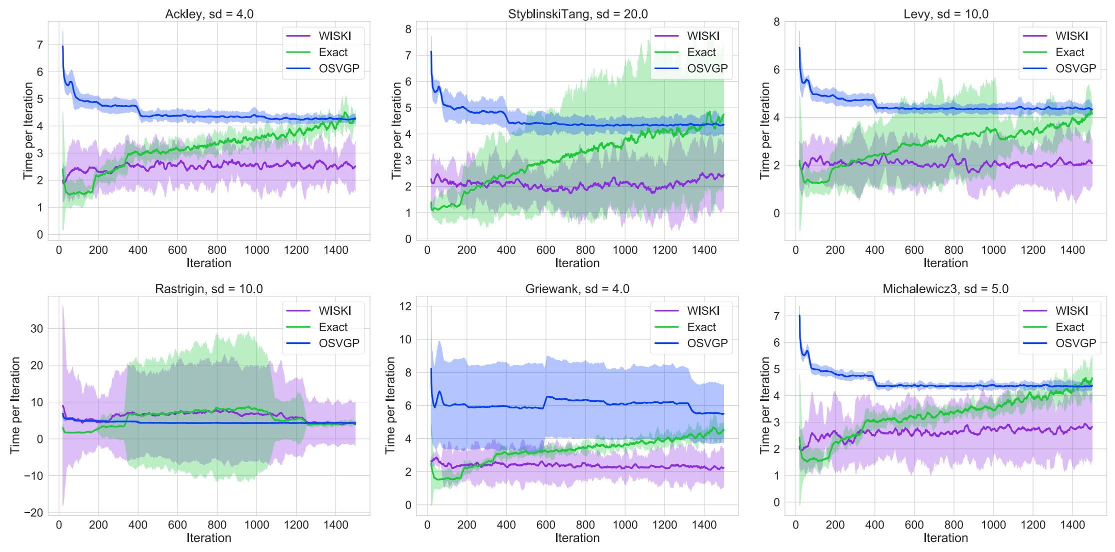


Figure A.8: Bayesian optimization results in terms of time complexity for the noisy 3D test functions. WISKI is the fastest method on the problems, while the exact GPs increase time per iteration at least linearly (they use LOVE predictive variances internally). While O-SVGP is constant time, it typically is somewhat slower due to larger constants with respect to n, p^3 versus p^2 for WISKI.

Appendix B: OVC Supplementary Material

Appendix B is structured as follows:

- Appendix B.1 discusses broader impacts and limitations.
- Appendix B.2 discusses the background in more detail.
- Appendix B.3 discusses Newton's method and training objectives for the variational models, before giving more detail on the methodological work, including implementation details.
- Appendix B.4 shows two more understanding experiments along with ablation studies, with Appendix B.4.2 giving detailed experimental and data descriptions.

Limitations and Societal Impacts

B.1.1 Limitations

From a practical perspective, we see several interrelated limitations:

- In our software implementation, we currently support single-output functions. This is partly because GPyTorch currently has limited support for fantasization of multi-task Gaussian processes; see `https://github.com/cornellius-gp/gpytorch/pull/805`.

- For non-Gaussian likelihoods, the approximation may break down for long rollout time steps due to accumulating error from the Laplace approximation.

- Incorrect modelling and thus incorrect rollouts will tend to be even more influential in high dimensional settings, as the kernels we use do not tend to work particularly well in high dimensions (Eriksson et al., 2019).

- If the underlying data is non-stationary, then long range predictions may suffer. For example, local models are superior on the rover problem in Figure 2.10, even compared to global models with advanced acquisition functions (e.g. Figure B.5c). This limitation is remedied somewhat by the local modelling approaches we use (Eriksson et al., 2019; Wang et al., 2020).

- Using many inducing points can worsen numerical conditioning, leading to less stable multi-step fantasization, despite the theoretical advantages of more inducing points (e.g., Bauer et al., 2016).

B.1.2 Societal Impacts

We do not anticipate that our work will have negative societal impacts. To the contrary, as we demonstrate in this paper, Bayesian optimization is naturally suited to applications in the public interest, such as public health surveillance (Andrade-Pacheco et al., 2020). These types of applications should help benefit global populations by allowing more targeted interventions in the public health setting. However, reliance solely on machine learning models, for example, in quantitative finance settings (as our volatility model in Figure 2.3c is designed for), could potentially lead to over-confidence and financial shocks as has previously been the case (MacKenzie & Spears, 2014).

Further Background

In this section, we describe further related work on both variational inference in the streaming setting as well as the use of sparse GPs in both BO and control before describing Newton's iteration for Laplace approximations and training methods for exact and variational GPs.

B.2.1 Extended Related Work

Because our work explores three distinct applications, namely black-box optimization, active learning, and control, there was more noteworthy related work than the space constraints of the main text would permit. Here we present a more complete literature review.

We do not focus on other approaches for streaming variational GP inference ranging from decoupled inducing points (Cheng & Boots, 2016; Kapoor et al., 2020; Shi et al., 2020) to alternative variational constructions (Moreno-Muñoz et al., 2019; 2020) to Kalman filtering based approaches (Huber, 2014; 2013) to deep linear model approaches (Pan et al., 2020; Titsias et al., 2019). These approaches could potentially be folded into OVC; however, we leave exploration of these for related work.

We also focus solely on the variationally sparse GPs introducted by Titsias (2009a) and Hensman et al. (2013b), rather than other approaches, which are potentially amenable to being used within OVC. Of particular note, the classical variational approximations for non-Gaussian likelihoods such as the approaches proposed originally by Csató & Opper (2002) and later Opper & Archambeau (2009) seem particularly promising, as is the development of kernel methods within the exponential family more generally (Canu & Smola, 2006).

From a software perspective, both GPFlowOpt (Knudde et al., 2017) and its successor Trieste (`https://github.com/secondmind-labs/trieste`) seem to support variational GPs for BO but we do not know of a comprehensive benchmarking of their implementation. Implementing variational GPs for most acquisitions in BoTorch (Balandat et al., 2020a) is entirely possible using GPyTorch, as we did, although it requires some engineering work and is not natively supported at this time.

Recent work on molecular design has combined variational auto-encoders (VAEs) trained to map high dimensional structured molecular representations to low-dimensional latent representations with SVGPs trained to predict a molecular property of interest from the latent encoding (Gómez-Bombarelli et al., 2018; Tripp et al., 2020), but they primarily use simple acquisition functions.

In the control literature, sparse GPs are more popular, stemming from the seminal works of Csató & Opper (2002) and Girard et al. (2002). Chowdhary et al. (2014) directly apply the approach of Csató & Opper (2002) to optimal control problems, while Ling et al. (2016) sparsify GPs for active learning and planning. Groot et al. (2011), Boedecker et al. (2014), and Bijl et al. (2016) perform multiple time step look ahead via moment matching using sparse GPs, while Pan et al. (2017) use a similar approach with random fourier features. Deisenroth & Rasmussen (2011) use sparse GPs to speed up dynamics models for robotics, while Sæmundsson et al. (2018) use sparse GPs for meta reinforcement learning. Finally, Xu et al. (2014) use sparse GPs for robot localization tasks in control.

B.2.2 Laplace Approximations and Newton's Iteration

Newton iteration iterates

$$\begin{aligned}
\mathbf{b}_{t+1} &= (K^{-1} - H)^{-1}(\nabla \log p(\mathbf{y}_{\mathbf{b}}|\mathbf{b}_t) - H\mathbf{b}_t) \\
&= (K + KH^{-1/2}B^{-1}H^{-1/2}K)(\nabla \log p(\mathbf{y}_{\mathbf{b}}|\mathbf{b}_t) - H\mathbf{b}_t), \\
B &:= (I - H^{1/2}KH^{1/2}),
\end{aligned}$$

with $H = \nabla^2_{\mathbf{b}} \log p(\mathbf{y_b}|\mathbf{b})$, i.e. the Hessian of the pointwise log-likelihood, until $||\mathbf{b}_{t+1} - \mathbf{b}_t||_2^2 < \epsilon$ (i.e. until a stationary point is reached). The expensive computational cost is the linear system involving B^{-1}, since H is diagonal. In the online case, we only apply Newton iteration to the latest batch of observations, so that the time complexity is just $\mathcal{O}(q^3)$. Please see Rasmussen & Williams (Chapter 3 of 2008) for further detail.

Implementation wise, for all but the GPCV experiment in Figure 2.3c, we manually implemented the gradient and Hessian terms as they are well known due to being natural parameterizations of exponential families. For the preference learning experiment, we followed the gradient derivation in Chu & Ghahramani (2005).

B.2.3 Training Mechanisms for Exact Gaussian Processes and Variational Gaussian Processes

Please see the more detailed summaries of Rasmussen & Williams (2008) for training methods of exact Gaussian processes as well as the theses of Van der Wilk (2019) and Matthews (2017) for training methods of sparse Gaussian processes. In what follows, we assume solely a zero mean function and suppress dependence on θ for the kernel matrices.

Log Marginal Likelihood for Exact GPs Training the GP's hyperparameters, θ, proceeds via maximizing the marginal log-likelihood (MLL).

The MLL is given by

$$\log p(\mathbf{y_a}|X_\mathbf{a}, \theta) = -\frac{1}{2}\mathbf{y_a}^\top (K_\mathbf{aa} + \sigma^2 I)^{-1}\mathbf{y_a} - \frac{1}{2}\log|K_\mathbf{aa} + \sigma^2 I| - \frac{n}{2}\log 2\pi, \quad \text{(B.1)}$$

over the training set, $\mathcal{D} = (X_\mathbf{a}, \mathbf{y_a})$.

Collapsed Evidence Lower Bound for SGPR Sparse Gaussian process regression (SGPR) begins by approximating the kernel with a lower rank version. The training data covariance, $K_\mathbf{aa}$, is replaced by the Nyström approximation $K_\mathbf{aa} \approx Q_\mathbf{aa} := K_\mathbf{au} K_\mathbf{uu}^{-1} K_\mathbf{ua}$, where $K_\mathbf{uu}$ is the kernel evaluated on the inducing point locations, Z. SGPR learns the locations of the inducing points and the kernel hyperparmeters through a 'collapsed' form of the evidence lower bound (ELBO), yielding a variational adaptation of older Nyström or projected process approximations (Rasmussen & Williams, 2008, Chapter 7).

The ELBO is called "collapsed" because the Gaussian likelihood allows the parameters of the variational distribution to be analytically optimized and integrated out (collapsed), yielding a bound that depends only on the inducing point locations and the kernel hyperparameters (Titsias, 2009a). The SGPR bound is as follows (see Titsias (2008) for the full derivation):

$$\log p(\mathbf{y_a}|\theta) \geq \log p(\mathbf{y_a}|0, \sigma^2 I + Q_\mathbf{aa}) - \frac{1}{2\sigma^2}\texttt{trace}(K_\mathbf{aa} - Q_\mathbf{aa}). \quad \text{(B.2)}$$

We can apply standard gradient based training to both the kernel hypers θ as well as the inducing locations Z. Jankowiak et al. (2020b) derive a variational version of this bound which enables sub-sampling across data points; we leave training with that bound for future work.

Evidence Lower Bound for SVGP The advance of Hensman et al. (2013b) is that they do not compute the optimal variational parameters at each time step proposing the sparse variational GP (SVGP). Their derivation, see Hensman et al. (2013b; 2015) for a full derivation, yields an 'uncollapsed' ELBO (so named due to its explicit dependence on the variational parameters),

$$\log p(\mathbf{y_a}|\theta) \geq \mathbb{E}_{q(\mathbf{a})}[\log p(\mathbf{y_a}|\mathbf{a})] - \mathrm{KL}(q(\mathbf{u})||p(\mathbf{u})), \tag{B.3}$$

where $q(\mathbf{a}) = \int p(\mathbf{a}|\mathbf{u})\phi(\mathbf{u})d\mathbf{u}$.[*] The GP posterior $p(\mathbf{u}|D)$ is replaced with a parametric variational distribution, $\phi(\mathbf{u}) = \mathcal{N}(\mathbf{m_u}, S_\mathbf{u})$. $q(\mathbf{a})$ can be determined via Gaussian marginalization from $\phi(\mathbf{u})$. and is $q(\mathbf{a}) = \mathcal{N}(K_\mathbf{au}K_\mathbf{uu}^{-1}\mathbf{m_u}, K_\mathbf{aa} - K_\mathbf{au}K_\mathbf{uu}^{-1}(K_\mathbf{uu} - S_\mathbf{u})K_\mathbf{uu}^{-1}K_\mathbf{ua})$. Mini-batching is possible because the first term in Eq. B.3 decomposes additively over each of the n data points since each $(\mathbf{y}_a)_i$ is conditionally independent given $\mathbf{u}$. Mini-batching enables the use of stochastic optimization techniques, reducing the per iteration cost to be constant w.r.t. n, the number of observations.

Bui et al. (2017a) introduce a streaming version of the ELBO that involves two further terms; the method is called O-SVGP. Their primary model, O-SGPR (called the "collapsed bound" of a streaming sparse GP in that paper) is trained through a variant of their ELBO bound that integrates out the vari-

[*]Note that $q(\mathbf{u}) = \phi(\mathbf{u})$.

ational parameters. We describe this collapsed bound in more detail in Appendix B.3.4.

Predictive Log Likelihood for SVGP We close this section by quickly describing the predictive log likelihood (PLL) method of Jankowiak et al. (2020b), which is motivated by attempting to improve the calibration of SVGP models trained via the ELBO. The key advance in Jankowiak et al. (2020b) is that they consider the expectation over both the data and the response, rather than simply the expectation over the response in the variational distribution, producing an objective that becomes:

$$\begin{aligned}\ell_{\text{ppgpr}}(\theta, Z) &= \mathbb{E}_{p(\mathbf{x},\mathbf{y})}[\log p(\mathbf{y}|\mathbf{x},\theta)] - \beta \mathrm{KL}(q(\mathbf{u})||p(\mathbf{u})) \\ &\approx \frac{1}{n}\sum_{i=1}^{n} \log\left\{\mathbb{E}_{q(f(\mathbf{x}_i))}[p(\mathbf{y}_i|f(\mathbf{x}_i),\theta)]\right\} - \beta \mathrm{KL}(q(\mathbf{u})||p(\mathbf{u})).\end{aligned}$$

We consider this optimization objective for several of the larger-scale problems here.

Further Methodological Details

In this Appendix, we begin by presenting our approach, OVC, as a generalization of exact Gaussian conditioning for SGPR (B.3.1) before describing an alternative interpretation of Bui et al. (2017a) that is equivalent to our approach in B.3.2. Then, in Appendix B.3.3, we describe the practical implementation of OVC. In B.3.4, we describe a flaw of the O-SGPR bound for small batch sizes. Finally, in B.3.5, we give an extended description of look-ahead Thompson sampling (LTS).

B.3.1 OVC Generalizes Efficient Batch SGPR Conditioning

In this section, we show that OVC can also be viewed as a generalization of Gaussian conditioning for SGPR. Gaussian conditioning for SGPR is recovered as a special case of OVC when $\theta' = \theta$ and $Z' = Z$. Under this assumption, Eqns. (2.21) and (2.22) simplify as follows:

$$\mathbf{c} = K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} \mathbf{y} + \mathbf{c}', \quad C = K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}} + C'. \tag{B.4}$$

Considering the predictive distribution given by Eq. 2.4 in the main text, as we add new data points, $(X_{\text{batch}}, \mathbf{y})$, into the model, we need to update $A = (K_{\mathbf{uu}} + C)^{-1}$, $\mathbf{v} = A\mathbf{c}$, and $R = (K_{\mathbf{uu}}^{-1} - A)^{1/2}$. For homoscedastic likelihoods the A update is a fast low-rank update via the Woodbury matrix identity,

$$A = A' - A' K_{\mathbf{ua}} (\sigma^2 I + K_{\mathbf{au}} A' K_{\mathbf{ua}})^{-1} K_{\mathbf{au}} A'. \tag{B.5}$$

This produces efficient Sherman-Morrison updates to generate $\mathbf{v}$ (via addition and matrix vector multiplication) and Woodbury based updates to update R, the predictive covariance cache via low-rank updates (e.g. Proposition 2 of Jiang et al. (2020a)).

Furthermore, these low rank updates can be used to produce updates to the exact caches. These updates are simply exact Gaussian conditioning with an approximate kernel. That is, the SGPR caches are merely transformed exact caches for any given set of data points. To demonstrate, we use a Nyström approximation for the kernel throughout, e.g. $K_{\mathbf{ab}} \approx K_{\mathbf{au}} K_{\mathbf{uu}}^{-1} K_{\mathbf{ub}}$, then after applying Woodbury we can write the exact caches using A:

$$\begin{aligned} VV^\top &= (K_{\mathbf{aa}} + \Sigma_{\mathbf{y}})^{-1} = \Sigma_{\mathbf{y}}^{-1} - \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}} (K_{\mathbf{uu}} + K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}})^{-1} K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} \\ &= \Sigma_{\mathbf{y}}^{-1} - \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}} A K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} \end{aligned}$$

with a similar expression for the predictive mean cache as

$$\mathbf{v} = (K_{\mathbf{aa}} + \Sigma_{\mathbf{y}})^{-1} \mathbf{y} = \Sigma_{\mathbf{y}}^{-1} \mathbf{y} - \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}} A \mathbf{c}.$$

Next we take the caches computed on every training point and project them into the space of inducing points by multiplying them by $K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}}$. It takes a bit of algebra, but we can derive updated expressions for $VV^\top$ and $\mathbf{v}$ in terms of solely the new covariance matrix, $K_{\mathbf{vu}}$ and the new responses, $\Sigma_{\mathbf{y}}^{-1}\mathbf{y}$. That is, $\mathbf{v}_{\text{SGPR}} = K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}} \mathbf{v}$ and $VV^\top_{\text{SGPR}} = K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}} VV^\top K_{\mathbf{au}} K_{\mathbf{uu}}^{-1}$.

$$\begin{aligned} \mathbf{v}_{\text{SGPR}} &= K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}} (\Sigma_{\mathbf{y}}^{-1} \mathbf{y} - \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}} A K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1} \mathbf{y}) \\ &= K_{\mathbf{uu}}^{-1} \mathbf{c} - K_{\mathbf{uu}}^{-1} C (K_{\mathbf{uu}} + C)^{-1} \mathbf{c} \\ &= K_{\mathbf{uu}}^{-1} ((K_{\mathbf{uu}} + C)(K_{\mathbf{uu}} + C)^{-1} - C (K_{\mathbf{uu}} + C)^{-1}) \mathbf{c} \\ &= (K_{\mathbf{uu}} + C)^{-1} \mathbf{c} \end{aligned}$$

and similarly

$$
\begin{aligned}
VV^\top_{\text{SGPR}} &= K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}} VV^\top K_{\mathbf{au}} K_{\mathbf{uu}}^{-1} \\
&= K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}} (\Sigma_{\mathbf{y}}^{-1} - \Sigma_{\mathbf{y}}^{-1} K_{\mathbf{au}} A K_{\mathbf{ua}} \Sigma_{\mathbf{y}}^{-1}) K_{\mathbf{au}} K_{\mathbf{uu}}^{-1} \\
&= K_{\mathbf{uu}}^{-1} K_{\mathbf{uu}} - K_{\mathbf{uu}}^{-1} C A C K_{\mathbf{uu}}^{-1} \\
&= K_{\mathbf{uu}}^{-1} (K_{\mathbf{uu}} (K_{\mathbf{uu}} + C)^{-1} C K_{\mathbf{uu}}^{-1}) \\
&= (K_{\mathbf{uu}} + C)^{-1} C K_{\mathbf{uu}}^{-1} = (K_{\mathbf{uu}} + K_{\mathbf{uu}} C^{-1} K_{\mathbf{uu}})^{-1} \\
&= K_{\mathbf{uu}}^{-1} - (K_{\mathbf{uu}} + C)^{-1}.
\end{aligned}
$$

Similarly one could follow this logic in reverse to go from SGPR caching to caching for exact GP inference. We can also use the updates in Eq. B.4 to update the exact GPs caches via first updating the SGPR caches.

To summarize, exact GP regression is just Gaussian conditioning, which can be viewed as a special case of SGPR if one inducing point is placed at every data point. SGPR in turn is again Gaussian conditioning through an approximate kernel on projected features, which can be viewed as a special case of O-SGPR if the inducing points and kernel hyperparameters are held fixed. Finally O-SGPR can be viewed as a special case of O-SVGP if the variational parameters are constrained to be optimal.

B.3.2 Interpreting Bui et al. (2017a) as O-SGPR

The approach outlined in Section 2.7.1 can be verified to be equivalent to streaming sparse GPs (e.g. the un-collapsed bound of Bui et al. (2017a)) mechanically by verifying that the expressions for the ELBO are equivalent (up to constants) and that the predictive mean and variance are exactly equivalent. Although verifying the equivalence is simply a matter of manipulating algebraic expressions, we have not yet justified the choice of $\hat{\mathbf{y}}$ and $\Sigma_{\hat{\mathbf{y}}}$. Bui et al. (2017a) obtained their expressions by means of variational calculus, and arrived at the correct result, but did not provide much in the way of intuition for the nature of the optimal solution.

We now show how the choice of pseudo-targets $\hat{\mathbf{y}}$ and pseudo-likelihood covariance $\Sigma_{\hat{\mathbf{y}}}$ has a clear interpretation that obviates any need to appeal to variational calculus except as a formal guarantee of optimality. Again, suppose we are given a sparse variational GP with inducing points Z', kernel hyperparameters θ' and a pre-computed optimal variational distribution $\phi(\mathbf{u}') = \mathcal{N}(\mathbf{m}_{\mathbf{u}'}, S_{\mathbf{u}'})$, and then asked to find the likelihood and dataset of size p that produced the model. Although the problem as stated is under-determined, if we choose $X = Z'$ and assume the likelihood is some Gaussian centered at f, then we can reverse Eqn. (2.20) (in the main text) to solve for $\mathbf{y}$ and $\Sigma_{\mathbf{y}}$ as follows:

$$\begin{aligned}\mathbf{m}_{\mathbf{u}'} &= K'_{\mathbf{u}'\mathbf{u}'}(K'_{\mathbf{u}'\mathbf{u}'} + K'_{\mathbf{u}'\mathbf{u}'}\Sigma_{\mathbf{y}}K'_{\mathbf{u}'\mathbf{u}'})^{-1}K'_{\mathbf{u}'\mathbf{u}'}\Sigma_{\mathbf{y}}^{-1}\mathbf{y}, \\ &\qquad \Rightarrow \mathbf{y} = (\Sigma_{\mathbf{y}}K'^{-1}_{\mathbf{u}'\mathbf{u}'} + I)\mathbf{m}_{\mathbf{u}'} = \hat{\mathbf{y}} \qquad \text{(B.6)} \\ S_{\mathbf{u}'} &= K'_{\mathbf{u}'\mathbf{u}'}(K'_{\mathbf{u}'\mathbf{u}'} + K'_{\mathbf{u}'\mathbf{u}'}\Sigma_{\mathbf{y}}K'_{\mathbf{u}'\mathbf{u}'})^{-1}K'_{\mathbf{u}'\mathbf{u}'}, \\ &\qquad \Rightarrow \Sigma_{\mathbf{y}} = (S^{-1}_{\mathbf{u}'} - K'^{-1}_{\mathbf{u}'\mathbf{u}'})^{-1} = \Sigma_{\hat{\mathbf{y}}}. \qquad \text{(B.7)}\end{aligned}$$

As a result, we can now provide new, intuitive interpretations of $\hat{\mathbf{y}}$, $\Sigma_{\hat{\mathbf{y}}}$ and $\phi(\mathbf{u})$. In simple terms, the streaming sparse GP (i.e. O-SGPR) of Bui et al. (2017a) is equivalent to a sequence of SGPR models, where instead of training on all previously observed data through the original likelihood at each timestep, each model trains *only* on the combination of the current batch of data $(X_{\text{batch}}, \mathbf{y})$, and the pseudo-data $(Z', \hat{\mathbf{y}})$ through a pseudo-likelihood with covariance $\Sigma = \text{blkdiag}(\Sigma_{\hat{\mathbf{y}}}, \Sigma_{\mathbf{y}})$. The pseudo-data and pseudo-likelihood together represent all the past data and models. Furthermore, $(Z', \hat{\mathbf{y}})$ and $\Sigma_{\hat{\mathbf{y}}}$ are the *unique* size-p dataset with $X = Z'$ and f-centered Gaussian likelihood that could have produced $\phi(\mathbf{u}')$, given Z' and θ'. In other words we can think of the tuple $(\theta', Z', \hat{\mathbf{y}}, \Sigma_{\hat{\mathbf{y}}})$ as a compressed representation of the sparse GP it defines.

B.3.3 Practical Implementation

Implementation wise and to reduce our engineering overhead, we focused on computing $\hat{\mathbf{y}}$ and $\Sigma_{\hat{\mathbf{y}}}$ in a numerically stable manner. We start with the pseudo-covariance term, which can be simplified as

$$\Sigma_{\hat{\mathbf{y}}} = I + S_{\mathbf{u}'}(K'_{\mathbf{u}'\mathbf{u}'} - S_{\mathbf{u}'})^{-1} S_{\mathbf{u}'}.$$

After some more algebra, we can rewrite the pseudo-observations that depend on the inducing points,

$$\begin{aligned}\hat{\mathbf{y}} &= \Sigma_{\hat{\mathbf{y}}} S_{\mathbf{u}'}^{-1} \mathbf{m}_{\mathbf{u}'} = \left(I + S_{\mathbf{u}'}(K'_{\mathbf{u}'\mathbf{u}'} - S_{\mathbf{u}'})^{-1} S_{\mathbf{u}'}\right) S_{\mathbf{u}'}^{-1} \mathbf{m}_{\mathbf{u}'} \\ &= S_{\mathbf{u}'}^{-1} \mathbf{m}_{\mathbf{u}'} + S_{\mathbf{u}'}(K'_{\mathbf{u}'\mathbf{u}'} - S_{\mathbf{u}'})^{-1} \mathbf{m}_{\mathbf{u}'} \end{aligned} \tag{B.8}$$

For numerical stability, we substitutes inverses of matrix subtractions with squared systems, i.e.

$$(K - S)^{-1} = \left((K - S)^{-1}(K - S)^{-\top}\right)(K - S)^{\top},$$

dropping subscripts. While there is still a matrix subtraction here, squaring the system improves the numerical stability of the systems, as we are forcing all of the eigenvalues of the systems to be non-negative.

In practice however, we use "whitening" of the variational distribution as introduced by (Matthews, 2017). We instead optimize $\bar{\mathbf{m}}_{\mathbf{u}} = K_{\mathbf{uu}}^{-1/2} \mathbf{m}_{\mathbf{u}}$ and $\bar{S}_{\mathbf{u}} = K_{\mathbf{uu}}^{-1/2} S_{\mathbf{u}} K_{\mathbf{uu}}^{-1/2}$. We can rewrite $\Sigma_{\hat{\mathbf{y}}}$ using the whitened variational covariance matrix $\bar{S}_{\mathbf{u}'}$ producing, $\Sigma_{\hat{\mathbf{y}}} = K'^{1/2}_{\mathbf{u}'\mathbf{u}'}(\bar{S}_{\mathbf{u}'} + \bar{S}_{\mathbf{u}'}(I - \bar{S}_{\mathbf{u}'})^{-1} \bar{S}_{\mathbf{u}'}) K'^{1/2}_{\mathbf{u}'\mathbf{u}'}$. Again, we square the second term to enhance stability, although it already has a symmetric form; that is, we compute

$$\Sigma_{\hat{\mathbf{y}}} = K'^{-1/2}_{\mathbf{u}'\mathbf{u}'}(\bar{S}_{\mathbf{u}'} + \bar{S}_{\mathbf{u}'}(I - \bar{S}_{\mathbf{u}'})^{-1}(I - \bar{S}_{\mathbf{u}'})^{-\top}(I - \bar{S}_{\mathbf{u}'})^{\top} \bar{S}_{\mathbf{u}'}) K'^{-1/2}_{\mathbf{u}'\mathbf{u}'}.$$

Similarly, Eq. B.7 simplifies to become

$$\hat{\mathbf{y}} = K'^{1/2}_{\mathbf{u}'\mathbf{u}'}(I - \tilde{S}_{\mathbf{u}'})^{-1}\bar{\mathbf{m}}_{\mathbf{u}'} = K'^{1/2}_{\mathbf{u}'\mathbf{u}'}(I - \tilde{S}_{\mathbf{u}'})^{-1}(I - \bar{S}_{\mathbf{u}'})^{-\top}(I - \bar{S}_{\mathbf{u}'})^{\top}\bar{\mathbf{m}}_{\mathbf{u}'}.$$

From an engineering point of view, as we condition into an exact GP for most of our applications, we are able to use the pseudo-likelihood covariance alongside the cache of the pseudo-data covariance, caching a root decomposition of the matrix $(K_{\text{joint}} + \texttt{blkdiag}(\Sigma_{\hat{\mathbf{y}}}, \Sigma_{\mathbf{y}}))^{-1}$ via low-rank updates to a pre-existing root decomposition of $K_{\mathbf{uu}} + \Sigma_{\hat{\mathbf{y}}}$ (and its inverse), where $K_{\text{joint}} = k_\theta(\texttt{cat}(Z, X_{\mathbf{b}}), \texttt{cat}(Z, X_{\mathbf{b}}))$ (Pleiss et al., 2018b; Jiang et al., 2020a). On performing several steps of conditioning (e.g. rollouts), we then use exact GP conditioning via low rank updates as implemented in GPyTorch (which uses the strategy of Jiang et al. (2020a) internally) after the first step.

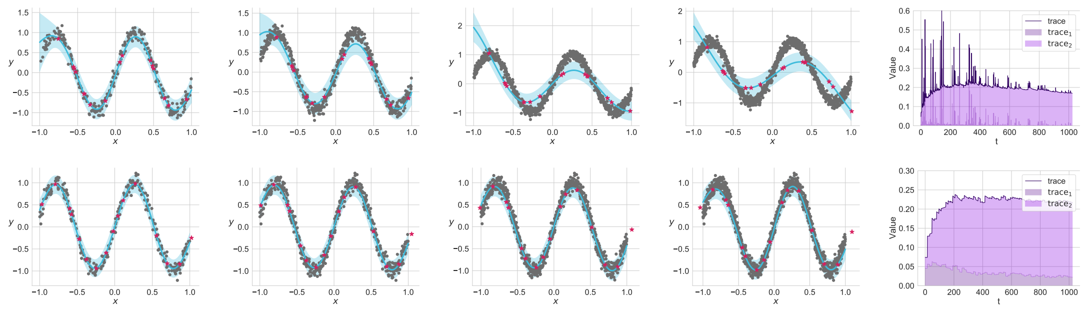
B.3.4 Incremental O-SGPR Tends to Underfit the Data

In this section we discuss a pathology of the ELBO derived in Bui et al. (2017a) that occurs when O-SGPR is updated on very small batches on new data (e.g. 1 new observation). Tellingly, Bui et al. (2017a) only considered tasks with large batch sizes (around 100 new observations per batch in each task).

Bui et al. (2017a) propose a "collapsed" evidence lower bound to train their O-SGPR model for each new batch of data. It is written as:

$$\begin{aligned}
\ell_{\mathrm{OSGPR}}(\theta, \mathbf{u}) &= -\log \mathcal{N}(\texttt{cat}(\hat{\mathbf{y}}, \mathbf{y}) | \mathbf{0}, Q_{\mathrm{joint}} - \texttt{blkdiag}(\Sigma_{\hat{\mathbf{y}}}, \Sigma_{\mathbf{y}})) \\
&\qquad + \frac{1}{2}(\texttt{trace}_1(\theta, \mathbf{u}) + \texttt{trace}_2(\theta, \mathbf{u})) + \text{constants}, \qquad \text{(B.9)} \\
\texttt{trace}_1(\theta, \mathbf{u}) &= \sigma^{-2} \mathrm{Tr}(K_{\mathbf{aa}} - K_{\mathbf{au}} K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}}), \\
\texttt{trace}_2(\theta, \mathbf{u}) &= \mathrm{Tr}((S_{\mathbf{u}'}^{-1} - K_{\mathbf{u}'\mathbf{u}'}'^{-1})(K_{\mathbf{u}'\mathbf{u}'} - K_{\mathbf{u}'\mathbf{u}} K_{\mathbf{uu}}^{-1} K_{\mathbf{uu}'})),
\end{aligned}$$

where $Q_{\mathrm{joint}} = [K_{\mathbf{uu}}, K_{\mathbf{ua}}]^\top K_{\mathbf{uu}}^{-1} [K_{\mathbf{uu}}, K_{\mathbf{ua}}]$, and constants is composed of terms that do not depend on θ or Z. In a close parallel to the observations of Titsias (2009a) (e.g. Eq. B.2), we see that the O-SGPR objective is composed of a likelihood term and a trace term (written as $\texttt{trace}_1 + \texttt{trace}_2$), the latter acting as a regularizer (Titsias, 2009a; Bauer et al., 2016). The first part of the trace term, $\texttt{trace}_1$, has the same interpretation as the trace term in the batch setting — it is minimized when the Nÿstrom approximation of the kernel matrix at $X_{\mathbf{b}}$ is exact (i.e. $K_{\mathbf{aa}} = K_{\mathbf{au}} K_{\mathbf{uu}}^{-1} K_{\mathbf{ua}}$). The second trace term, $\texttt{trace}_2$, is minimized when $Z' = Z$, so it regularizes the new inducing point locations to be close to the old locations. If the batch size q is much less than the number of inducing points p then the trace term is dominated by $\texttt{trace}_2$, which is after all a sum over p terms, compared to $\texttt{trace}_1$ which is a sum over q terms. Since the loss encourages the model to keep Z' close to Z to minimize $\texttt{trace}_2$, the model can simply increase σ^2 to also decrease $\texttt{trace}_1$ to explain new observations by under-fitting. An analogous problem for small batch sizes was



(a) $T = 256$ **(b)** $T = 512$ **(c)** $T = 768$ **(d)** $T = 1024$ **(e)** Trace term

Figure B.1: O-SGPR ($p = 16$) learning on i.i.d observations without re-sampling the inducing points from a noisy sine function. **Top row:** As the number of data points increases for a batch size of 1, the model progressively underfits due to excess regularization. The trace term is entirely dominated by trace$_2$. **Bottom row:** For a larger batch size ($q = 16$), there is no under-fitting and trace$_1$ takes up more of the overall trace term.

described in the Appendix of Stanton et al. (2021) for the un-collapsed bound (e.g. the training procedure of an O-SVGP model) of Bui et al. (2017a), necessitating Stanton et al. (2021) to propose a variant that down-weights the prior terms in the objective. The cost of down-weighting is an increased tendency towards over-fitting as well as an additional hyper-parameter, both of which were observed by Stanton et al. (2021).

In Figure B.1, we empirically demonstrate the pathology of the O-SGPR bound with small batch sizes in the i.i.d. setting. In the top row, we add $q = 1$ data point at a time while continuing to re-train. Although the O-SGPR model originally fits the data well at $T = 256$, it progressively begins under-fitting, which becomes more and more noticeable, especially by $T = 1024$. By comparison, a larger batch size, $q = p = 16$, prevents any under-fitting from occurring. In the far right panel, we see the two terms in the trace component; in the small batch setting, the inducing trace term dominates the total trace.

```
Algorithm 6: LTS with SVGP
Input: Observed data $\mathcal{D} = (X, \mathbf{y})$; SVGP model, $\mathcal{M}$; candidate set generation utility, $C_{\text{gen}}()$, rollout steps, $T$, parallel path parameter $l$, top $k$ parameter, $q$ batch size.
1. Generate initial candidate set, $X_1 = C_{\text{gen}}()$. 2. Compute posterior over candidate set, drawing a posterior sample: $y_1 \sim p(y|X_1, \mathcal{M})$. 3. Sort $y_1$ and keep top $l$ samples $\tilde{y}_0$ and corresponding candidates, $\tilde{X}_1$. 4. Generate $\mathcal{M}_1 \leftarrow$ OVC($M, (\tilde{X}_i, \tilde{y}_i)$.unsqueeze(-1)) via Algorithm 1. $\mathcal{M}_1$ is a batch of $l$ models each conditioned on a single data point. for t in 2:T do
    5. Generate initial candidate set, $X_t = C_{\text{gen}}()$. 6. Compute posterior over candidate set, drawing a posterior sample: $y_t \sim p(y|X_t, \mathcal{M}_{t-1})$. 7. Sort $y_t$ and keep top $l$ samples $\tilde{y}_t$ and corresponding candidates, $\tilde{X}_t$. 8. Generate $\mathcal{M}_t \leftarrow$ OVC($\mathcal{M}_{t-1}, (\tilde{X}_t, \tilde{y}_t)$) using Algorithm 2.
9. Generate $\mathcal{M}_{\text{end}} \leftarrow$ OVC($\mathcal{M}, (\tilde{X}_i, \tilde{y}_i)_{i=1}^{T}$) using Algorithm 2. Generate final candidate set, $X_{\text{end}} = C_{\text{gen}}()$. 10. Compute posterior over candidate set, drawing a posterior sample: $y_{\text{end}} \sim p(y|X_{\text{end}}, \mathcal{M})$. 11. Sort $y_{\text{end}}$ and return top $q$ corresponding candidates, $\bar{\mathbf{X}}_{\text{end}}$. return Candidates for evaluation $\bar{\mathbf{X}}_{\text{end}}$.
```

The effect is mediated by a larger batch size, as shown in the bottom right panel.

Informally, if the number of old inducing points is much greater than the number of new observations, then O-SGPR will focus on replicating the old variational distribution. Note that either aggregating multiple batches for each update or conditioning O-SGPR into an exact GP remedies the issue.

B.3.5 Look-Ahead Thompson Sampling

Thompson sampling in continuous domains: in the context of black-box optimization, the action space is simply the input space, $\mathcal{X}$, since we are deciding which input $\mathbf{x} \in \mathcal{X}$ we will query next. Thompson sampling draws the next query point from the Bayes-optimal distribution over the possible choices, $\mathbf{x}^* \sim p_{\mathrm{TS}}(\mathbf{x})$, where

$$p_{\mathrm{TS}}(\mathbf{x}) \propto \int \mathbb{1}\{f(\mathbf{x}) = \sup_{\mathbf{x}' \in \mathcal{X}} f(\mathbf{x}')\} p(f|\mathcal{D}) df. \tag{B.10}$$

When $\mathcal{X}$ is a continuous domain, as is usually the case, we replace the $\sup_{\mathbf{x}' \in \mathcal{X}}$ with a $\max_{\mathbf{x}' \in X_{\mathrm{cand}}}$, where $X_{\mathrm{cand}} \subset \mathcal{X}$. As the name suggests, rather than attempting to evaluate the integral in Eq. (B.10), Thompson sampling instead draws samples $f_i \sim p(f|\mathcal{D})$, $i \in \{1, \dots, q\}$ and take $\mathbf{x}_i^* = \mathrm{argmax}_{\mathbf{x}' \in X_{\mathrm{cand}}} f_i(\mathbf{x}')$.

The look-ahead case: If we were able to evaluate each f_i on every point $\mathbf{x} \in \mathcal{X}$ and compute $\sup_{\mathbf{x}' \in \mathcal{X}} f_i(\mathbf{x}')$ exactly, there would be no benefit to multiple rounds of Thompson sampling. However, as we noted above, typically we rely on a max over a discrete set X_{cand}, typically obtained from a (quasi-)Monte Carlo method (e.g. Sobol sequences) to cover $\mathcal{X}$, and the number of candidate points is restricted by compute and memory. Therefore we can do multiple rounds of Thompson sampling to try to refine the estimate of $\mathbf{x}_i^*$ by evaluating $\mathrm{argmax}_{\mathbf{x}' \in X_j} f_i(\mathbf{x}')$ for a sequence of candidate sets X_j, $j \in \{0, \dots, h\}$. The key challenge with GPs is to ensure that f_i is consistent across the sequence of candidate sets, which we accomplish by drawing $f_i(\mathbf{x}') \sim p(f|\mathcal{D} \cup \{(\mathbf{x}_{j-1}^*, f_i(\mathbf{x}_{j-1}^*))\}$ for $\mathbf{x}' \in X_j$ and $j > 0$. The result is again $\mathbf{x}_i^* = \max_{\mathbf{x}' \in X_{\mathrm{cand}}} f_i(\mathbf{x}')$, but now $X_{\mathrm{cand}} = \bigcup_j X_j$.

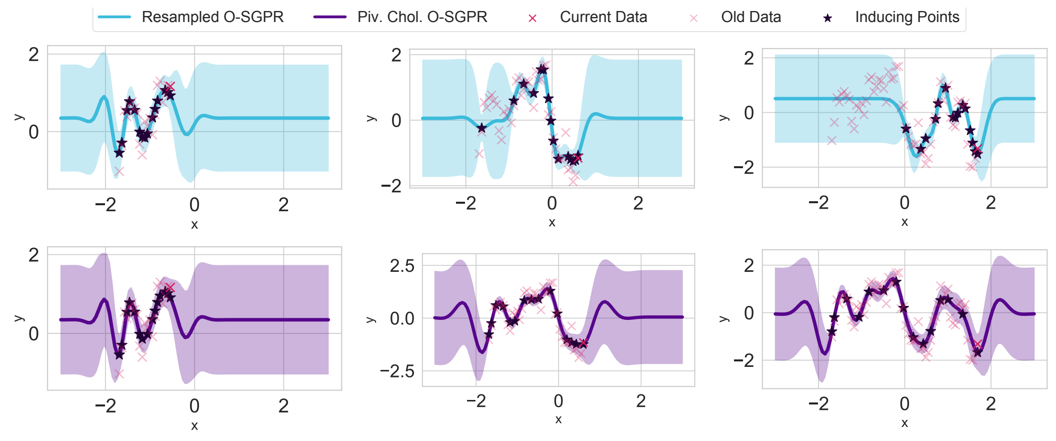
In Algorithm 6, we describe how OVC is used within LTSs as an example of its usecase. Here, of course, we are only performing Thompson sampling over a discrete set of values and so do not end up needing to use gradient based acquisitions. Specifically, we continue using OVC (or really after $T = 1$, exact GP conditioning with low-rank updates (Jiang et al., 2020a) as the GP is now exact), to condition our model on each step's fantasy responses $\tilde{y}_t$ and the observations $\tilde{X}_{\text{batch}}$.

Further Experimental Details and Results

B.4.1 Updating O-SGPR Inducing Points

We illustrate the efficacy of this choice of new inducing points in Figure B.2 using the same time series data as in Stanton et al. (2021) originally from `https://raw.githubusercontent.com/trungngv/cogp/master/data/fx/fx2007-processed.csv` (that repo uses BSD License). Re-sampling the old inducing points is shown in the top row, and tends to first perform well, but then begins to catastrophically forget by $t = 40$ and dramatically so by $t = 60$, as all of the inducing points have moved over to the right. By comparison, our approach of iteratively running a pivoted cholesky on the current inducing points and the new data point, prevents catastrophic forgetting, while also enabling the model to learn on the new data stream.

In Figure B.3, we illustrate the effect of Laplace approximations during a rollout following the streaming classification example of Bui et al. (2017a) as we use OVC. We first trained a SVGP model with 25 inducing points on 100 data points, as shown in the first rows, then performed three steps of rollouts each with 100 data points each as we observe progressively more and more of



(a) $t = 20$ **(b)** $t = 40$ **(c)** $t = 60$

Figure B.2: Online SVGP modelling a time series **Top row:** inducing points are updated by replacing an inducing point with the new location as in Bui et al. (2017a). **Bottom row:** Inducing points are updated by re-running a pivoted cholesky on both the new data point and the current inducing points. The recursive refitting procedure of the pivoted cholesky placement remedies the catastrophic overfitting and forgetting of merely resampling the inducing points.

the dataset. In the middle row, we show the predicted probability on a held-out test set; as we observe more data, the predictions become more and more confident throughout the entire region, and are un-confident in the regions where we have not observed any data. This effect is similarly observed by the predictive variances, which are high in regions where we have not seen any data, but decay as we observe each region successively. Data from `https://github.com/thangbui/streaming_sparse_gp/tree/master/data` (Apache 2.0 License).

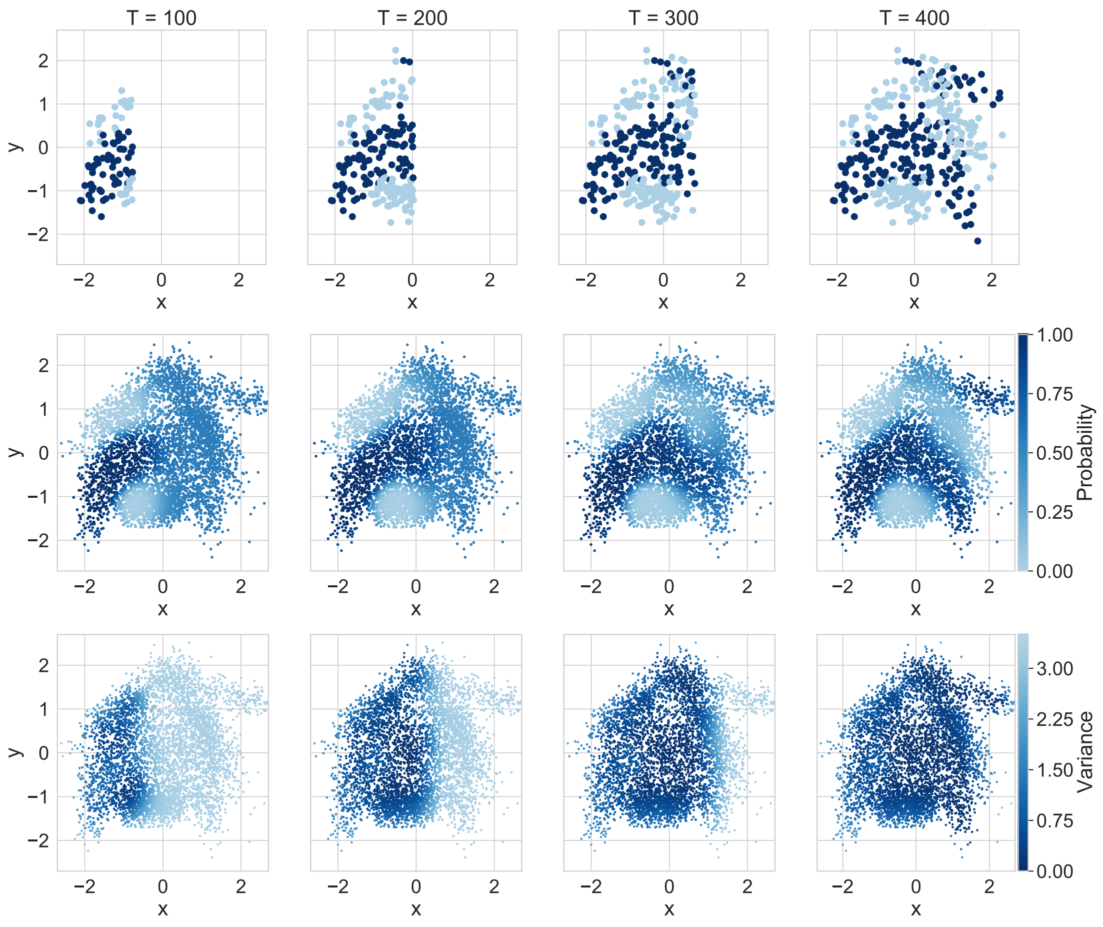


Figure B.3: Top row: Data from the bananas dataset arriving in a non-i.i.d fashion in four sucessive batches. **Middle row:** Predictive probabilities of a SVGP with OVC to rollout conditional on these batches. Even in the non-Gaussian setting, OVC is able to adapt to new data without catastrophic forgetting. **Bottom row:** Variance of the latent function during the OVC rollout. The variances decay as we observe new data.

B.4.2 Experimental and Data Details

Unless otherwise specified, all data is simulated. The code primarily relies on PyTorch (Paszke et al., 2019a) (MIT License), BoTorch (Balandat et al., 2020a) (MIT License), GPyTorch (Gardner et al., 2018b) (MIT License). All

GPs (variational and exact) used a constant mean, scaled Matern-5/2 kernels with ARD with lengthscale priors of `Gamma`$(3, 6)$ and outputscale priors of `Gamma`$(2, 0.15)$, which are the current BoTorch defaults for single task GPs. All variational GPs used GPyTorch's whitened variational strategy. Unless otherwise specified, we normalized all inputs to $[0, 1]^d$ and standardized outputs to have zero mean and standard deviation one during the model fitting stage. We trained all models to convergence with an exponential moving average stopping rule using Adam with a learning rate of 0.1. We re-fit each model independently at each iteration. When plotting the median, we plot the 95% confidence interval around it following https://www-users.york.ac.uk/~mb55/intro/cicent.htm. Unless otherwise specified, all acquisitions were optimized with multi-start L-BFGS-B with 10 random restarts and 512 samples for initialization for up to 200 iterations with a batch limit of 5 following Balandat et al. (2020a).

Understanding Experiments

All understanding experiments, e.g. Figures 2.3, B.1, B.2, B.3 were run on CPUs with Intel i5 processors. Computational costs were negligible to the cost of writing this paper.

Figure 2.3: We fit each non-Gaussian model using the ELBO. The Gaussian function is $f(x) = \sin(2|x| + x^2/2)$ with $n = 100$ and $n_{\text{test}} = 25$. For the GPCV model, we follow the model definition of Wilson & Ghahramani (2010) and parameterize the scale of the Gaussian as a linear softplus transform, but

we implemented a variational version, rather than the Laplace or MCMC implementations that are considered in the original paper. The data itself is a forward simulation of the well known SABR volatility model (Hagan et al., 2002) with parameters $F_0 = 10$, $V_0 = 0.2$, $\mu = 0.2$, $\alpha = 1.5$, $\beta = 0.9$, $\rho = -0.2$. We model the scaled log returns and plot volatility rather than the latent function. We use $n = 250$ and $n_{\text{test}} = 150$, so that $T = 400$; we standardize the inputs. Here, to perform Laplace approximations, we used PyTorch's higher order AD software as deriving the gradients and Hessians would be tedious.

Knowledge Gradient on Branin: We used the Branin test function as implemented in BoTorch (Balandat et al., 2020a) with $n = 50$ and 25 inducing points, 8 fantasies per data point, 250 candidate points and a grid of size 15×15. These were run on a single Nvidia Titan 24GB RTX. Computation took several minutes.

Incremental Learning on Protein: We followed the experimental protocol of Stanton et al. (2021) but substituted in Matern-5/2 kernels with ARD instead of linear projections. The data comes from Dua & Graff (2017). The experiment is run over 10 random seeds and we show the mean and two standard deviations of the mean. Computation took several hours per trial.

Batch Knowledge Gradient Experiments

qEI and qNEI optimization used quasi Monte Carlo (QMC) integration with 256 random samples, while qKG optimization used QMC integration with 64 random samples (BoTorch defaults).

Hartmann6: Data comes from the Hartmann6 test function from https://github.com/pytorch/botorch/blob/master/botorch/test_functions/synthetic.py and the experiment is inspired by https://botorch.org/tutorials/closed_loop_botorch_only. These were run on CPUs on an internal cluster. Computation took several hours per trial. We used and 1000 randomly sampled candidate points to estimate the knowledge gradient.

Laser: Comparison scripts are from https://github.com/ermongroup/bayes-opt/. No license was provided. These were run on CPUs on an internal cluster. Computation took several hours per trial.

Preference Learning: Function is inspired by https://botorch.org/tutorials/preference_bo; we used noise of $\sigma = 0.1$ to make the function more difficult. The Laplace implementation comes from https://github.com/pytorch/botorch/blob/master/botorch/models/pairwise_gp.py. These were run on CPUs on an internal cluster. Computation took several hours per trial. qNEI optimization used QMC integration with 128 random samples, while qKG optimization used QMC integration with 64 random samples. Here, we used 3 random restarts and 128 raw samples for acquisition function optimization.

Active Learning Experiments

Malaria: Data is originally from Weiss et al. (2019) under a creative commons 3 license, https://malariaatlas.org/malaria-burden-data-download/#FAQ. From the reference, the data is modelling predictions off of survey data and thus not human responses. For all models, we used Matern-1/2 kernels due to the lower smoothness and fixed noise models as variance is known. Comparison is to WISKI (Stanton et al., 2021), with their code `https://github.com/wjmaddox/online_gp` which uses Apache License 2.0. These were run on a combination of Nvidia 32GB V100s and 48GB RTXes on an internal cluster. Here, we used 4 random restarts with 64 base samples to optimize the aquisition. Computation took several hours per trial.

Hotspot Modelling: Simulated data and comparison data is from `https://github.com/disarm-platform/adaptive_sampling_simulation_r_functions`. No license was provided for either. Our trials were run on AMD 32GB Mi50 GPUs on an internal cluster. Computation took close to eight hours per trial. We used tempering with $\beta = 0.1$ (Jankowiak et al., 2020b) and Matern-3/2 kernels for these models following the kriging setup in Andrade-Pacheco et al. (2020). Overall, we used 16 inner and outer samples for the entropy search objective, enumerating over all remaining test points to select a new point to query.

We note that the model fitting procedure of Andrade-Pacheco et al. (2020) seems to possibly encourage test-set leakage as they seemingly use a random forest trained on all of the data, rather than on simply the first n observations. See the third from final paragraph in their description of spatial methods in that section. We do not follow this as we do not use any random forests.

TurBO Experiments

Rover: We use the opensource Turbo implementation from https://botorch.org/tutorials/turbo_1 and the rover function setup code from https://github.com/zi-w/Ensemble-Bayesian-Optimization. Both are licensed under the MIT License. These were run on Nvidia $24GB$ RTXes on an eight GPU server. We repeated experiments 24 times. See the wall-clock time panels for estimates of computational budgets (about an hour). As not all trials reached exactly $40,000$ iterations (due to cholesky decomposition errors, memory errors, and TurBO not restarting after 190 steps), we assume that the maximum achieved value was the best evaluation out to 200 steps to mimic the performance that one would see if using the method in practice. Early failures were only an issue for the exact GPs and then due to numerical instability, actually inspiring Figure B.5a. As this truncation wreaked a bit of havoc with our timings, we only report the first 150 step timings in the main text (170 for Figure B.6).

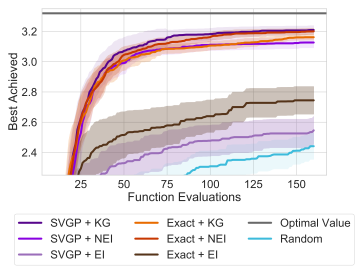
We used 500 data points and the same model classes for the conditioning experiment (Figure B.5a). Error bars are two standard deviations of the mean over the 10 paths used.

For the global models experiment (Figure B.5c), we used a combination strategy that first used half the batch with Thompson sampling (TS) to select the points and then used qGIBBON (Moss et al., 2021) to select the other half of the batch by setting the TS half of the batch as pending points. Performance using qGIBBON alone was about twice as slow and was slightly worse due to the lack of exploration that TS provides. This strategy also enforces that qGIBBON's implementation actually uses fantasization and OVC, which would not natively have been the case without using the pending points. For the timings, we report the first 106 steps over 8 seeds.

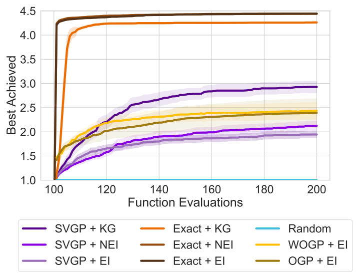
MuJoCo: We use the codebase of Wang et al. (2020) including their patched TurBO implementation with an optimization loop. This codebase is available from `https://github.com/facebookresearch/LaMCTS/tree/master/LA-MCTS` with a Creative Commons 4.0 License with the included, modified TurBO implementation following a non-commercial license. The MuJoCo experiments use mujoco-py (`https://github.com/openai/mujoco-py`, MIT License) and an institutional license key for MuJoCo itself (Todorov et al., 2012). These were run on a combination of Nvidia 32GB V100s and 48GB RTXes on an internal cluster. We repeated these experiments over 10 trials and computation took close to 14 hours for hopper (where one trial failed to reach 4000 samples for all methods but exact LTS), and several hours for swimmer.

On hopper, we struggled with wide variation in model fits, so we changed the regularization strategy on the base GP models to account for the high dimensional feature space. Inspired by Eriksson & Jankowiak (2021), we continued using ARD Matern-5/2 kernels but placed `HalfCauchy`(τ) priors on the inverse lengthscales and then placed a `HalfCauchy`(1.0) prior on τ itself. Rather than using MCMC as Eriksson & Jankowiak (2021) did, we used MAP to estimate both the lengthscales and τ.

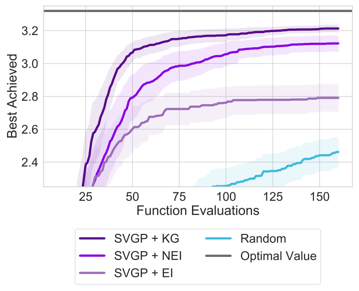
B.4.3 Bayesian Optimization with the Knowledge Gradient



(a) Hartmann6.



(b) Laser.



(c) Poisson-Hartmann6.

Figure B.4: (a) Comparison to a broader suite of methods on Hartmann6, 1 constraint. **(b)** Comparison includng BoTorch exact GPs and their acquisitions on the free electron laser problem. Comparing to the results in Figure 3 of McIntire et al. (2016), these exact GPs vastly outperform their implementation, presumably due to advances in acquisition function optimization. **(c)** Hartmann6 test problem with count responses (Poisson likelihood). Only approximate inference can be used here, and qKG vastly outperforms qNEI.

In Figure B.4, we present the results for a wider set of acquisition functions using the one-shot knowledge gradient on three test functions. These results complement Figure 2.8 and only include these additional methods. Overall, qKG (with either an exact GP or a SVGP) generally performs best, followed by qNEI and then qEI.

In Figure B.4a, we include results on the constrained Hartmann6 problem with random baselines as well as exact and SVGPs with expected improvement (EI). Exact and SVGPs with EI are signficantly outperformed by the other, more advanced acquisitions, but do outperform a random baseline.

In Figure B.4b, we also show the results of exact GPs as well as an online GP (OGP, Csató & Opper, 2002) with EI using the implementation of McIntire et al. (2016). Interestingly, all of the exact GPs significantly outperform

Table B.1: Best achieved values on Hartmann-6 for batch size, $q = 3$ for both NEI and KG for exact and SVGPs.

Noise Level	Acqf	Method	Best Value
0.1	NEI	Exact	3.19 (0.05)
0.1	NEI	SVGP	3.08 (0.06)
0.1	KG	Exact	3.17 (0.03)
0.1	KG	SVGP	3.13 (0.04)
0.5	NEI	Exact	3.20 (0.03)
0.5	NEI	SVGP	3.12 (0.06)
0.5	KG	Exact	3.15 (0.03)
0.5	KG	SVGP	3.21 (0.05)

their variational counterparts. This is quite surprising in some sense, as the true simulator is a weighted OGP with fixed hyper-parameters, and this method (WOGP + EI) performs much worse. Note that the random baseline makes no progress.

Finally, in Figure B.4c, we display the results on constrained Hartmann-6 with Poisson observations, where each method outperforms random querying, but as expected SVGP + qEI, performs worse than qNEI and qKG.

Batch Size and Noise Level Ablation: In Tables B.1 and B.2, we display the final optimization results after 150 function evaluations on the Hartmann-6 test problem for varying levels of noise and for each acquisition. These results are over 20 trials and we display the mean maximum achieved value.

Overall, KG tends to outperform NEI at both low and noise levels, with both exact and SVGP models performing very similarly overall with the SVGPs getting a slight edge in the high noise setting. Furthermore, larger batch sizes tend to perform slightly better as the mean maximum achieved tends to have lower variation. Finally, higher noise levels tend to be somewhat harder to optimize as expected.

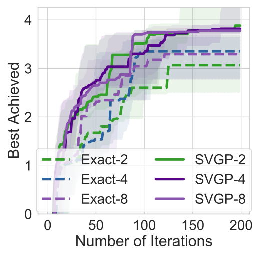
As we add more Gaussian noise into the function, we might a priori expect that qNEI should outperform qKG given the class of models. However, all things being held equal, a model that is more robust to the observed noise should tend to perform better, particularly if we are using more than just its mean and variance. Thus, SVGP models, by virtue of having more parameters to tune, tend to be more robust to the observed noise than the exact GPs.

Table B.2: Best achieved values on Hartmann-6 for batch size, $q = 1$ for both NEI and KG for exact and SVGPs.

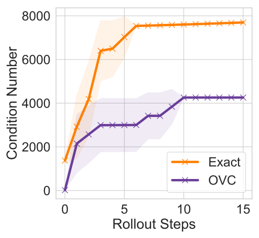
Noise Level	Acqf	Method	Best Value
0.1	NEI	Exact	3.14 (0.13)
0.1	NEI	SVGP	3.10 (0.10)
0.1	KG	Exact	3.20 (0.02)
0.1	KG	SVGP	3.18 (0.03)
0.5	NEI	Exact	3.10 (0.10)
0.5	NEI	SVGP	3.10 (0.14)
0.5	KG	Exact	3.12 (0.04)
0.5	KG	SVGP	3.16 (0.05)

B.4.4 Ablations on Rover

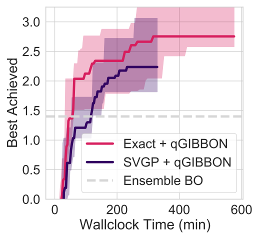
Effects of LTS: To ablate the effects of rollouts and improved conditioning, we consider several step rollouts on the rover function as shown in Figure B.5a. We find that performance is similar across depths. However, the conditioning of the resulting training data covariance is vastly improved when using OVC, as shown in Figure B.5b. Taking these two results together, we see that



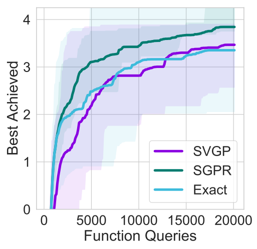
(a) LTS tree depths.



(b) Conditioning.



(c) Wall-Clock time.

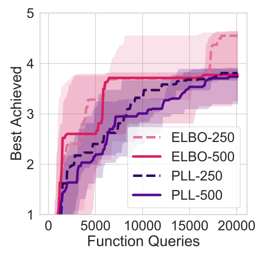


(d) SGPR ablation.

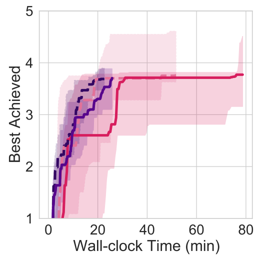
Figure B.5: **(a)** Conditioning of exact and OVC conditioning across LTS tree depths on rover. **(b)** Performance of different rollout depths is similar, as the bigger gains are conditioning. **(c)** Time efficiency using global models. SVGPs are still a strong baseline. **(d)** Comparison with SGPR using pivoted cholesky initializaiton on rover. SGPR is competitive.

in most cases, a tree depth of 4 should be enough to gain the improvements from rollouts without increasing the conditioning of the system too much.

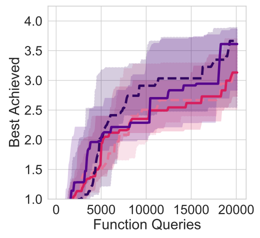
Global models and SGPR: To further demonstrate time efficiency of using OVC in the context of even global models, we perform large batch BO with the recently introduced qGIBBON acquisition (a max value entropy search variant) but using half the batch with TS to enforce fantasization over the TS-acquired batch (Moss et al., 2021). This strategy is significantly slower than TurBO; however, even in this setting using SVGPs is twice as fast, and achieves a similar result to the exact model, as shown in Figure B.5c. Both are orders of magnitude faster than Ensemble BO, which uses batch max value entropy search and exact GPs with addtive kernels (Wang & Jegelka, 2017; Wang et al., 2018). Ensemble BO takes at least several days of compute time (Eriksson et al., 2019). Our result here compares very favorably to the (un-timed) results using SVGPs as well as exact GPs with ARD kernels that Wang et al. (2018) also compared to, as neither of those methods reached reward values ≥ 1 on this problem, even after $35,000$ steps.



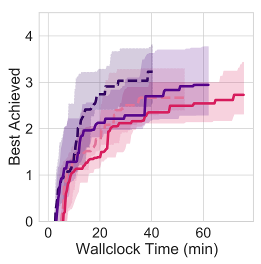
(a) TS data efficiency.



(b) TS time.



(c) LTS data efficiency.



(d) LTS time.

Figure B.6: Inducing point ablation on rover **(a,b)** use Thompson sampling and **(c,d)** use rollouts.

Number of Inducing Points and Training Loss: Finally, in Figure B.6 we ablate between training SVGPs with either 250 or 500 inducing points as well as the training loss, either the evidence lower bound (ELBO) or the predictive log likelihood (PLL), (see Appendix B.2.3 for further descriptions) on the $d = 60$ rover problem. We find that there is not a significant amount of difference between any of the four approaches whether using the ELBO or the PLL. In general, the PLL approaches are somewhat more quick to train (Figures B.6b and B.6b) in comparison to the ELBO models. They also tend to slightly outperform the ELBO-trained SVGPs when using LTS (Figure B.6c) in terms of function efficiency, but perform similarly for standard Thompson sampling (Figure B.6a). We leave a detailed benchmarking of these methods in the context of downstream tasks for future work.

Appendix C: LaMBO Supplementary Material

In Appendix C.1 we outline the four evaluation tasks: **Bigrams**, **logP + QED**, **DRD3 docking + SA**, and **Stability + SASA**. In Appendix C.2, we describe in detail the neural network architecture, the DKL GP implementation, the denoising autoencoder implementation, the string kernel implementation, and the hyperparameters used for our experiments. Finally, in Appendix C.3, we present additional experimental results to supplement the figures in the main text.

Evaluation Task Details

C.1.1 Bigrams Task

The bigrams task is a simple toy example of discrete sequence optimization. We draw random strings from an alphabet $\mathcal{V}$ and count the occurrence of k predetermined bigrams, which we use as proxy fitness targets. The task is to maximize the counts of each bigram in the sequence, restricting the sequence length to 36 tokens including utility tokens (”[PAD]”, ”[CLS]”, ”[SEP]”, ”[UNK]”, ”[MASK]”). For our experiments we used the same amino acid vocabulary as our protein task and chose 3 complementary bigrams, “AV”, “VC” and “CA”. The initial sequences were sampled with lengths between 32 and 36 tokens We ensured there were an equal number of positive examples (sequences with at least one occurrence of one of the bigrams) as negative examples in the starting pool.

C.1.2 logP + QED Task

The original ZINC logP optimization task, popularized in the BayesOpt community by Gómez-Bombarelli et al. (2018), is to optimize the octanol-water partition coefficient of a small molecule. Molecules with high logP values are hydrophobic and molecules with low values are hydrophilic. Hydrophobicity can be desirable for absorption and solubility, for example in pharmaceuticals. As a property that is easy to calculate, it has risen to prominence despite being undesirable on its own. Very high logP can result in molecules with limited practical application, and moreover finding molecules with high logP reduces trivially to the problem of finding long hydrocarbon chains, as these compounds are extremely hydrophobic relative to the size of the molecule. The *penalized* logP objective adds auxiliary terms measuring synthetic accessibility (Ertl & Schuffenhauer, 2009) and the number of cycles. Unfortunately these terms do not fix the underlying problem, and so penalized logP is similarly vulnerable to optimization hacking, as we discuss in Section C.3.2.

Because logP in itself is a deeply flawed objective, both in its relevance to real-world drug design and its ability to be hacked by optimizers, we also consider a multi-objective optimization task that is closer in form to real design problems. Instead of solely optimizing for logP, we jointly optimize for logP and QED (Quantitative Estimate of Druglikeness), a composite metric that captures many elements of druglikeness, with bioaavailability among the most prominent Bickerton et al. (2012). Unfortunately QED has its own limitations as an objective, since it is a simple parametric model trained on a small dataset to emulate heuristics such as Lipinski's Rule of Five (Lipinski et al., 1997).

The shortcomings of objectives like logP and QED appear to be well-known (Nigam et al., 2019; Coley et al., 2020; Fu et al., 2020; Tripp et al., 2021; Maus et al., 2022), but superior alternatives have not yet been accepted by the research community. For example, at the time of writing the only molecule generation benchmark in TorchDrug is maximization of QED and logP of ZINC-like molecules.[*] Angermueller et al. (2020a) evaluated BBO algorithms on a substantial number of *in silico* sequence design tasks, however the large molecule tasks they considered were relatively simple, single-objective problems (e.g. maximization of the likelihood of a hidden Markov model). The vacuum of rigorous *in silico* evaluation tasks for large-molecule design motivated us to propose our RFP task as a new benchmark.

We construct the start pool by inverting the scores and selecting the top-k

[*]https://torchdrug.ai/docs/benchmark/generation.html

non-dominating sequences (i.e. we found the k *most* dominated sequences in the ZINC dataset w.r.t. logP and QED). Constructing the task in this way is better than simply sampling randomly from ZINC because QED is bounded above by 1 and many ZINC sequences already score fairly close to 1. Starting with dominated sequences ensures that there is sufficient headroom for improvement to observe variations in optimizer behavior. We capped the max sequence length at 128 SELFIES tokens, including utility tokens. The SELFIES vocabulary was precomputed from the entire ZINC dataset (Krenn et al., 2020).

C.1.3 DRD3 Docking + SA Task

We use the DRD3 docking score oracle from Huang et al. (2021), and the following quotation is reproduced from `https://tdcommons.ai/benchmark/docking_group/overview`:

" Docking is a theoretical evaluation of affinity between a ligand (a small molecular drug) and a target (a protein involved in the disease). As a molecule with higher affinity is more likely to have higher bioactivity, docking is widely used for virtual screening of compounds (Lyu et al., 2019). "

Because optimizing solely for docking may produce molecules that are difficult to synthesize, we also optimize for synthetic accessibility (SA) (Ertl & Schuffenhauer, 2009).

To construct the start pool for this task we selected 512 molecules from ZINC uniformly at random and labeled them with the objective oracles. We then used the same start pool and SELFIES encodings for all experimental trials and all optimization methods.

C.1.4 Stability + SASA Task

In this work we present a new *in silico* benchmark task designed to simulate searching for improved red fluorescent protein (RFP) variants *in vitro*, a problem of significant interest to biomedical researchers (Dance, 2021). We optimize red-spectrum proteins with known structures for stability (-dG or negative change in Gibbs free energy) and solvent-accessible surface area (SASA) (Shrake & Rupley, 1973; Cock et al., 2009) in simulation, using the FoldX suite (Schymkowitz et al., 2005) and BioPython to evaluate our objective function. Stability as evaluated by FoldX—particularly in the negative case—has been shown to correlate with protein function (Høie et al., 2021). Solvent-accessible surface area will correlate with factors that influence the brightness and photostability of the fluorescent protein: aggregation propensity due to exposed hydrophobic surface (Mishra et al., 2018) and shielding by the beta-barrel, which encapsulates the fluorophore (Chudakov et al., 2010). Since both of these benchmark tasks are functions of the protein's three-dimensional structure, it is expected that training a model on these tasks will require the model to learn a latent representation for structure, which in turn determines function.

We constructed the start pool in two phases. First we searched FPBase for all red-spectrum (defined in this context as having an emission wavelength at least 580 nm) proteins with at most 244 residues with known 3D structures, selecting the highest resolution structure if more than one was available. If more than one chain was present in the structure, we selected the longest chain as the representative residue sequence. Starting from these base proteins, we used NSGA-2 to collect additional labelled sequences to use in the start pool for subsequent experiments.

Although this task is a significant step forward for *in silico* evaluation of discrete sequence design, it is currently limited by the capabilities of FoldX, which can only compute structures from substitution mutations (i.e. the sequence length cannot change). Deep learning structure oracles such as AlphaFold (Jumper et al., 2021) or RoseTTAFold (Baek et al., 2021) could also be used, but we found FoldX to be simpler and more amenable for rapid prototyping.

C.1.5 Wet lab experimental procedure

We synthesized fluorescent proteins with PUREfrex 2.1 (Cosmo Bio LTD) in 50 uL reactions from linear DNA purchased from IDT as eBlock dsDNA gene fragments. We ran reactions at 30°C overnight in black, half-area microplates (Corning #3993) with optically clear plate adhesive and measured excitation and emission through a series of sweeps (fixing the excitation wavelength and scanning emission every 1–2 nm, or vice versa). We determined peak excitation and peak emission as the wavelength that gave maximum fluorescence units. Using NanoDSF on an Uncle instrument (Unchained Labs), we measured protein thermostability as T_m, defined as the midpoint transition temperature of the thermal melt curve.

Implementation Details

Our models are implemented in PyTorch (Paszke et al., 2019b), BoTorch (Balandat et al., 2020a), and GPyTorch (Gardner et al., 2018a). Our genetic optimizer baselines are implemented in PyMOO (Blank & Deb, 2020). Our code is publicly available at `https://github.com/samuelstanton/lambo`. Hyperparameters are summarized in Appendix C.2.4.

C.2.1 Architecture Details

We used the same base architecture for all experiments, relying on 1D convolutions (masking positions corresponding to padding tokens). We used standard pre-activation residual blocks with two conv layers, layernorm, and swish activations. We used a kernel size of 5, 64 intermediate channels and 16 latent channels.

The shared encoder and decoder each were composed of 3 of these residual blocks (for a total of 6 convolutional layers each). The shared encoder embeds input sequences with standard vocabulary and sinusoidal position embeddings. The discriminative encoder was composed of a single residual block.

Note that transformer encoder layers could be substituted as a drop-in replacement for these convolutional residual blocks, we used small convolutional layers because they are fast to train and performed adequately in our experiments.

For multi-task GPs, we chose ICM kernels for their efficiency, particularly in sampling (Bonilla et al., 2007; Maddox et al., 2021b), their flexibility and popularity.

C.2.2 DKL Implementation Details

Training DKL models is an art. Some best practices apply to both stochastic variational and exact GP inference, others are specific to the former.

Applicable to exact and variational GP inference

1. *Kernel hyperparameter priors matter.* Allowing the DKL GP to easily change both the inputs to the final conventional GP kernel (e.g. RBF) and the lengthscale of that kernel doesn't work well. We placed a tight Gaussian prior ($\sigma = 0.01$) around the initial lengthscale value and forced the encoder to learn features appropriate for that lengthscale. Note that this is distinct from simply fixing the kernel hyperparameters *a priori.*

2. *Optimizer hyperparameters matter.* Adam is really convenient to avoid too much learning rate tuning, but it can cause unexpected issues when jointly training supervised and unsupervised heads. We almost completely disabled the running estimates of the first two moments in Adam, using $\beta_1 = 0.$, and $\beta_2 = 0.01$.

3. *Normalization matters.* This is more of an issue for SVGPs than exact GPs, but in both cases batchnorm can cause undesirable and unexpected

behavior. Use layernorm.

Applicable to variational GP inference

1. *Initialization matters.* We use the procedure described in Maddox et al. (2021c) to reinitialize the inducing point locations and the variational parameters every time the model was retrained. This trick significantly improves results and saves computation, since the GP training does not completely start over every outer loop iteration.

2. One final trick that is very useful for SVGPs is to turn off gradients to all GP-related parameters every other epoch (so half the epochs are only train the encoder).

As we show in the main text and Figure C.2, DKL SVGPs can consistently be trained to similar levels of accuracy as exact DKL GPs with very little trouble, once the proper training procedures are in place. With these practical insights we were able to jointly train supervised GP heads and unsupervised language model heads on a shared encoder simply by taking one gradient step on the supervised GP loss and one gradient step on the unsupervised DAE loss per minibatch, using the same optimizer and learning rate schedule. We used diagonal Gaussian likelihoods for all our experiments, with the noise variance initialized at 0.25.

We found that DKL GPs (both exact and variational) were not immune to overfitting, so we used weight decay (1e-4) and reserved 10% of all collected data (including online queries) as validation data for early stopping.

C.2.3 DAE Implementation Details

MLM Head We used a mask ratio of 0.125 for all experiments when training MLM heads. The MLM loss is computed by randomly masking input tokens, and computing the empirical cross-entropy between the original sequence and the predictive distribution of the MLM head at the masked positions. During sequence optimization the MLM predictive distribution is modified to prevent sampling the original token (to encourage diversity) and to prevent the sampling of special tokens.

LANMT Head Our LANMT head is identical to our MLM head, except for the addition of a length prediction head and length transform module (Shu et al., 2020), a different corruption procedure and training objective. We used a max length change of 8 in our experiments for Figure C.3, so the corruption function randomly sampled a length change Δt between -8 and 8. Δt tokens were subsequently deleted, replaced, or inserted into the sequence. The corrupted sequence was forwarded through the model, which was also given the original sequence length as a label during training. A training step takes a gradient step on the cross-entropy between the predicted length and the actual length, and on the cross-entropy between the predictive distribution over the whole decoded sequence and the original sequence.

C.2.4 Hyperparameters

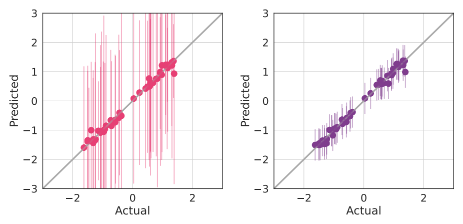
Table C.1: LaMBO hyperparameters

Sequence Optimization	
Name	Value
$\lvert\mathcal{D}_0\rvert$	512
Query batch size (b)	16
$\lvert X_{\text{base}}\rvert$	b
# Optimization rounds ($i_{\max}$)	64
# Inner loop restarts	16
# Inner loop gradient steps ($j_{\max}$)	32
Inner loop step size (η)	0.1
Entropy penalty (λ)	1e-2
# MC acquisition samples	2
Random seeds	$\{0, \dots, 9\}$

DAE Architecture	
Name	Value
Shared enc. depth (# residual blocks)	3
Disc. enc. depth (# residual blocks)	1
Decoder depth (# residual blocks)	3
Conv. kernel width (# tokens)	5
# conv. channels	64
Latent dimension	16
GP likelihood variance init	0.25
GP lengthscale prior	$\mathcal{N}(0.7, 0.01)$
# inducing points (SVGP head)	64

DAE Training	
Name	Value
DAE corruption ratio (training)	0.125
DAE learning rate (MTGP head)	5e-3
DAE learning rate (SVGP head)	1e-3
DAE weight decay	1e-4
Adam EMA params (β_1, β_2)	(0., 1e-2)
Early stopping holdout ratio	0.1
Early stopping relative tolerance	1e-3
Early stopping patience (# epochs)	32
Max # training epochs	256

C.3.1 What About Substring Kernels?



(a) SSK GP **(b)** DKL GP

Figure C.1: Evaluating the effect of kernel choice on exact GP regression with a discrete SSK **(left)** and a deep Matérn kernel with a CNN encoder **(right)** when predicting the SASA property of RFP large molecules. The SSK GP is under-confident and less accurate than the DKL GP, suggesting SSK GPs would not improve optimization performance.

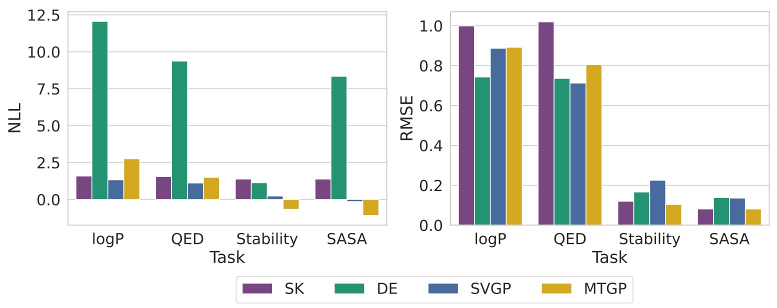


Figure C.2: Negative log-likelihood (NLL) and root mean squared error (RMSE) for various discriminative models in the offline regression setting. DKL MTGPs and SVGPs have good performance across the board, while bootstrapped CNN ensembles (DE) are very overconfident. Exact GP inference with a substring kernel (SK) is very underconfident and has poor accuracy when predicting logP and QED.

Since the start pools used in the evaluation in Figure 3.3 are fairly small (i.e. 512 examples), it is natural to wonder how a substring kernel (SSK) GP (e.g., Moss et al., 2020)) would compete with our DKL-based GPs. Due to constraints on time and computation, and SSK scalability issues, we do not directly compare SSK GPs and DKL GPs in the online setting. However, in Figure C.1, we compare SSK GPs and DKL GPs in the offline regression setting, training both models only through the supervised loss to predict the SASA property of RFP large molecules, using 410 examples for training and 102 examples for validation and test. The SSK GP performs fairly well, but is very under-confident when compared to the DKL GP. At an empirical level, our results do not support the claim that SSK GPs are superior models for biological sequences. SSKs have further drawbacks which make it hard to justify additional investment into SSK GPs for drug design.

Lack of Positional Information: at a conceptual level, SSKs cannot identify regions of the sequence that have little effect on the objective values, since matching substrings increase the prior covariance between sequences regardless of where the substrings are found. Simply put, an SSK just counts the occurrences of every possible n-gram across every possible combination of n positions in a sequence. The gap decay hyperparameter downweights occurrences corresponding to position combinations with elements that are not closely colocated. Hence SSKs have very limited positional awareness in the sense that sequences with similar n-gram counts have high prior covariance, regardless of where the n-grams actually occurred. Positional awareness is important when dealing with biological sequences from some subpopulation (e.g. a family of fluorescent proteins) since they have many identical subsequences, only varying at positions that strongly affect function.

Difficult to Scale: at a practical level, SSKs are difficult to integrate with deep generative models since they do not operate on continuous embeddings (Nigam et al., 2019), and standard methods for scaling GPs — inducing point methods (e.g., Hensman et al., 2013a), random feature expansions (e.g., Lázaro-Gredilla et al., 2010), and CG-based methods (e.g., Gardner et al., 2018a) — are difficult to apply. In particular inducing points methods are impractical because the inducing point domain would be the discrete input space, introducing a challenging discrete optimization subproblem just to train the surrogate. Furthermore SSKs struggle to scale not just to large datasets, but also to long sequences. The dynamic programming algorithm used by Moss et al. (2020) to compute their SSK is parallelizable, but becomes prohibitively memory intensive for sequences longer than 100 tokens, even when chunking the sequence into smaller pieces. In fact, we used an Nvidia RTX 8000 GPU with 48 GB of memory just to produce Figure C.3.1. We also implemented a memory-efficient trie-based SSK, which could handle longer sequences but was prohibitively slow and difficult to parallelize.

C.3.2 Comparison to LSBO for single-objective BBO

Here we evaluate LaMBO in the single-objective setting, using the *penalized* logP task described in Tripp et al. (2020). In contrast to SSK-based methods, LaMBO can successfully be scaled to large datasets with standard variational GP inference, allowing us to compare to the popular latent space BayesOpt approach (LSBO) (Gómez-Bombarelli et al., 2018) on a larger-scale problem. The start pool for this task is composed of the 2000 highest scoring sequences

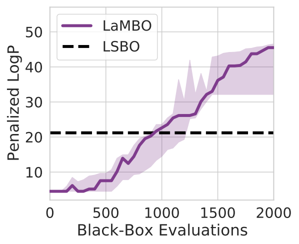


Figure C.3: LaMBO reaches a higher objective value than the best reported LSBO result. When constrained to 500 evaluations LSBO is more sample efficient (ignoring pretraining data). The midpoint, lower, and upper bounds of the LaMBO curve depict the 60%, 40%, and 80% quantiles, estimated from 5 trials.

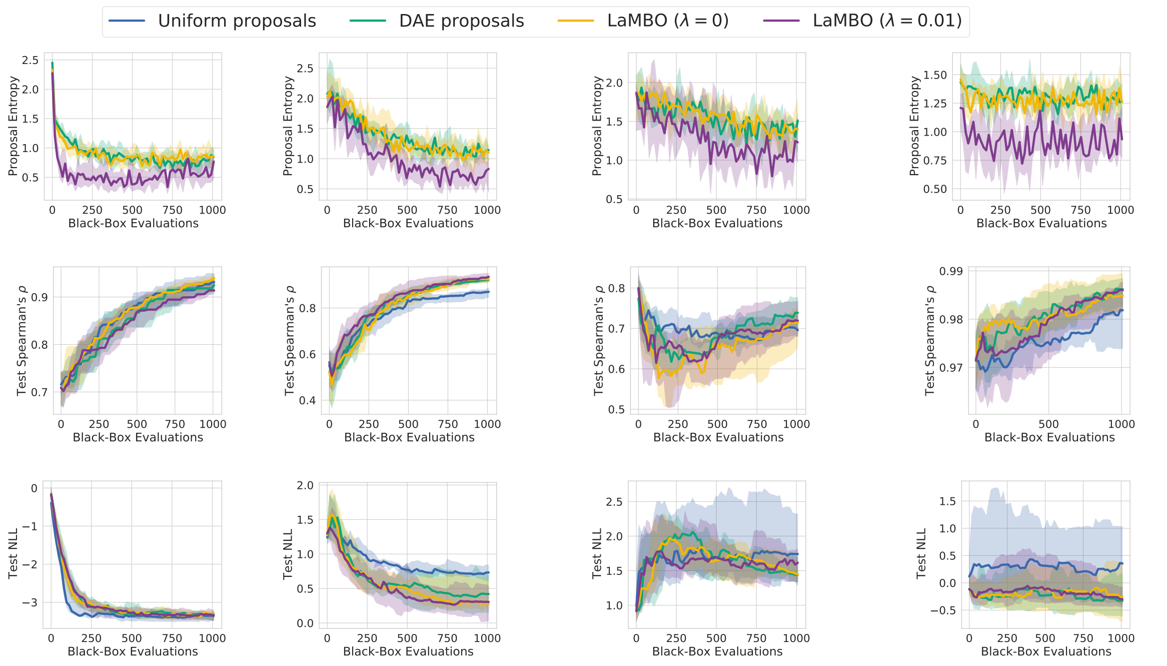
from ZINC, and 8000 random sequences, replicating the setup in Tripp et al. (2020). To accommodate the larger dataset, the discriminative head uses k independent stochastic variational GPs (SVGPs) with 64 shared inducing points rather than an exact MTGP.

In Figure C.3 we demonstrate that LaMBO is competitive with a variant of LSBO specifically designed for this task, requiring about twice as many online observations before reaching the reported median best score attained by LSBO (Tripp et al., 2020). As noted in Section 3.3, LSBO uses the entire ZINC dataset for pretraining, so we do not directly compare sample efficiency. In addition to the differences between LaMBO and LSBO already noted, we use SELFIES encodings rather than SMILES. We use a seq2seq LANMT-style decoder head for this task, since logP heavily favors large molecules. High-scoring molecules such as those found by LSBO are larger than any found in ZINC, so it is important that the optimizer allow insertions.

Overall, LaMBO outperforms the best reported LSBO score (27.84) by a wide margin, reaching scores as high as 50 for some seeds, while using a more general architecture and requiring less data (counting both pretraining data and online queries). The factor that ultimately bounds the penalized logP objective in practice is the max sequence length constraint imposed by the positional encoding scheme we use. Therefore we note that, despite its widespread use, unconstrained logP (penalized or otherwise) is a poor optimization benchmark, since it can be manipulated by altering the positional encoding to permit longer sequences (Nigam et al., 2019; Tripp et al., 2021; Fu et al., 2020).

In short, while LaMBO is designed to facilitate multi-objective optimization — a central feature of drug design — it can also outperform the widely used single-objective LSBO, even in a single-objective setting.

C.3.3 Ablating LaMBO: Proposal Entropy and Discriminative Performance



(a) Bigrams (b) logP + QED (c) DRD3 docking + SA (d) Stability + SASA

Figure C.4: Additional metrics for the ablation experiment described in Section 3.5.3 and Figure 3.4, where we cumulatively add the elements described in Section 3.4.4: (1) DAE-generated proposals, (2) DAE proposal optimization following $\nabla_Z[\ell_{\text{query}}]$ with $\lambda = 0$. (see Eq. (3.3)), and (3) DAE proposal optimization with $\lambda = 0.01$. The **top** row shows the average entropy of the generative DAE proposal distributions over time. As expected, the entropy penalty decreases the proposal entropy. The **middle** and **bottom** rows show the discriminative Spearman's ρ and NLL (averaged across objectives) on heldout data over time. Training the LaMBO architecture with both the unsupervised DAE objective and the supervised GP objective improves discriminative performance compared to the same model trained only through the supervised objective. Otherwise the methods behave similarly, verifying that better solutions are the result of better proposals, rather than a better discriminative model. The midpoint, lower, and upper bounds of each curve depict the 50%, 20%, and 80% quantiles, estimated from 10 trials.

Appendix D: Conformal BayesOpt Supplementary Material

Appendix D is structured as follows:

- In Appendix D.1, we describe limitations and broader impacts.
- In Appendix D.2, we give theoretical statements on conformal Bayesian posteriors and our implementation of the acquisition functions.
- In Appendix D.3, we give more methodological details.
- In Appendix D.5, we display more experimental results of coverage of each trial.
- In Appendix D.6, we list the hyper-parameters used for each implementation.

Limitations and Broader Impacts

D.1.1 Limitations

Marginal vs. conditional coverage guarantees: full and split conformal prediction sets have marginal coverage guarantees that are easy to confuse with conditional coverage guarantees (Angelopoulos & Bates, 2021). Marginal coverage guarantees must be interpreted with the same frequentist mindset as other frequentist measures of uncertainty, such as confidence intervals and p-values, with similar risks of misinterpretation by inexperienced users. We have attempted to make clear in the main text that marginal coverage guarantees are only realized in the aggregate, as the average of coverages observed in many independent, parallel experiments. Coverage observed within any specific trial can (and does) vary substantially from the aggregate tendency. There is very recent work which seeks to provide a stronger validity guarantee that can be expected to hold for some $(1-\delta)$ fraction of trials, which we hope to apply to conformal BayesOpt in future work (Bates et al., 2021).

Approximation error we have introduced some necessary approximations in this work, notably the discretization of continuous labels and the continuous relaxation of conformal prediction sets. While we have given empirical evidence that the error introduced by these approximations does not appear to be too severe, practitioners should be aware that some deviation from the expected coverage level may occur, as we discuss in Chapter 4.7. This limitation is analogous to the limitations of numerical linear algebra implemented with floating point arithmetic.

D.1.2 Broader Impacts:

Potential negative social impacts: black-box optimization algorithms are application-agnostic. The same algorithms that are being used to design new therapeutics could in theory be used to discover new toxins for bioterror or biowarfare. Similarly, the same algorithms used to design new materials for scientific discovery could be used to design new weapons or rocket fuels. Our work is not particularly vulnerable to misuse relative to the large body of existing work on black-box optimization algorithms.

Machine learning research: phenomena like model misspecification and covariate shift are often blamed on complexity in the external world, but they are also induced by our own behavior, such as choosing a convenient likelihood for a model (even when a more sophisticated option is available) or actively selecting new training data. We hope this work spurs more interest in understanding how to reliably interact with the models we have *today*, rather than only working on "better" models for tomorrow.

Experimental design: applications like materials science and drug discovery require the coordination of large, interdisciplinary teams of scientists and engineers. If machine learning systems are to play a central role in that coordination, they must be reliable, in the sense that the systems have stable behavior and consistently valid predictions. That kind of reliability requires more than faith in an ad hoc collection of model assumptions with limited experimental validation. This work is a step towards machine learning systems with interpretable certificates of reliability that can serve as the foundation on which to build teams which push the boundaries of experimental science. However, these types of design technologies do have the potential for misuse as described above.

Proofs and Derivations

D.2.1 Characterizing the conformal Bayes posterior

All conditional distributions are also conditioned on D, which we omit from the notation for the sake of clarity. Recall that the conformal Bayes posterior is written as

$$\begin{aligned}
p(f(\mathbf{x})|\mathbf{x}) &= \int_{y\in\mathcal{Y}} p(f(\mathbf{x})|\mathbf{x},y)p(y|\mathbf{x})dy, \\
&= \int_{y\in C_\alpha(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x},y)p(y|\mathbf{x})dy + \int_{y\in\mathcal{Y}-C_\alpha(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x},y)p(y|\mathbf{x})dy.
\end{aligned}$$

Now we define a new conformal Bayes posterior distribution as a mixture distribution over y,

$$p_\alpha(y = y'|\mathbf{x}) = (1-\alpha)q_1(y|\mathbf{x}) + \alpha q_2(y|\mathbf{x}) \tag{D.1}$$

$$Z_1 = \int_{y \in C_\alpha(\mathbf{x})} 1 dy, \qquad q_1(y|\mathbf{x}) = \begin{cases} 1/Z_1 & \text{if } y' \in C_\alpha(\mathbf{x}), \\ 0 & \text{else,} \end{cases}$$

$$Z_2 = \int_{y \in \mathcal{Y} - C_\alpha(\mathbf{x})} p(y|\mathbf{x}) dy, \qquad q_2(y|\mathbf{x}) = \begin{cases} 0 & \text{if } y' \in C_\alpha(\mathbf{x}), \\ p(y|\mathbf{x})/Z_2 & \text{else,} \end{cases}$$

where the normalizing constants Z_1, Z_2 ensure that $\int p_\alpha(y|\mathbf{x})dy = 1$ (assuming $C_\alpha(\mathbf{x})$ is bounded and non-empty, so Z_1 is non-zero and finite). Therefore the corresponding conformal Bayes posterior distribution over f is

$$\begin{aligned} p_\alpha(f(\mathbf{x})|\mathbf{x}) &= \int_{y \in \mathcal{Y}} p(f(\mathbf{x})|\mathbf{x}, y) p_\alpha(y|\mathbf{x}) dy \\ &= \frac{1-\alpha}{Z_1} \int_{y \in C_\alpha(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x}, y) dy + \frac{\alpha}{Z_2} \int_{y \in \mathcal{Y} - C_\alpha(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x}, y) p(y|\mathbf{x}) dy \end{aligned}$$

Finally we can rewrite both integrals over all $\mathcal{Y}$ by introducing a binary mask, Definition D.2.1.

$$
\begin{aligned}
p_\alpha(f(\mathbf{x})|\mathbf{x}) &:= \frac{1-\alpha}{Z_1}\int m_\alpha(\mathbf{x},y)p(f(\mathbf{x})|\mathbf{x},y)dy \\
&\quad + \frac{\alpha}{Z_2}\int (1-m_\alpha(\mathbf{x},y))p(f(\mathbf{x})|\mathbf{x},y)p(y|\mathbf{x})dy, \\
m_\alpha(\mathbf{x},y) &:= \begin{cases} 1 & \text{if } y \in C_\alpha(\mathbf{x}), \\ 0 & \text{else.} \end{cases}
\end{aligned}
$$

Proposition D.2.1. Let $n > 1$ and $p_\alpha(f|D)$ be defined according to Definition D.2.1. Then $p_\alpha(f|D)$ converges pointwise in $\mathbf{x}$ to $p(f(\mathbf{x})|\mathbf{x}, D)$ as $\alpha \to 1$,

$$
\lim_{\alpha\to 1} p_\alpha(f(\mathbf{x})|\mathbf{x}) = p(f|\mathbf{x}).
$$

Proof:

Let $\varepsilon > 0$, $n > 1$, and define $\alpha_k = 1 - 1/(k+1)$ for $k \in \mathbb{N}$ such that $k < n$.

$$
\begin{aligned}
|p_{\alpha_k}(f(\mathbf{x})|\mathbf{x}) - p(f(\mathbf{x})|\mathbf{x})| &= |\Delta_1 + \Delta_2|, \\
&\leq |\Delta_1| + |\Delta_2|,
\end{aligned}
$$

where

$$
\begin{aligned}
\Delta_1 &= \frac{1-\alpha_k}{Z_1}\int_{y\in C_{\alpha_k}(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x},y)dy - \int_{y\in C_{\alpha_k}(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x},y)p(y|\mathbf{x})dy, \\
\Delta_2 &= \frac{\alpha_k}{Z_2}\int_{y\in\mathcal{Y}-C_{\alpha_k}(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x},y)p(y|\mathbf{x})dy - \int_{y\in\mathcal{Y}-C_{\alpha_k}(\mathbf{x})} p(f(\mathbf{x})|\mathbf{x},y)p(y|\mathbf{x})dy.
\end{aligned}
$$

Recalling the definition of $C_\alpha(\mathbf{x})$ (Def. 4.4.1), we observe that $C_{\alpha_k}(\mathbf{x}) \supset C_{\alpha_{k+1}}(\mathbf{x}), \forall k \in \mathbb{N}$.[*] Furthermore we see that since the importance weights $\mathbf{w}$ must sum to 1 that $\lim_{k\to\infty} C_{\alpha_k}(\mathbf{x}) = \emptyset$.

Bounding $|\Delta_1|$:

$$\begin{aligned} |\Delta_1| &\le |\mathcal{O}(1-\alpha_k) - \mathcal{O}(1-\alpha_k)|, \\ \Rightarrow |\Delta_1| &\le c_1(1-\alpha_k). \end{aligned}$$

Bounding $|\Delta_2|$:

$$\begin{aligned} |\Delta_2| &\le |(\alpha_k - 1)\mathcal{O}(1)|, \\ \Rightarrow |\Delta_2| &\le c_2(1-\alpha_k). \end{aligned}$$

Choose $k \in \mathcal{N}$ large enough that $(c_1 + c_2)(1-\alpha_k) < \varepsilon$. ■

[*] $A \supset B$ indicates that A is a strict superset of B.

D.2.2 Monte Carlo integration of conformal acquisition functions

We want to integrate acquisition functions of the form

$$\begin{aligned}
a(\mathbf{x}) &= \int u(\mathbf{x}, f) p_\alpha(f|\mathbf{x}) df \\
&= \frac{1-\alpha}{Z_1} \int \int u(\mathbf{x}, f) m_\alpha(\mathbf{x}, y) p(f(\mathbf{x})|\mathbf{x}, y) dy df \\
&\quad + \frac{\alpha}{Z_2} \int \int u(\mathbf{x}, f)(1 - m_\alpha(\mathbf{x}, y)) p(f(\mathbf{x})|\mathbf{x}, y) p(y|\mathbf{x}) dy df,
\end{aligned}$$

Suppose we have sampled $Y_{\text{cand}} = \{y_j\}_{j=0}^{m-1}$, with $y_j \sim p(y|\mathbf{x})$, and $v_j \sim p(f|\mathbf{x}, y_j)$. Starting with the first term in the sum, we have

$$\frac{1-\alpha}{Z_1} \int \int u(\mathbf{x}, f) m_\alpha(\mathbf{x}, y) p(f(\mathbf{x})|\mathbf{x}, y) dy df \approx \frac{1-\alpha}{Z_1 |Y_{\text{cand}}|} \sum_j \frac{m_\alpha(\mathbf{x}, y_j)}{p(y_j|\mathbf{x})} u(\mathbf{x}, v_j).$$

We estimate the normalization constant Z_1 as follows:

$$\begin{aligned}
Z_1 = \int_{y \in C_\alpha(\mathbf{x})} 1 dy &= \int m_\alpha(\mathbf{x}, y) dy \\
&\approx \frac{1}{|Y_{\text{cand}}|} \sum_j \frac{m_\alpha(\mathbf{x}, y_j)}{p(y_j|\mathbf{x})}
\end{aligned}$$

By similar logic the second term in the sum is estimated as follows:

$$\begin{aligned}
&\frac{\alpha}{Z_2} \int \int u(\mathbf{x}, f)(1 - m_\alpha(\mathbf{x}, y)) p(f(\mathbf{x})|\mathbf{x}, y) p(y|\mathbf{x}) dy df \\
&\qquad \approx \frac{\alpha}{Z_2 |Y_{\text{cand}}|} \sum_j (1 - m_\alpha(\mathbf{x}, y_j)) u(\mathbf{x}, v_j),
\end{aligned}$$

where

$$Z_2 = \int_{y \in \mathcal{Y} - C_\alpha(\mathbf{x})} p(y|\mathbf{x}) dy = \int (1 - m_\alpha(\mathbf{x}, y)) p(y|\mathbf{x}) dy$$
$$\approx \frac{1}{|Y_{\text{cand}}|} \sum_j (1 - m_\alpha(\mathbf{x}, y_j))$$

Upon further inspection we see that $|Y_{\text{cand}}|$ drops out of the equations, and in effect we are simply computing weighted sums, where the weights have been normalized to sum to 1, i.e.

$$\begin{aligned} a(\mathbf{x}) \approx \hat{a}(\mathbf{x}) &= (1-\alpha)\beta_0^\top \mathbf{u} + \alpha \beta_1^\top \mathbf{u}, \\ \mathbf{u} &= [u(\mathbf{x}, v_0), \dots, u(\mathbf{x}, v_m)]^\top, \\ (\beta_0)_j &= \frac{m_\alpha(\mathbf{x}, y_j)}{p(y_j|\mathbf{x})} \left(\sum_k \frac{m_\alpha(\mathbf{x}, y_k)}{p(y_k|\mathbf{x})} \right)^{-1}, \\ (\beta_1)_j &= (1 - m_\alpha(\mathbf{x}, y_j)) \left(\sum_k (1 - m_\alpha(\mathbf{x}, y_k)) \right)^{-1}. \end{aligned} \tag{D.2}$$

D.2.3 Conformalized Single Objective Acquisitions

Conformal NEI: rather than taking $u(\mathbf{x}, D, f) = [f(\mathbf{x}) - \max_{y_i \in D} y_i]_+$ (which corresponds to EI), take $u(\mathbf{x}, D, f) = [f(\mathbf{x}) - \max_{\mathbf{x}_i' \in D} f(\mathbf{x}_i)]_+$. Note that u is now a function of the joint collection of function evaluations $(f(\mathbf{x}'), f(\mathbf{x}_1), \dots, f(\mathbf{x}_{n-1}))$.

Conformal UCB: the reparameterized form of UCB was originally derived in Wilson et al. (2017) as follows:

$$\mathrm{UCB}(\mathbf{x}) = \int u(\mathbf{x}, f)\mathcal{N}\left(f \middle| \mu, \frac{\beta\pi}{2}\Sigma\right) df,$$

where $\beta > 0$ is a hyperparameter, μ, Σ are the mean and covariance of $p(f|D)$, and $u(\mathbf{x}, f) = \mu + \frac{\beta\pi}{2}|\mu - f|$. Because UCB is optimistic, the conformalization procedure is a little different than the previous acquisition functions. When marginalizing out the outcomes y to obtain the conformal Bayes posterior, we integrate over the restricted outcome space $\mathcal{Y}_\mu = \{y \in \mathcal{Y} | y \geq \mu\}$. Hence we derive conformal UCB as

$$\mathrm{CUCB}_\alpha(\mathbf{x}) = \int \int_{y \in \mathcal{Y}_\mu} u(\mathbf{x}, f)\mathcal{N}\left(f \middle| \mu(y), \frac{\beta\pi}{2}\Sigma(y)\right) p_\alpha(y|\mathbf{x}, D) dy df,$$

where $\mu(y), \Sigma(y)$ are the predictive mean and covariance of $p(f|\mathbf{x}, y, D)$.

D.2.4 Conformalized Multi-Objective Acquisition Functions

Following Daulton et al. (2020b), we have that EHVI can be written as

$$\mathrm{EHVI}(\mathbf{x}) = \int \mathrm{HVI}(f(\mathbf{x}))p(f|D, \mathbf{y})df = \sum_{k=1}^{K} \int \mathrm{HVI}_k(f(\mathbf{x}))p(f|D, \mathbf{y})df,$$

where $\mathrm{HVI}_k(f(\mathbf{x}))$ is the kth box hypervolume improvement. This is our utility function and is defined as

$$\mathrm{HVI}_k(f(\mathbf{x})) = \prod_{m=1}^{M} [\min(u_k^{(m)}, f(\mathbf{x})) - l_k^{(m)}]_+,$$

where $u_k^{(m)}$ ($l_k^{(m)}$) is the mth upper (lower) vertex corresponding to the kth non-dominated hyper-rectangle in function space.

Instead of $f(\mathbf{x})$, we use $y(\mathbf{x})$ to compute NEHVI instead, following Daulton et al. (2021a).

Our derivations hold for so-called composite acquisitions as well, so that we could also easily extend to qParEGO and qNParEGO variants for multi-objective optimization (Daulton et al., 2020b; 2022).

D.2.5 Conformalizing Batch Acquisitions

In general batch acquisitions have the form

$$a(\mathbf{x}_0, \dots, \mathbf{x}_{q-1}) = \int \max_{i<q} u(\mathbf{x}_i, f) p(f|D) df. \tag{D.3}$$

Note that $f(\mathbf{x}_0), \dots, f(\mathbf{x}_{q-1})$ are sampled jointly when estimating Eq. (D.3) with Monte Carlo. Increasing the query batch size to q increases the dimensionality of the outcome to $q \times p$, where p is the number of objectives. Our importance-sampling MC integration procedure introduced in Section 4.6.2 scales gracefully with higher outcome dimensionality, we simply sample the elements of Y_{cand} from $p(y(\mathbf{x}_0), \dots, y(\mathbf{x}_{q-1})|\mathbf{x}_{0:q-1}, D)$.

The bigger challenge arises in computing the conformal masks for batched query outcomes. In our current implementation we compute the conformal scores (the Bayes posterior log-likelihood) pointwise for each query batch element, with corresponding pointwise conformal prediction masks. We apply the pointwise masks before computing $\max_{i<q} u$ across query batch elements.

The alternative would be to compute a joint conformal score across all query batch elements (similarly computing joint scores for each of the previous query batches in the training data). Note that this second approach essentially reduces to replacing each datum $(\mathbf{x}_i, y_i)$ in Eq. (4.1) with

$$(X_i, \mathbf{y}_i) = ([\mathbf{x}_0, \dots, \mathbf{x}_{q-1}]^\top, [y_0, \dots, y_q]^\top).$$

We leave the implementation of this second approach for future work.

Implementation Details

D.3.1 Bringing everything together

In Algorithm 7 we summarize the entire conformal BayesOpt inner loop used to select new queries.

D.3.2 Stable Predictions on the Training Set

We found that computing the negative log likelihood (and its gradients) on our set of training data to be numerically unstable and so used stochastic diagonal estimation to estimate the posterior variances. Plugging in K_{XX} into that posterior mean and variance, we get that the posterior mean is $K(K + \sigma^2)^{-1}y$ and the posterior covariance is $\Sigma = \sigma^2 I + K - K(K + \sigma^2 I)^{-1}K$. Unfortunately, the second term ends up being unstable as it requires solving (and then subtracting) a (batched) system of size $n \times n$. To see the reason for instability, note that as $\sigma^2 I \to 0$ then the entire covariance matrix tends to zero.

Algorithm 7: Pseudocode for the conformal BayesOpt inner loop

```
Input: train data $D = \{(\mathbf{x}_i, y_i)\}_{i=0}^{n-1}$, initial solution $\mathbf{x}_n$, score function $s$,
  miscoverage tolerance $\alpha$, sigmoid temperature $\tau_\sigma$, SGLD learning rate $\eta_{\mathbf{x}}$,
  # SGLD steps $t_{\max}$, SGLD temperature $\tau_{\text{sgld}}$, classifier learning rate $\eta_\theta$,
  EMA parameter $\gamma$.
Initialize classifier $q_\theta$, set weight average $\bar{\theta} = \mathbf{0}$.
Initialize classifier dataset $D' = \{(\mathbf{x}_i, 0)\}_{i=0}^{n-1}$
for $t = 0, \dots, t_{\max} - 1$ do
    Estimate $\hat{r}_t(\mathbf{x}_i), \forall i \in \{0, \dots, n\}$ with $q_{\bar{\theta}}$.                (Eq. 4.8)
     $(\mathbf{w}_t)_i = \hat{r}_t(\mathbf{x}_i) / \sum_k \hat{r}_t(\mathbf{x}_k), \forall i \in \{0, \dots, n\}$. Draw $Y_t = \{y_j\}_{j=0}^{m-1}$ s.t.
     $y_j \sim \hat{p}(y | \mathbf{x}'_t, D)$.
    $\mathbf{m} = \text{outcome\_mask}(D, \mathbf{x}_n, \mathbf{w}_t, Y_t, s, \alpha, \tau_\sigma)$.          (Algorithm 4)
    Estimate acquisition value $a(\mathbf{x}_n)$.                    (Eq. D.2) Update
     $\mathbf{x}_n \leftarrow \text{sgld\_step}(\mathbf{x}_n, a(\mathbf{x}_n), \eta_{\mathbf{x}}, \tau_{\text{sgld}})$. Update $D' \leftarrow D' \cup \{(\mathbf{x}_n, 1)\}$.
     Update $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \ell(\theta, D')$ Update $\bar{\theta} \leftarrow (1 - \gamma)\bar{\theta} + \gamma\theta$.
return $\mathbf{x}_n$
```

We originally tried backpropagating through an eigendecomposition; however, this produced ill-defined gradients, see the explanation in Ionescu et al. (2015), before instead computing a stochastic diagonal estimate. For the stochastic diagonal estimate, we used the identities

$$\Sigma = \sigma^2 I + \sigma^2 K (K + \sigma^2 I)^{-1} \tag{D.4}$$

$$\text{diag}(\Sigma) \approx \sigma^2 \left(1 + \frac{\sum_{i=1}^{J} z_i \odot K(K + \sigma^2 I)^{-1} z}{\sum_{i=1}^{J} z_i \odot z_i} \right), \tag{D.5}$$

where z_i has i.i.d Bernoulli entries and $\odot$ is the Hadamard product. This estimator comes from Bekas et al. (2007) and is in spirit quite similar to Hutchinson's trace estimator for the log determinant. We used $J = 10$ probe vectors.

Philosophical Discussion

This section contains an extended discussion on the ideas presented in Section 4.3.

Highlighting model assumptions: f^* is an element of some true hypothesis space $\mathcal{F}^*$, which we may or may not be able to fully realize in practice (due to computational constraints, finite numerical precision, etc.). Instead we *assume* f^* either lies in, or is "close" to some f in a realizable model class $\mathcal{F}$. We also usually do not observe f^* directly, but instead receive a dataset of examples $D := (X_{\text{seen}}, Y_{\text{seen}}) = \{(\mathbf{x}_i, y_i)\}_{i=0}^{n-1}$, where $\mathbf{x}_i \in \mathcal{X}$ are drawn from a true marginal distribution $p(\mathbf{x})$, and noisy labels $\mathbf{y}_i \in \mathcal{Y}$ are drawn from a true conditional distribution $p^*(\mathbf{y}|\mathbf{x})$ defined by the composition of f^* and some noise process. We rarely know the true noise process, instead we often *assume* the noise process has some convenient form, such as $\mathbf{y}_i = f(\mathbf{x}_i) + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_p)$. Our model of the noise process determines the model likelihood, $p(D|f) = p(\mathbf{y}_0, \ldots, \mathbf{y}_{n-1}|\mathbf{x}_0, \ldots, \mathbf{x}_{n-1}, f)$.

Credibility and coverage are often confused. Neither $p(f|D)$, nor $p(\mathbf{y}|\mathbf{x}, D)$, nor credible sets derived from either posteriors should be interpreted as approximating some long-term, empirical, observed frequency. For $p(f|D)$ and its corresponding credible sets Y_{cred}, it would scarcely be sensible to talk of the observed frequency of events that are never actually observed. However the distinction is often missed when considering $p(\mathbf{y}|\mathbf{x}, D)$ and its corresponding credible sets Y_{cred}. Indeed, there is an abundance of literature discussing the "calibration" of $p(\mathbf{y}|\mathbf{x}, D)$ in the context of Bayesian deep learning (Guo et al., 2017; Ovadia et al., 2019), where calibration means the empirical frequency with which labels in a fixed set of test examples fall within their corresponding ϕ-credible sets (conditioned on a fixed set of train examples) for different choices of ϕ, epitomized by the wide usage of the ECE metric introduced by Naeini et al. (2015). Notions of calibration like ECE resemble the frequentist concept of *coverage*, which we discuss in more detail in the next section, but are not exactly equivalent. Furthermore, even if ECE was measuring frequentist coverage, there is the more troubling fact that in general *Bayesian credibility does not directly correspond to frequentist coverage.*[*] A Bayesian ϕ-credible set may contain the true labels roughly $(\phi \times 100)\%$ of the time, or it may not (Wasserman, 2008).

Informally this conceptual difference is due to the fact that credible sets are the realization of the Bayesian ideal of perfect *internal* coherence (assuming exact inference) when our model assumptions perfectly represent our true beliefs, with well-known rationales such as the Dutch book argument. By contrast, frequentist coverage measures *external* correspondence, measuring the degree to which our predictions agree with real measurements, regardless of the modeling assumptions we made to arrive at our predictions. The ubiquity of well-worn maxims like "all models are wrong, but some are useful" should hint to us that internal coherence and external correspondence are often in

[*]Under very specific assumptions such as those found in Monard et al. (2021) some Bayesian credible sets can be shown to be valid, optimal frequentist predictions sets, but such cases are the exception, not the rule.

conflict. This tension was recognized by Dawid (1982) who argued that *any* post-hoc calibration procedure is incoherent in the internal, Bayesian sense, though such procedures may indeed have practical value.

So why do Bayesian inference? The real utility of Bayesian inference lies not in the coverage (or lack thereof) of Bayesian credible intervals, but in the way Bayesian posteriors provide a straightforward means to compute *useful decision rules*. Briefly put, suppose we have a utility function $u(\delta(D), f)$ which measures the utility of selecting a query point $\mathbf{x}$ via decision rule δ based on information D if $f^* = f$. It is easier to find a Bayes-optimal decision rule δ^* that maximizes the posterior expected utility $E_{p(f|D)}[u(\delta(D), f)]$, than it is to compute a frequentist u-admissible decision rule satisfying $\mathbb{E}_{p(D)}[u(\delta(D), f)] \geq \mathbb{E}_{p(D)}[u(\delta'(D), f)]$, $\forall f \in \mathcal{F}$ and all possible alternative decision rules δ'.

Frequentist decision rule criteria like u-admissibility are also subject to modeling assumptions (particularly the choice of $\mathcal{F}$), and are arguably overly-conservative in many situations, since they ignore the fact that after seeing D there are certain choices of f that can be all but eliminated from consideration, even after accounting for uncertainty in the model assumptions. For a thorough introduction to a decision-theoretic motivation of Bayesian inference, see Berger (2013).

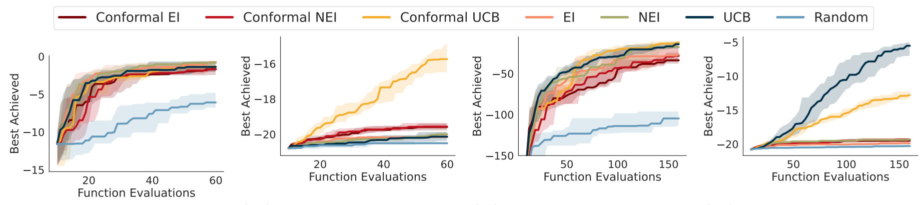
Additional Experimental Results

D.5.1 Benchmark Problems

Single Queries We display optimization performance for $q = 1$ and $q = 3$ in Figure D.1 across conformal expected improvement (EI), noisy expected improvement (NEI) and upper confidence bound (UCB) alongside their non-conformal counterparts. Overall, we see that the conformalized aquisitions tend to slightly outperform their non-conformalized counterparts, especially for $q = 1$ and on the higher dimensional problems. However, the picture is somewhat noisy overall as these problems are relatively straightforward.

Next, in Figure D.2, we evaluate the coverage of the conformalized set and the Bayesian posterior as we increaase the dimensionality of the problem. These are the same models as the $q = 3$ row in Figure D.1. Here, we plot the median and its 95% confidence interval as shading, finding that the conformal sets are better calibrated in a frequentist sense than the equivalent coverage level of the Bayesian posterior. The gaps in coverage levels improves as we increase the dimensionality from 5 to 20 as the GP models become increasingly more misspecified and their fits degrade.

Batch Queries Next in Figure D.1c and D.1d, querying $q = 3$ points at a time. Coverage plots are shown in Appendix D.5. As discussed in Appendix D.2.5, we compute the conformal scores and masks pointwise across query batch elements, which introduces some additional error that may hinder performance. The conformal acquisition variants still generally perform well rela-



(a) Levy-5, $q = 1$. **(b)** Ackley-20, $q = 1$. **(c)** Levy-20, $q = 3$. **(d)** Ackley-20, $q = 3$.

Figure D.1: BayesOpt best objective value found with conformal and standard acquisition functions on single-objective tasks Levy-d and Ackley-d (reporting median and its 95% conf. interval, estimated from 25 trials). qEI, qNEI, conformal qEI, and conformal qNEI all perform similarly, conformal qUCB is best everywhere except Ackley-20, where it comes second after qUCB.

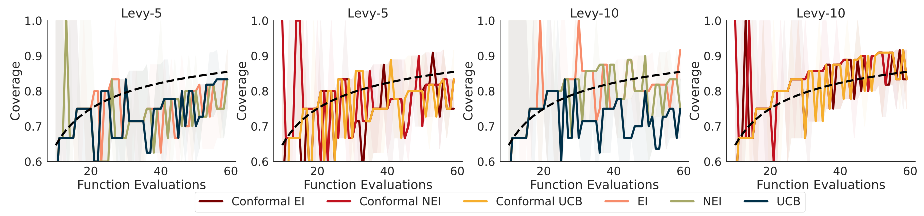
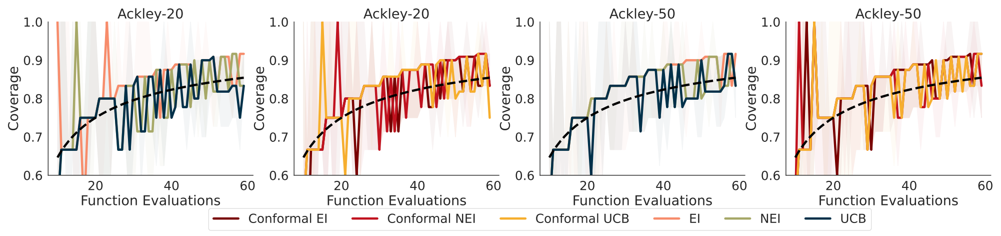


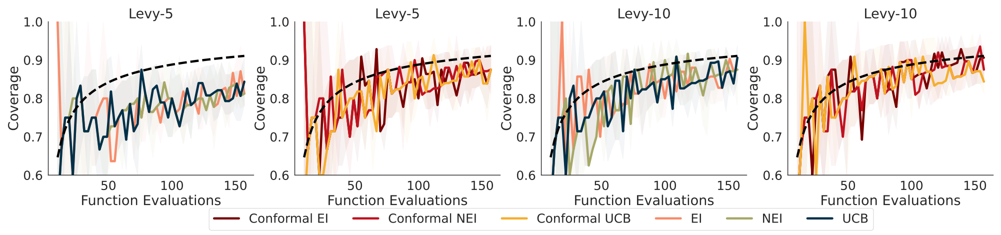
Figure D.2: BayesOpt empirical coverage of conformal and credible prediction sets evaluated on holdout data from single-objective task Levy-d (reporting median and its 95% conf. interval, estimated from 25 trials). The conformal coverage curves track the target $1 - \alpha$ (black dashed line) well, significantly better than the credible curves, which tend to be overconfident. We expect these intervals to have a percentage coverage across repeated trials (shown in black), but find that the credible intervals tend to be overconfident as the dimensionality increases, an indication of misspecification and underfitting. Median w/ 95% confidence interval is shown.

tive to the standard variants, except for Ackley-20 where qUCB beats all competitors.

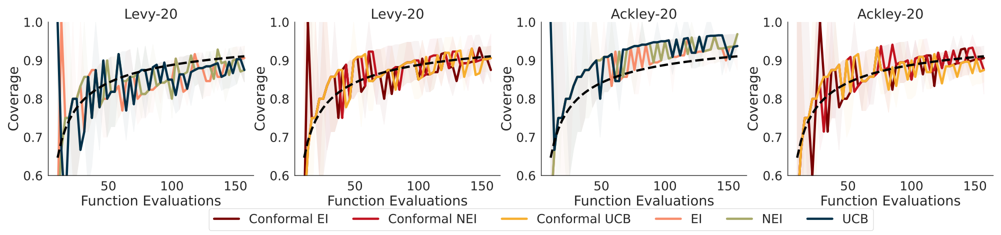
In Figure D.3, we display the single objective coverages for both $q = 1$ and $q = 3$ on the Levy and Ackley test functions. Again, we see that the conformal posterior distributions have better empirical coverage than the corresponding Bayesian posteriors throughout the optimization process. Specifically, we find that on the higher dimensional problems, the amount of variation is reduced as the models become increasingly poorly specified. We also see that the Bayesian credible sets often tend to be slightly over-confident, except on Ackley-20, and indication of over-fitting.



(a) Ackley-20D and Ackley-50D, $q = 1$.



(b) Levy-5D and Levy-10D, $q = 3$.



(c) Levy-5D and Ackley-20D, $q = 3$.

Figure D.3: Coverage of posterior intervals (either conformal for conformalized acquisitions or posterior for standard) across Levy problems of varying dimensionalities for 25 trials (median and its 95% confidence interval are shown). We expect these intervals to have a $1 - \alpha$ percentage coverage across repeated trials (shown in black), but find that the credible intervals tend to be overconfident as the dimensionality increases, an indication of misspecification and underfitting.

In Figure D.4, we display the coverages of the conformal acquisitions on the BraninCurrin test problem, finding surprisingly that the conformal prediction sets tend to be slightly under-confident on this problem, perhaps due to its

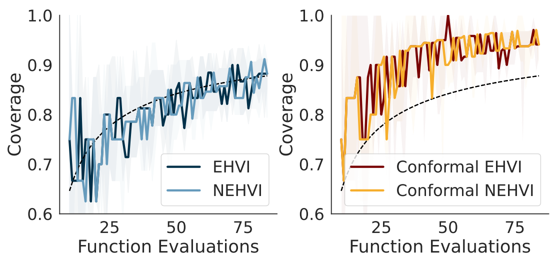


Figure D.4: Coverage for both objectives on BraninCurrin. Here, we see that the conformal regions tend to surprisingly be quite under-confident compared to the standard credible sets

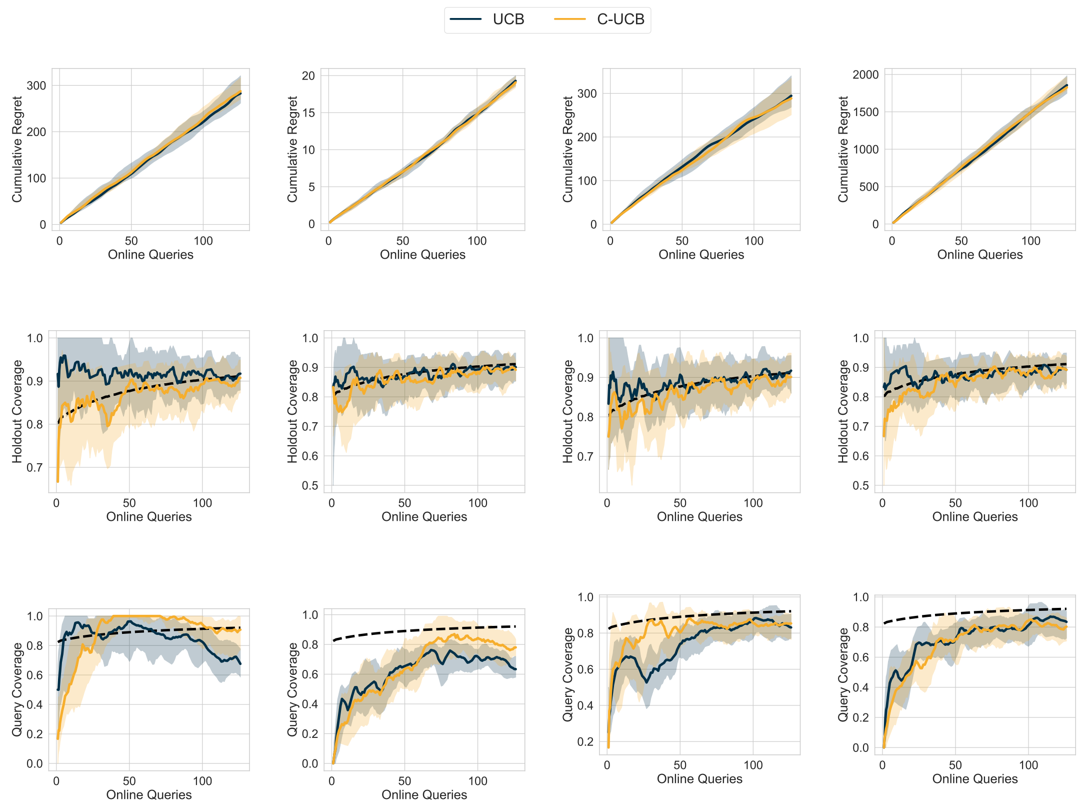
simplicity with two objectives.

D.5.2 Tabular Bandits with Real-World Drug and Antibody Data

In Figure D.5, we consider a ranking task and querying models online using either standard upper confidence bound (UCB) or conformal UCB (C-UCB). The first three tasks are a penalized logP task, a single objective QED task, and the DRD3 binding affinity task, are all derived from encodings on small molecule descriptors, please see Stanton et al. (2022) for further details about the objectives. Penalized logP and QED were originally proposed in Gómez-Bombarelli et al. (2018), while we used the SELFIES encodings from Krenn et al. (2020) to produce a continuous space over the ZINC dataset.

Following Stanton et al. (2022) in (c), we use the DRD3 docking score oracle from Huang et al. (2021), see https://tdcommons.ai/benchmark/docking_group/overview for a further description.

The data for task (d) was derived from the OAS dataset (Hornung et al., 2014), with additional descriptors annotated with the NARD protein annotation tool.



(a) penalized logP

(b) QED

(c) DRD3 binding affinity

(d) OAS Protein Stability

Figure D.5: Result ranking tabular molecular datasets for drug-related properties such as solubility (a), empirical drug-likeness (b), dopamine receptor binding affinity (c) (all from Stanton et al. (2022)) and antibody stability (d) (derived from OAS (Hornung et al., 2014). Across datasets, our query coverage is noticeably better than UCB (bottom two rows), while we also achieve some gains in sample efficiency (top row).

Our query coverage is noticeably better than UCB, and we also see small gains to sample efficiency. Again, we consider both query and heldout set coverage, comparing to the given choice of α across 4 random seeds.

Modelling Hyperparameters:

For all GP models in this paper, we used the default single task GP (`SingleTaskGP`) model from BoTorch, which uses a scaled Matern-5/2 kernel with automatic relevance determination and a Gamma$(3, 6)$ prior over the lengthscales and a Gamma$(2, 0.15)$ prior over the outputscales. We used constant means. For the likelihood, we used a softplus transformation to optimize the raw noise, constraining the noise to be between 5×10^{-4} and 0.5.

To fit the models, we used BoTorch's default fitting utility, `fit_gpytorch_model`, which uses L-BFGS-B to fit the hyperparameters.

Conformal Prediction Hyperparameters:

We used a minimum α level of 0.05 after noting that with fewer than α^{-1} points, it was impossible to exclude grid points from the prediction set. We used $\tau = 0.01$, a target grid size of 64, a maximu

Optimization Hyperparameters:

To optimize the acquisitions, we used 256 MC samples for the single objective tasks and 64 tasks for the multi-objective problems, 100 steps of SGLD with 25 burn-in, a SGLD temperature of 10^{-3}, and a SGLD learning rate of 10^{-3}.

For non-conformal acquisitions, we used the same amount of samples and then optimized with L-BFGS-B with 5 random restarts, 128 raw samples, and a maximum of 200 steps.

We initialized with 10 Sobol points (or points drawn from a random orthant of the data on Ackley in Figure D.1b). All functions had data drawn with a noise standard error of 0.1 on a normalized scale; the normalization was constructed from 1000 Sobol points on the function.

Compute: Our experiments were conducted on a range of NVIDIA GPUs, including RTX 2080 Tis, Titan RTXs, V100s, and A100s on internal clusters. All experiments used a single GPU at a time. It would require approximately 250 GPU hours to reproduce the experiments in this paper by our estimate,

$$\begin{aligned}
&1 \text{ GPU hr/seed} \\
&\times 25 \text{ seeds per variant} \\
&\times 1 \text{ variant per experiment} \\
&\times 10 \text{ experiments} \\
&= 250 \text{ hrs.}
\end{aligned}$$

Other experimental runs, e.g. debugging, probably consumed an order of magnitude more GPU hours.

Software packages:

- Python 3, PSF License Agreement,
- Matplotlib, Matplotlib License Agreement,
- Seaborn, BSD License,
- NumPy, BSD License,

- PyTorch, BSD License.
- GPyTorch, MIT License (Gardner et al., 2018a).
- BoTorch, MIT License (Balandat et al., 2020a).
- TorchSort[*], Apache 2.0 License.

All functions used in this paper are ultimately synthetic with implementations coming from BoTorch (Balandat et al., 2020a). The Penicillin function comes from Liang & Lai (2021).

[*]https://github.com/teddykoker/torchsort

Bibliography

Alvarez, M. A., Rosasco, L., & Lawrence, N. D. (2011). Kernels for vector-valued functions: A review. *arXiv preprint arXiv:1106.6251*.

Andrade-Pacheco, R., Rerolle, F., Lemoine, J., Hernandez, L., Meïté, A., Juziwelo, L., Bibaut, A. F., van der Laan, M. J., Arnold, B. F., & Sturrock, H. J. (2020). Finding hotspots: development of an adaptive spatial sampling approach. *Scientific reports*, 10(1), 1–12.

Angelopoulos, A. N. & Bates, S. (2021). A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*.

Angermueller, C., Belanger, D., Gane, A., Mariet, Z., Dohan, D., Murphy, K., Colwell, L., & Sculley, D. (2020a). Population-based black-box optimization for biological sequence design. In *International Conference on Machine Learning* (pp. 324–334).: PMLR.

Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., & Colwell, L. (2020b). Model-based reinforcement learning for biological sequence design. In *International Conference on Learning Representations*.

Assael, J.-A. M., Wang, Z., Shahriari, B., & de Freitas, N. (2014). Heteroscedastic treed bayesian optimisation. *arXiv preprint arXiv:1410.7172*.

Bach, F. R. & Jordan, M. I. (2002). Kernel independent component analysis. *Journal of machine learning research*, 3(Jul), 1–48.

Bachoc, F. (2013). Cross validation and maximum likelihood estimations of hyper-parameters of gaussian processes with model misspecification. *Computational Statistics & Data Analysis*, 66, 55–69.

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R., Wang, J., Cong, Q., Kinch, L. N., Schaeffer, R. D., et al. (2021). Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557), 871–876.

Bai, Y., Mei, S., Wang, H., Zhou, Y., & Xiong, C. (2022). Efficient and differentiable conformal prediction with general function classes. *arXiv preprint arXiv:2202.11091*.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020a). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 33.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020b). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*.

Bates, S., Candès, E., Lei, L., Romano, Y., & Sesia, M. (2021). Testing for outliers with conformal p-values. *arXiv preprint arXiv:2104.08279*.

Bauer, M., van der Wilk, M., & Rasmussen, C. E. (2016). Understanding probabilistic sparse gaussian process approximations. In *Advances in neural information processing systems*, volume 29 (pp. 1533–1541).

Beck, D. & Cohn, T. (2017). Learning kernels over strings using gaussian processes. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (pp. 67–73).

Beckers, T., Umlauft, J., & Hirche, S. (2018). Mean square prediction error of misspecified gaussian process models. In *2018 IEEE Conference on Decision and Control (CDC)* (pp. 1162–1167).: IEEE.

Bekas, C., Kokiopoulou, E., & Saad, Y. (2007). An estimator for the diagonal of a matrix. *Applied numerical mathematics*, 57(11-12), 1214–1229.

Berger, J. O. (2013). *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media.

Bertsekas, D. P. (2020). *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena Scientific.

Beume, N., Fonseca, C. M., Lopez-Ibanez, M., Paquete, L., & Vahrenhold, J. (2009). On the complexity of computing the hypervolume indicator. *IEEE Transactions on Evolutionary Computation*, 13(5), 1075–1082.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., & Hopkins, A. L. (2012). Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2), 90–98.

Bijl, H., Schön, T. B., van Wingerden, J.-W., & Verhaegen, M. (2016). Online sparse gaussian process training with input noise. *arXiv preprint arXiv 1601.08068*.

Bishop, C. M. & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M., & Church, G. M. (2021). Low-n protein engineering with data-efficient deep learning. *Nature Methods*, 18(4), 389–396.

Blank, J. & Deb, K. (2020). Pymoo: Multi-objective optimization in python. *IEEE Access*, 8, 89497–89509.

Boedecker, J., Springenberg, J. T., Wülfing, J., & Riedmiller, M. (2014). Approximate real-time optimal control based on sparse gaussian process models. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)* (pp. 1–8).: IEEE.

Bogunovic, I. & Krause, A. (2021). Misspecified gaussian process bandit optimization. *Advances in Neural Information Processing Systems*, 34.

Bonilla, E. V., Chai, K., & Williams, C. (2007). Multi-task gaussian process prediction. *Advances in neural information processing systems*, 20.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., & Jordan, M. I. (2013). Streaming variational bayes. In *Advances in Neural Information Processing Systems*, volume 26.

Bui, T. D., Nguyen, C. V., & Turner, R. E. (2017a). Streaming sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems 31*, volume 31 (pp. 3301–3309). Long Beach, California, USA: Curran Associates Inc.

Bui, T. D., Yan, J., & Turner, R. E. (2017b). A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *The Journal of Machine Learning Research*, 18(1), 3649–3720.

Burt, D., Rasmussen, C. E., & Van Der Wilk, M. (2019). Rates of convergence for sparse variational Gaussian process regression. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 (pp. 862–871).: PMLR.

Cakmak, S., Astudillo Marban, R., Frazier, P., & Zhou, E. (2020). Bayesian optimization of risk measures. *Advances in Neural Information Processing Systems*, 33, 20130–20141.

Canu, S. & Smola, A. (2006). Kernel methods and the exponential family. *Neurocomputing*, 69(7), 714–720. New Issues in Neurocomputing: 13th European Symposium on Artificial Neural Networks.

Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6, 100134.

Chang, C.-C. & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chen, Y., Huang, A., Wang, Z., Antonoglou, I., Schrittwieser, J., Silver, D., & de Freitas, N. (2018). Bayesian optimization in alphago. *arXiv preprint arXiv:1812.06855*.

Cheng, C.-A. & Boots, B. (2016). Incremental variational sparse Gaussian process regression. In *Advances in Neural Information Processing Systems 30*, volume 30 (pp. 4410–4418). Barcelona, Spain: Curran Associates Inc.

Chithrananda, S., Grand, G., & Ramsundar, B. (2020). Chemberta: Large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*.

Chowdhary, G., Kingravi, H. A., How, J. P., & Vela, P. A. (2014). Bayesian nonparametric adaptive control using gaussian processes. *IEEE transactions on neural networks and learning systems*, 26(3), 537–550.

Chu, W. & Ghahramani, Z. (2005). Preference learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine learning* (pp. 137–144).

Chudakov, D. M., Matz, M. V., Lukyanov, S., & Lukyanov, K. A. (2010). Fluorescent proteins and their applications in imaging living cells and tissues. *Physiological Reviews*, 90(3), 1103–1163. PMID: 20664080.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., et al. (2009). Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11), 1422–1423.

Coley, C. W., Eyke, N. S., & Jensen, K. F. (2020). Autonomous discovery in the chemical sciences part ii: outlook. *Angewandte Chemie International Edition*, 59(52), 23414–23436.

Csató, L. & Opper, M. (2002). Sparse on-line gaussian processes. *Neural computation*, 14(3), 641–668.

Dance, A. (2021). The hunt for red fluorescent proteins. *Nature*, 596, 152–153. (Online) https://doi.org/10.1038/d41586-021-02093-6.

Daulton, S., Balandat, M., & Bakshy, E. (2020a). Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. *Advances in Neural Information Processing Systems*, 33.

Daulton, S., Balandat, M., & Bakshy, E. (2020b). Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *arXiv preprint arXiv:2006.05078*, 33, 9851–9864.

Daulton, S., Balandat, M., & Bakshy, E. (2021a). Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *arXiv preprint arXiv:2105.08195*, 34.

Daulton, S., Cakmak, S., Balandat, M., Osborne, M. A., Zhou, E., & Bakshy, E. (2022). Robust multi-objective bayesian optimization under input noise. *arXiv preprint arXiv:2202.07549*.

Daulton, S., Eriksson, D., Balandat, M., & Bakshy, E. (2021b). Multi-objective bayesian optimization over high-dimensional search spaces. *arXiv preprint arXiv:2109.10964*.

Dawid, A. P. (1982). The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379), 605–610.

De Finetti, B. (1975). *Theory of probability: A critical introductory treatment*, volume 6. John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2), 182–197.

Deisenroth, M. & Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on Machine Learning* (pp. 465–472).

Delbridge, I., Bindel, D., & Wilson, A. G. (2020). Randomly projected additive Gaussian processes for regression. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research* (pp. 2453–2463). Virtual: PMLR.

Deshwal, A. & Doppa, J. (2021). Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. *Advances in Neural Information Processing Systems*, 34.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dua, D. & Graff, C. (2017). UCI machine learning repository.

Duris, J., Kennedy, D., Hanuka, A., Shtalenkova, J., Edelen, A., Baxevanis, P., Egger, A., Cope, T., McIntire, M., Ermon, S., et al. (2020). Bayesian optimization of a free-electron laser. *Physical Review Letters*, 124(12), 124801.

Emmerich, M. (2005). Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels. *dissertation, Universität Dortmund*.

Emmerich, M. T., Deutz, A. H., & Klinkenberg, J. W. (2011). Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *2011 IEEE Congress of Evolutionary Computation (CEC)* (pp. 2147–2154).: IEEE.

Eriksson, D. & Jankowiak, M. (2021). High-dimensional bayesian optimization with sparse axis-aligned subspaces. *arXiv preprint arXiv:2103.00349*.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., & Poloczek, M. (2019). Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32 (pp. 5496–5507).: Curran Associates, Inc.

Ertl, P. & Schuffenhauer, A. (2009). Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1), 1–11.

Evans, T. & Nair, P. (2018). Scalable gaussian processes with grid-structured eigenfunctions (gp-grief). In *International Conference on Machine Learning* (pp. 1417–1426).

Fannjiang, C., Bates, S., Angelopoulos, A., Listgarten, J., & Jordan, M. I. (2022). Conformal prediction for the design problem. *arXiv preprint arXiv:2202.03613*.

Fiedler, C., Scherer, C. W., & Trimpe, S. (2021). Practical and rigorous uncertainty bounds for gaussian process regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35 (pp. 7439–7447).

Fine, S. & Scheinberg, K. (2001). Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec), 243–264.

Fong, E. & Holmes, C. C. (2021). Conformal bayesian computation. *Advances in Neural Information Processing Systems*, 34.

Frazier, P. I. (2018). A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Fröhlich, L., Klenske, E., Vinogradska, J., Daniel, C., & Zeilinger, M. (2020). Noisy-input entropy search for efficient robust bayesian optimization. In

S. Chiappa & R. Calandra (Eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research* (pp. 2262–2272).: PMLR.

Fu, T., Xiao, C., Li, X., Glass, L. M., & Sun, J. (2020). Mimosa: Multi-constraint molecule sampling for molecule optimization. *arXiv preprint arXiv:2010.02318*.

Gao, W., Mercado, R., & Coley, C. W. (2021). Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design. *arXiv preprint arXiv:2110.06389*.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., & Wilson, A. G. (2018a). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, volume 31 (pp. 7576–7586).

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., & Wilson, A. G. (2018b). GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Advances in Neural Information Processing Systems*, volume 31.

Gill, P. E., Golub, G. H., Murray, W., & Saunders, M. A. (1974). Methods for modifying matrix factorizations. *Mathematics of computation*, 28(126), 505–535.

Ginsbourger, D., Le Riche, R., & Carraro, L. (2010). Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems* (pp. 131–162). Springer.

Girard, A., Rasmussen, C. E., Candela, J. Q., & Murray-Smith, R. (2002). Gaussian process priors with uncertain inputs-application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems*, volume 15.

Gligorijevic, V., Berenberg, D., Ra, S., Watkins, A., Kelow, S., Cho, K., & Bonneau, R. (2021). Function-guided protein design by deep manifold sampling. *bioRxiv*.

Golub, G. H. & Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU press.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., & Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2), 268–276.

Goovaerts, P. et al. (1997). *Geostatistics for natural resources evaluation.* Oxford University Press on Demand.

Gramacy, R. B. (2005). *Bayesian treed Gaussian process models.* University of California, Santa Cruz.

Griffiths, R.-R., Aldrick, A. A., Garcia-Ortegon, M., Lalchand, V. R., & Lee, A. A. (2019). Achieving robustness to aleatoric uncertainty with heteroscedastic bayesian optimisation. *arXiv preprint arXiv:1910.07779.*

Groot, P., Lucas, P., & Bosch, P. (2011). Multiple-step time series forecasting with sparse gaussian processes. In *Causmaecker, P. De (ed.), Proceedings of the 23rd Benelux Conference on Artificial Intelligence* (pp. 1–8).: [Sl: sn].

Grosnit, A., Tutunov, R., Maraval, A. M., Griffiths, R.-R., Cowen-Rivers, A. I., Yang, L., Zhu, L., Lyu, W., Chen, Z., Wang, J., et al. (2021). High-dimensional bayesian optimisation with variational autoencoders and deep metric learning. *arXiv preprint arXiv:2106.03609.*

Grünwald, P. & Van Ommen, T. (2017). Inconsistency of bayesian inference for misspecified linear models, and a proposal for repairing it. *Bayesian Analysis*, 12(4), 1069–1103.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International Conference on Machine Learning* (pp. 1321–1330).: PMLR.

Gupta, A., Anpalagan, A., Guan, L., & Khwaja, A. S. (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10, 100057.

Hagan, P. S., Kumar, D., Lesniewski, A. S., & Woodward, D. E. (2002). Managing smile risk. *The Best of Wilmott*, 1, 249–296.

Hennig, P. & Schuler, C. J. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(6).

Hensman, J., Fusi, N., & Lawrence, N. D. (2013a). Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*.

Hensman, J., Fusi, N., & Lawrence, N. D. (2013b). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13 (pp. 282–290). Arlington, Virginia, USA: AUAI Press.

Hensman, J., Fusi, N., & Lawrence, N. D. (2013c). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13 (pp. 282–290). Arlington, Virginia, USA: AUAI Press.

Hensman, J., Matthews, A., & Ghahramani, Z. (2015). Scalable Variational Gaussian Process Classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics* (pp. 351–360). San Diego, California, USA: PMLR.

Hoang, T. N., Hoang, Q. M., & Low, B. K. H. (2015a). A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research* (pp. 569–578). Lille, France: PMLR.

Hoang, T. N., Hoang, Q. M., & Low, B. K. H. (2015b). A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data. In *International Conference on Machine Learning*, volume 37 (pp. 569–578).: PMLR.

Hoff, P. (2021). Bayes-optimal prediction with frequentist coverage control. *arXiv preprint arXiv:2105.14045*.

Høie, M. H., Cagiada, M., Frederiksen, A. H. B., Stein, A., & Lindorff-Larsen, K. (2021). Predicting and interpreting large scale mutagenesis data using analyses of protein stability and conservation. *bioRxiv*.

Hornung, V., Hartmann, R., Ablasser, A., & Hopfner, K.-P. (2014). Oas proteins and cgas: unifying concepts in sensing and responding to cytosolic nucleic acids. *Nature Reviews Immunology*, 14(8), 521–528.

Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C. W., Xiao, C., Sun, J., & Zitnik, M. (2021). Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*.

Huber, M. F. (2013). Recursive gaussian process regression. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 3362–3366).: IEEE.

Huber, M. F. (2014). Recursive gaussian process: On-line regression and learning. *Pattern Recognition Letters*, 45, 85–91.

Ionescu, C., Vantzos, O., & Sminchisescu, C. (2015). Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE international conference on computer vision* (pp. 2965–2973).

Jankowiak, M., Pleiss, G., & Gardner, J. (2020a). Parametric Gaussian process regressors. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research* (pp. 4702–4712). Virtual: PMLR.

Jankowiak, M., Pleiss, G., & Gardner, J. (2020b). Parametric gaussian process regressors. In *International Conference on Machine Learning*, volume 119 (pp. 4702–4712).: PMLR.

Jensen, J. H. (2019). A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12), 3567–3572.

Jiang, S., Jiang, D., Balandat, M., Karrer, B., Gardner, J., & Garnett, R. (2020a). Efficient Nonmyopic Bayesian Optimization via One-Shot Multi-Step Trees. In *Advances in Neural Information Processing Systems*, volume 33.

Jiang, S., Jiang, D., Balandat, M., Karrer, B., Gardner, J., & Garnett, R. (2020b). Efficient nonmyopic bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33, 18039–18049.

Jin, W., Barzilay, R., & Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning* (pp. 2323–2332).: PMLR.

Jin, W., Wohlwend, J., Barzilay, R., & Jaakkola, T. (2021). Iterative refinement graph neural network for antibody sequence-structure co-design. *arXiv preprint arXiv:2110.04624.*

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4), 455–492.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873), 583–589.

Kandasamy, K., Dasarathy, G., Schneider, J., & Póczos, B. (2017). Multi-fidelity bayesian optimisation with continuous approximations. In *International Conference on Machine Learning* (pp. 1799–1808).: PMLR.

Kapoor, S., Karaletsos, T., & Bui, T. D. (2020). Variational auto-regressive gaussian processes for continual learning. *arXiv preprint arXiv:2006.05468.*

Khan, A., Cowen-Rivers, A. I., Deik, D.-G.-X., Grosnit, A., Dreczkowski, K., Robert, P. A., Greiff, V., Tutunov, R., Bou-Ammar, D., Wang, J., et al. (2022). Antbo: Towards real-world automated antibody design with combinatorial bayesian optimisation. *arXiv preprint arXiv:2201.12570.*

Khan, M. & Lin, W. (2017). Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. In *Artificial Intelligence and Statistics* (pp. 878–887).: PMLR.

Khan, Z. Y., Niu, Z., Sandiwarno, S., & Prince, R. (2021). Deep learning techniques for rating prediction: a survey of the state-of-the-art. *Artificial Intelligence Review*, 54(1), 95–135.

Kim, J., Park, S., Min, D., & Kim, W. (2021). Comprehensive survey of recent drug discovery using deep learning. *International Journal of Molecular Sciences*, 22(18), 9983.

Kirschner, J., Bogunovic, I., Jegelka, S., & Krause, A. (2020). Distributionally robust bayesian optimization. In *International Conference on Artificial Intelligence and Statistics* (pp. 2174–2184).: PMLR.

Knoblauch, J., Jewson, J., & Damoulas, T. (2019). Generalized variational inference. *arXiv preprint arXiv:1904.02063*.

Knudde, N., van der Herten, J., Dhaene, T., & Couckuyt, I. (2017). GPflowOpt: A Bayesian Optimization Library using TensorFlow. *arXiv preprint arXiv:1711.03845*.

Koppel, A. (2019). Consistent Online Gaussian Process Regression Without the Sample Complexity Bottleneck. In *2019 American Control Conference (ACC)* (pp. 3512–3518). ISSN: 2378-5861.

Krenn, M., Häse, F., Nigam, A., Friederich, P., & Aspuru-Guzik, A. (2020). Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4), 045024.

Krityakierne, T. & Ginsbourger, D. (2015). Global optimization with sparse and local gaussian process models. In *International Workshop on Machine Learning, Optimization and Big Data* (pp. 185–196).: Springer.

Kuhn, T. S. (1970). *The structure of scientific revolutions*, volume 111. Chicago University of Chicago Press.

Lambert, T. J. (2019). Fpbase: a community-editable fluorescent protein database. *Nature methods*, 16(4), 277–278.

Lanczos, C. (1950). *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA.

Lázaro-Gredilla, M. & Figueiras-Vidal, A. (2009). Inter-domain gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems* (pp. 1087–1095).

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., & Figueiras-Vidal, A. R. (2010). Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11, 1865–1881.

Lee, J., Mansimov, E., & Cho, K. (2018). Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*.

Letham, B., Karrer, B., Ottoni, G., Bakshy, E., et al. (2019). Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2), 495–519.

Liang, Q. & Lai, L. (2021). Scalable bayesian optimization accelerates process optimization of penicillin production. In *NeurIPS 2021 AI for Science Workshop*.

Lin, J., Obeng, A., & Bakshy, E. (2020). Preference learning for real-world multi-objective decision making. *ICML 2020 Workshop on Real World Experiment Design and Active Learning*.

Ling, C. K., Low, K. H., & Jaillet, P. (2016). Gaussian process planning with lipschitz continuous reward functions: Towards unifying bayesian optimization, active learning, and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lipinski, C. A., Lombardo, F., Dominy, B. W., & Feeney, P. J. (1997). Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3), 3–25.

Liu, X., Xue, W., Xiao, L., & Zhang, B. (2017). Pbodl: Parallel bayesian online deep learning for click-through rate prediction in tencent advertising system. *arXiv preprint arXiv:1707.00802*.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb), 419–444.

Lyu, J., Wang, S., Balius, T. E., Singh, I., Levit, A., Moroz, Y. S., O'Meara, M. J., Che, T., Algaa, E., Tolmachova, K., et al. (2019). Ultra-large library docking for discovering new chemotypes. *Nature*, 566(7743), 224–229.

MacKenzie, D. & Spears, T. (2014). 'the formula that killed wall street': The gaussian copula and modelling practices in investment banking. *Social Studies of Science*, 44(3), 393–417.

Maddox, W., Feng, Q., & Balandat, M. (2021a). Optimizing high-dimensional physics simulations via composite bayesian optimization. *arXiv preprint arXiv:2111.14911*.

Maddox, W. J., Balandat, M., Wilson, A. G., & Bakshy, E. (2021b). Bayesian optimization with high-dimensional outputs. *Advances in Neural Information Processing Systems*, 34.

Maddox, W. J., Stanton, S., & Wilson, A. G. (2021c). Conditioning sparse variational gaussian processes for online decision-making. *Advances in Neural Information Processing Systems*, 34.

Makarova, A., Usmanova, I., Bogunovic, I., & Krause, A. (2021). Risk-averse heteroscedastic bayesian optimization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems*, volume 34 (pp. 17235–17245).: Curran Associates, Inc.

Mania, H., Guy, A., & Recht, B. (2018). Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31 (pp. 1805–1814).

Marx, C., Zhao, S., Neiswanger, W., & Ermon, S. (2022). Modular conformal calibration. In *International Conference on Machine Learning* (pp. 15180–15195).: PMLR.

Matthews, A. G. d. G. (2017). *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge.

Maus, N., Jones, H. T., Moore, J. S., Kusner, M. J., Bradshaw, J., & Gardner, J. R. (2022). Local latent space bayesian optimization over structured inputs. *arXiv preprint arXiv:2201.11872*.

McIntire, M., Ratner, D., & Ermon, S. (2016). Sparse gaussian processes for bayesian optimization. In *Uncertainty in Artifical Intelligence*, volume 32: Association for Uncertainty in Artifical Intelligence (AUAI).

Meier, J., Rao, R., Verkuil, R., Liu, J., Sercu, T., & Rives, A. (2021). Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*.

Milios, D., Camoriano, R., Michiardi, P., Rosasco, L., & Filippone, M. (2018). Dirichlet-based gaussian processes for large-scale calibrated classification. In *Advances in Neural Information Processing Systems* (pp. 6005–6015).

Mishra, A., Ranganathan, S., Jayaram, B., & Sattar, A. (2018). Role of solvent accessibility for aggregation-prone patches in protein folding. *Scientific Reports*, 8(1), 12896.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Monard, F., Nickl, R., & Paternain, G. P. (2021). Statistical guarantees for bayesian uncertainty quantification in nonlinear inverse problems with gaussian process priors. *The Annals of Statistics*, 49(6), 3255–3298.

Moreno-Muñoz, P., Artés-Rodríguez, A., & Álvarez, M. A. (2019). Continual multi-task gaussian processes. *arXiv preprint arXiv:1911.00002*.

Moreno-Muñoz, P., Artés-Rodríguez, A., & Álvarez, M. A. (2020). Recyclable gaussian processes. *arXiv preprint arXiv:2010.02554*.

Moss, H. B., Beck, D., Gonzalez, J., Leslie, D. S., & Rayson, P. (2020). Boss: Bayesian optimization over string spaces. *arXiv preprint arXiv:2010.00979*.

Moss, H. B., Leslie, D. S., Gonzalez, J., & Rayson, P. (2021). Gibbon: General-purpose information-based bayesian optimisation. *arXiv preprint arXiv:2102.03324*.

Mukadam, M., Yan, X., & Boots, B. (2016). Gaussian process motion planning. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 9–15).: IEEE.

Muller, C., Rabal, O., & Diaz Gonzalez, C. (2022). Artificial intelligence, machine learning, and deep learning in real-life drug design cases. *Artificial Intelligence in Drug Design*, (pp. 383–407).

Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Neiswanger, W. & Ramdas, A. (2021). Uncertainty quantification using martingales for misspecified gaussian processes. In *Algorithmic Learning Theory* (pp. 963–982).: PMLR.

Nguyen-Tuong, D., Peters, J., & Seeger, M. (2008). Local Gaussian process regression for real time online model learning and control. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, volume 31 of *NIPS'08* (pp. 1193–1200). Vancouver, British Columbia, Canada: Curran Associates Inc.

Nickson, T., Osborne, M. A., Reece, S., & Roberts, S. (2014). Automated machine learning using stochastic algorithm tuning. In *NIPS Workshop on Bayesian Optimization*.

Nigam, A., Friederich, P., Krenn, M., & Aspuru-Guzik, A. (2019). Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *arXiv preprint arXiv:1909.11655*.

Opper, M. (1998). A bayesian approach to online learning. *On-line learning in neural networks*, (pp. 363–378).

Opper, M. & Archambeau, C. (2009). The variational gaussian approximation revisited. *Neural computation*, 21(3), 786–792.

Osborne, M. A. (2010). *Bayesian Gaussian processes for sequential prediction, optimisation and quadrature*. PhD thesis, Oxford University, UK.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., & Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32.

O'Donoghue, B., Osband, I., Munos, R., & Mnih, V. (2018). The uncertainty bellman equation and exploration. In *International Conference on Machine Learning* (pp. 3836–3845).

Pan, P., Swaroop, S., Immer, A., Eschenhagen, R., Turner, R. E., & Khan, M. E. (2020). Continual deep learning by functional regularisation of memorable past. *arXiv preprint arXiv:2004.14070*.

Pan, Y., Yan, X., Theodorou, E. A., & Boots, B. (2017). Prediction under uncertainty in sparse spectrum gaussian processes with applications to filtering and control. In *International Conference on Machine Learning*, volume 70 (pp. 2760–2768).: PMLR.

Panos, A., Dellaportas, P., & Titsias, M. K. (2018). Fully scalable gaussian processes using subspace inducing inputs. *arXiv preprint arXiv:1807.02537*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019a). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, volume 32 (pp. 8024–8035).: Curran Associates, Inc.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019b). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 8026–8037.

Pearlmutter, B. A. (1994). Fast exact multiplication by the hessian. *Neural computation*, 6(1), 147–160.

Pleiss, G., Gardner, J., Weinberger, K., & Wilson, A. G. (2018a). Constant-time predictive distributions for Gaussian processes. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research* (pp. 4114–4123). Stockholmsmässan, Stockholm Sweden: PMLR.

Pleiss, G., Gardner, J. R., Weinberger, K. Q., & Wilson, A. G. (2018b). Constant-Time Predictive Distributions for Gaussian Processes. In *Artificial Intelligence and Statistics*, volume 84: PMLR.

Pournader, M., Ghaderi, H., Hassanzadegan, A., & Fahimnia, B. (2021). Artificial intelligence applications in supply chain management. *International Journal of Production Economics*, 241, 108250.

Quinonero-Candela, J. & Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6, 1939–1959.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X., Canny, J., Abbeel, P., & Song, Y. S. (2019). Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 9689.

Rasmussen, C. E. & Williams, C. K. I. (2008). *Gaussian processes for machine learning*. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 3. print edition.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. (2021). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15).

Ruffolo, J. A., Gray, J. J., & Sulam, J. (2021). Deciphering antibody affinity maturation with language models and weakly supervised learning. *arXiv preprint arXiv:2112.07782*.

Sæmundsson, S., Hofmann, K., & Deisenroth, M. (2018). Meta reinforcement learning with latent variable gaussian processes. In *34th Conference on Uncertainty in Artificial Intelligence*, volume 34 (pp. 642–652).: Association for Uncertainty in Artificial Intelligence (AUAI).

Savage, L. J. (1972). *The foundations of statistics*. Courier Corporation.

Schymkowitz, J., Borg, J., Stricher, F., Nys, R., Rousseau, F., & Serrano, L. (2005). The foldx web server: an online force field. *Nucleic acids research*, 33(suppl_2), W382–W388.

Seeger, M. W., Williams, C. K., & Lawrence, N. D. (2003). Fast forward selection to speed up sparse gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics* (pp. 254–261).: PMLR.

Seo, S., Wallat, M., Graepel, T., & Obermayer, K. (2000). Gaussian process regression: Active data selection and test point rejection. In *Neural Networks, IEEE-INNS-ENNS International Joint Conference on*, volume 3 (pp. 3241–3241).

Shafer, G. & Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3).

Shah, A. & Ghahramani, Z. (2016). Pareto frontier learning with expensive correlated objectives. In *International Conference on Machine Learning* (pp. 1919–1927).: PMLR.

Shahriari, B., Bouchard-Côté, A., & Freitas, N. (2016). Unbounded bayesian optimization via regularization. In *Artificial intelligence and statistics* (pp. 1168–1176).: PMLR.

Shi, J., Titsias, M., & Mnih, A. (2020). Sparse orthogonal variational inference for gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, volume 108 (pp. 1932–1942).: PMLR.

Shrake, A. & Rupley, J. A. (1973). Environment and exposure to solvent of protein atoms. lysozyme and insulin. *Journal of molecular biology*, 79(2), 351–371.

Shu, R., Lee, J., Nakayama, H., & Cho, K. (2020). Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34 (pp. 8846–8853).

Siivola, E., Vehtari, A., Vanhatalo, J., González, J., & Andersen, M. R. (2018). Correcting boundary over-exploration deficiencies in bayesian optimization with virtual derivative sign observations. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)* (pp. 1–6).: IEEE.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676), 354–359.

Snelson, E. & Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems* (pp. 1257–1264).

Srinivas, N., Krause, A., Kakade, S., & Seeger, M. (2010). Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (pp. 1015–1022).

Stanton, S., Maddox, W., Gruver, N., Maffettone, P., Delaney, E., Greenside, P., & Wilson, A. G. (2022). Accelerating bayesian optimization for biological sequence design with denoising autoencoders. *arXiv preprint arXiv:2203.12742*.

Stanton, S., Maddox, W. J., Delbridge, I., & Wilson, A. G. (2021). Kernel Interpolation for Scalable Online Gaussian Processes. In *Artificial Intelligence and Statistics*, volume 130: PMLR.

Stutz, D., Cemgil, A. T., Doucet, A., et al. (2021). Learning optimal conformal classifiers. *arXiv preprint arXiv:2110.09192*.

Sugiyama, M., Suzuki, T., & Kanamori, T. (2012). *Density ratio estimation in machine learning*. Cambridge University Press.

Sun, T.-X., Liu, X.-Y., Qiu, X.-P., & Huang, X.-J. (2022). Paradigm shift in natural language processing. *Machine Intelligence Research*, 19(3), 169–183.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4), 285–294.

Tibshirani, R. J., Foygel Barber, R., Candes, E., & Ramdas, A. (2019). Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32.

Titsias, M. (2008). Variational Model Selection for Sparse Gaussian Process Regression. *www.aueb.gr/users/mtitsias/papers/sparseGPv2.pdf*, (pp.20).

Titsias, M. (2009a). Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *Artificial Intelligence and Statistics* (pp. 567–574).: PMLR. ISSN: 1938-7228.

Titsias, M. (2009b). Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics* (pp. 567–574).

Titsias, M. K., Schwarz, J., Matthews, A. G. d. G., Pascanu, R., & Teh, Y. W. (2019). Functional regularisation for continual learning with gaussian processes. In *International Conference on Learning Representations*.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5026–5033).: IEEE.

Trefethen, L. N. & Bau III, D. (1997). *Numerical linear algebra*, volume 50. Siam.

Tripp, A., Daxberger, E., & Hernández-Lobato, J. M. (2020). Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33.

Tripp, A., Simm, G. N., & Hernández-Lobato, J. M. (2021). A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*.

Van Der Vaart, A. & Van Zanten, H. (2011). Information rates of nonparametric gaussian process methods. *Journal of Machine Learning Research*, 12(6).

Van der Wilk, M. (2019). *Sparse Gaussian process approximations and applications*. PhD thesis, University of Cambridge.

Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic learning in a random world*. Springer Science & Business Media.

Vovk, V., Shen, J., Manokhin, V., & Xie, M.-g. (2017). Nonparametric predictive distributions based on conformal prediction. In *Conformal and Probabilistic Prediction and Applications* (pp. 82–102).: PMLR.

Wang, K. A., Pleiss, G., Gardner, J. R., Tyree, S., Weinberger, K. Q., & Wilson, A. G. (2019). Exact Gaussian Processes on a Million Data Points. In *Advances in Neural Information Processing Systems*, volume 32. arXiv: 1903.08114.

Wang, L., Fonseca, R., & Tian, Y. (2020). Learning search space partition for black-box optimization using monte carlo tree search. In *Advances in Neural Information Processing Systems*, volume 33.

Wang, Z., Gehring, C., Kohli, P., & Jegelka, S. (2018). Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, volume 84 (pp. 745–754).: PMLR.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., & de Feitas, N. (2016). Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55, 361–387.

Wang, Z. & Jegelka, S. (2017). Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, volume 70 (pp. 3627–3635).: PMLR.

Wasserman, L. (2008). Comment on article by gelman. *Bayesian Analysis*, 3(3), 463–465.

Weiss, D. J., Lucas, T. C., Nguyen, M., Nandi, A. K., Bisanzio, D., Battle, K. E., Cameron, E., Twohig, K. A., Pfeffer, D. A., Rozier, J. A., et al. (2019). Mapping the global prevalence, incidence, and mortality of plasmodium falciparum, 2000–17: a spatial and temporal modelling study. *The Lancet*, 394(10195), 322–331.

Welling, M. & Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (pp. 681–688).: Citeseer.

Williams, C. K. & Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Wilson, A. & Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning* (pp. 1067–1075).

Wilson, A. & Ghahramani, Z. (2010). Copula Processes. In *Advances in Neural Information Processing Systems*, volume 23 (pp.9).

Wilson, A. & Nickisch, H. (2015). Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP). In *International Conference on Machine Learning* (pp. 1775–1784). ISSN: 1938-7228 Section: Machine Learning.

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016a). Deep kernel learning. In *Artificial intelligence and statistics* (pp. 370–378).: PMLR.

Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016b). Deep Kernel Learning. In *Artificial Intelligence and Statistics*. arXiv: 1511.02222.

Wilson, J. T., Moriconi, R., Hutter, F., & Deisenroth, M. P. (2017). The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*.

Wouters, O. J., McKee, M., & Luyten, J. (2020). Estimated research and development investment needed to bring a new medicine to market, 2009-2018. *Jama*, 323(9), 844–853.

Wu, J. & Frazier, P. I. (2016). The parallel knowledge gradient method for batch Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 30 of *NIPS'16* (pp. 3134–3142). Red Hook, NY, USA: Curran Associates Inc.

Wu, J., Toscano-Palmerin, S., Frazier, P. I., & Wilson, A. G. (2019). Practical multi-fidelity bayesian optimization for hyperparameter tuning. *arXiv pre-print arXiv:1903.04703*.

Wynne, G., Briol, F.-X., & Girolami, M. (2021). Convergence guarantees for gaussian process means with misspecified likelihoods and smoothness. *Journal of Machine Learning Research*, 22.

Xu, N., Low, K. H., Chen, J., Lim, K. K., & Ozgul, E. B. (2014). Gp-localize: Persistent mobile robot localization using online sparse gaussian process observation model. *arXiv preprint arXiv:1404.5165*, 28.

Yamashita, T., Sato, N., Kino, H., Miyake, T., Tsuda, K., & Oguchi, T. (2018). Crystal structure prediction accelerated by bayesian optimization. *Physical Review Materials*, 2(1), 013803.

Yang, K. K., Chen, Y., Lee, A., & Yue, Y. (2019a). Batched stochastic bayesian optimization via combinatorial constraints design. In *The 22nd International Conference on Artificial Intelligence and Statistics* (pp. 3410–3419).: PMLR.

Yang, K. K., Wu, Z., & Arnold, F. H. (2019b). Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8), 687–694.

Zaytsev, A., Romanenkova, E., & Ermilov, D. (2018). Interpolation error of gaussian process regression for misspecified case. In *Conformal and Probabilistic Prediction and Applications* (pp. 83–95).: PMLR.

Zhang, D., Fu, J., Bengio, Y., & Courville, A. (2021). Unifying likelihood-free inference with black-box sequence design and beyond. *arXiv preprint arXiv:2110.03372*.