On Safety in Safe Bayesian Optimization

Anonymous authors
Paper under double-blind review

Abstract

Safe Bayesian Optimization (BO) is increasingly used to optimize an unknown function under safety constraints, a central task in robotics, biomedical engineering, and many other disciplines. Due to the safety-critical nature of these applications, it is crucial that theoretical safety guarantees for these algorithms translate into the real world. In this work, we investigate three safety-related issues in SafeOpt-type algorithms, a popular class of safe BO methods. First, these algorithms critically rely on frequentist uncertainty bounds for Gaussian Process (GP) regression, but concrete implementations typically utilize heuristics that invalidate all safety guarantees. We provide a detailed analysis of this problem and introduce Real-β-SafeOpt, a variant of the SafeOpt algorithm that leverages recent GP bounds and thus retains all theoretical guarantees. Second, we identify a key technical assumption in SafeOpt-like algorithms, the assumption of an upper bound on the reproducing kernel Hilbert space (RKHS) norm of the target function, as a central obstacle to real-world usage. To overcome this obstacle, we introduce the Lipschitz-only Safe Bayesian Optimization (LoSBO) algorithm, which guarantees safety without an assumption on the RKHS bound, and show empirically that this algorithm is not only safe, but also outperforms the state-of-the-art on several function classes. Third, SafeOpt and derived algorithms rely on a discrete search space, complicating their application to higher-dimensional problems. To broaden the applicability of these algorithms, we introduce Lipschitz-only Safe GP-UCB (LoS-GP-UCB), a LoSBO variant that is applicable to moderately high-dimensional problems, while retaining safety. By analyzing practical safety issues in an important class of safe BO algorithms, and providing ready-to-use algorithms that overcome these issues, this work contributes to bringing safe and reliable machine learning techniques closer to real world applications.

1 Introduction

In science, engineering, and business it is often necessary to optimize an unknown target function. Typically, such functions are expensive to evaluate and only noisy function values are available. If it is possible to actively query the function, i.e., to select the inputs that are to be evaluated, this problem is commonly addressed using Bayesian Optimization (BO), see (Shahriari et al., 2015) or (Garnett, 2023) for an overview. However, in many real-world applications, BO algorithms should avoid the use of certain inputs, frequently for safety reasons. For example, the target function might be a reward function for a robotic task, with the input a control policy for a physical robot. In this example, any inputs leading to physical damage or unsafe behavior should be avoided. An important special case of such a safety constraint is the restriction of query inputs to those with function values not lower than a given threshold (Kim et al., 2020). Inputs that satisfy this constraint are termed safe inputs, and a BO algorithm is considered safe if it queries only safe inputs throughout its run. This type of safety constraint has been introduced in (Sui et al., 2015) and arises for example in biomedical applications (where the target function is patient comfort and the input corresponds to treatment settings) or controller tuning (where the target function is a measure of controller performance and the inputs are tuning parameters).

A popular BO algorithm for this problem setting is SafeOpt (Sui et al., 2015). Starting from a given set of safe inputs, this algorithm iteratively searches for a maximum while aiming to avoid unsafe inputs with high

probability. It achieves this by utilizing Gaussian Processes (GPs) together with a frequentist uncertainty bound (Srinivas et al., 2010; Chowdhury & Gopalan, 2017) *and* a known upper bound on the Lipschitz constant of the target function. Provided the algorithmic parameters are set correctly, then with (high) pre-defined probability, SafeOpt demonstrably converges to the safely reachable maximum while avoiding unsafe inputs. SafeOpt and its variants have been used in various applications, e.g., safe controller optimization (Berkenkamp et al., 2016b), automated deep brain stimulation (Sarikhani et al., 2021) and safe robot learning (Baumann et al., 2021).

To ensure safety, SafeOpt and its variants require frequentist uncertainty sets that are valid (i.e., they hold with specified high probability) and explicit (i.e., they can be numerically evaluated). However, two issues arise here: First, the uncertainty bounds from (Srinivas et al., 2010; Chowdhury & Gopalan, 2017) used in most SafeOpt-type algorithms tend to be conservative, even if all necessary ingredients for their computation are available. This restriction can entirely prevent exploration of the target function. Second, such uncertainty bounds rely upon a particular property of the target function (a known finite upper bound on the Reproducing Kernel Hilbert Space (RKHS) norm, see Section 2.2 for details), which in practice is very difficult to derive from reasonable prior knowledge. As a consequence of these issues, algorithmically usable uncertainty sets are not yet available for SafeOpt. Indeed, to the best of our knowledge, all implementations of SafeOpt and its variants have instead used heuristics (Berkenkamp et al., 2016b; Kirschner et al., 2019a; Baumann et al., 2021; Koller et al., 2019; Helwa et al., 2019).[1] This shortcoming means that in practice such implementations lose all their theoretical safety guarantees. In this work, we carefully investigate this issue and propose practical solutions, bringing safe BO closer to real-world usage.

Outline In Section 2, we review the necessary technical background on Gaussian Process (GP) regression, reproducing kernel Hilbert spaces (RKHSs), and frequentist uncertainty bounds for GP regression, while in Section 3 we present the problem setting and our objectives, including a detailed description of the original SafeOpt algorithm. Section 4 provides a comprehensive discussion of related work and our investigation into safety aspects of SafeOpt begins in Section 5, with a discussion of practical problems arising from the use of heuristics in SafeOpt-type algorithms and a demonstration of these issues via numerical experiments. To overcome these problems, we propose using state-of-the-art uncertainty bounds in the current algorithms, leading to an algorithmic variant we call Real-β-SafeOpt, which also forms the foundation for the following numerical experiments. In Section 6, we discuss the central assumption of a known upper bound on the RKHS norm of the target function, and highlight the practical safety problems it poses. To address these problems, we introduce Lipschitz-only Safe Bayesian Optimization (LoSBO), a SafeOpt-type algorithm that does not rely on this assumption for safety. We also prove appropriate safety guarantees for LoSBO, and use numerical experiments to thoroughly evaluate the algorithm. To enable this type of safe BO to be applied in high dimensions, Section 7 introduces another algorithmic variant, called Lipschitz-only Safe Gaussian Process-Upper Confidence Bound (LoS-GP-UCB). We describe and discuss this algorithm in detail, and perform numerical experiments to evaluate it empirically. Finally, Section 8 closes the article with a summary and outlook. The source code for all experiments is available in the anonymous repository.[2]

2 Background

In this section, we provide a brief review of Gaussian Process (GP) regression, reproducing kernel Hilbert spaces (RKHSs), and frequentist uncertainty bounds for GP regression, as these three components form the foundations for SafeOpt-type algorithms. In our review of frequentist uncertainty bounds for GP regression, we also point out a computable form of such an uncertainty bound, which is based on existing theory, but which seems to have been underutilized in the literature.

[1] The precise experimental settings are not reported in (Sui et al., 2015). However, based on the descriptions provided, it can be inferred that some form of heuristic was used.

[2] https://anonymous.4open.science/r/losbo-482C

2.1 Gaussian processes

A GP is a collection of $\mathbb{R}$-valued random variables, here indexed by the set D, such that every finite collection of those random variables has a multivariate normal distribution. A GP g is uniquely defined by its mean function $m(x) = \mathbb{E}[g(x)]$ and covariance function $k(x, x') = \mathbb{E}[(g(x) - m(x))((g(x') - m(x'))]$, and we denote such a GP by $g \sim \mathcal{GP}_D(m, k)$. In GP regression, we start with a prior GP. Assuming independent and identically distributed (i.i.d.) additive normal noise, $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, this prior GP can be updated with data $(x_1, y_1), \ldots, (x_t, y_t)$, leading to a posterior GP. Without loss of generality we assume that the prior GP has zero mean. Then the posterior mean, posterior covariance and posterior variance are given by $\mu_t(x) = \boldsymbol{k}_t(x)^T(\boldsymbol{K}_t + \sigma^2\boldsymbol{I}_t)^{-1}\boldsymbol{y}_t$, $k_t(x, x') = k(x, x') - \boldsymbol{k}_t(x)^T(\boldsymbol{K}_t + \sigma^2\boldsymbol{I}_t)^{-1}\boldsymbol{k}_t(x')$, and $\sigma_t^2(x) = k_t(x, x)$, respectively, where $\boldsymbol{y}_t = [y_1, ..., y_t]^T$ is the vector of observed, noisy function values of f, the kernel matrix $\boldsymbol{K}_t \in \mathbb{R}^{t\times t}$ has entries $[k(x, x')]_{x,x'\in D_t}$, the vector $\boldsymbol{k}_t(x) = [k(x_1, x) \cdots k(x_t, x)]^T$ contains the covariances between x and the observed data points, and I_t is the $t \times t$ identity matrix.

In practice, the prior mean, prior covariance function, and noise level, are (partially) chosen based on prior knowledge. If no specific prior knowledge regarding the mean is available, the zero function is usually chosen. Furthermore, in practice these three components are only partially specified, usually up to some parameters, which are then called hyperparameters in the context. In BO, the latter are often determined during the optimization via hyperparameter optimization (Garnett, 2023).

For more details on GPs, GP regression, and related method, we refer to (Rasmussen & Williams, 2006) and (Garnett, 2023).

2.2 Reproducing kernel Hilbert spaces

Consider a function $k : D \times D \to \mathbb{R}$. We call k *positive semidefinite* if for all $x_1, \ldots, x_N \in D$, $N \in \mathbb{N}_{>0}$, the matrix $\big(k(x_i, x_j)\big)_{i,j=1,\ldots,N}$ is positive semidefinite. Equivalently, the function k is symmetric ($k(x_i, x_j) = k(x_j, x_i)$ for all $i, j = 1, \ldots, N$), and for all $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$ we have $\sum_{i,j=1}^N \alpha_i\alpha_j k(x_i, x_j) \geq 0$. In the literature such a function is often called *positive definite* or *of positive type*. Additionally, k is called *positive definite*, if for all pairwise distinct $x_1, \ldots, x_N \in D$, the matrix $\big(k(x_i, x_j)\big)_{i,j=1,\ldots,N}$ is positive definite. This property is sometimes referred to as *strict positive definiteness* in the literature.

Let H be a Hilbert space functions on D. We call H a *reproducing kernel Hilbert space (RKHS)* if every evaluation functional is continuous, i.e., for all $x \in D$ the mapping $H \ni f \mapsto f(x) \in \mathbb{R}$ is continuous w.r.t. the norm induced by the scalar product in H. Furthermore, k is called a *reproducing kernel (for H)* if 1) $k(\cdot, x) \in H$ for all $x \in D$, and 2) $f(x) = \langle f, k(\cdot, x)\rangle_H$ for all $f \in H$ and $x \in D$.

As is well-known, H is a RKHS if and only if it has a reproducing kernel, and in this case the latter is unique (Steinwart & Christmann, 2008, Lemma 4.19, Theorem 4.20). Furthermore, every reproducing kernel is positive semidefinite, and every positive semidefinite function is a reproducing kernel for a unique RKHS (Steinwart & Christmann, 2008, Theorem 4.16, 4.21). If k is positive semidefinite, then we denote its unique RKHS as $(H_k, \langle\cdot,\cdot\rangle_k)$, and the induced norm by $\|\cdot\|_k$. Furthermore, we define the *pre RKHS* by

$$H_k^{\text{pre}} = \operatorname{span}\{k(\cdot, x) \mid x \in D\} = \left\{\sum_{n=1}^N \alpha_n k(\cdot, x_n) \mid N \in \mathbb{N}_{>0}, \alpha_n \in \mathbb{R}, x_n \in D,\ n = 1, \ldots, N\right\}, \tag{1}$$

and this subspace is dense in H_k w.r.t. $\|\cdot\|_k$. Given $f = \sum_{n=1}^N \alpha_n k(\cdot, x_n)$, $g = \sum_{m=1}^M \beta_m k(\cdot, y_m) \in H_k^{\text{pre}}$, we have

$$\langle f, g\rangle_k = \sum_{n=1}^N \sum_{m=1}^M \alpha_n\beta_m k(y_m, x_n), \tag{2}$$

see (Steinwart & Christmann, 2008, Theorem 4.21).

If k is the covariance function of a GP, then it is positive semidefinite (since every covariance matrix is positive semidefinite), and hence the reproducing kernel of a unique RKHS. Conversely, if k is the reproducing kernel of a RKHS, then k is positive semidefinite, there exists a GP having k as its covariance function, and the GP can be chosen with a zero mean function (Berlinet & Thomas-Agnan, 2004).

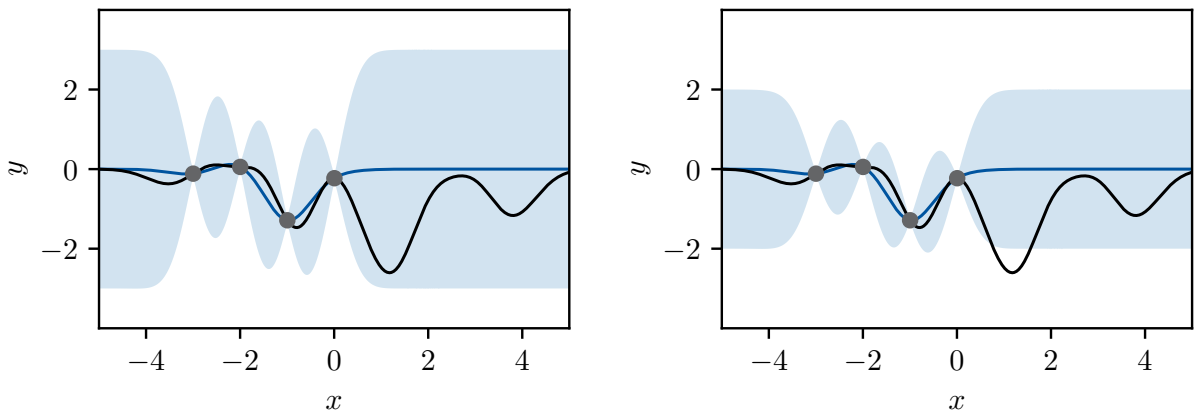

Figure 1: Illustration of the required GP error bounds. Consider a fixed ground truth (solid black line), of which only finitely many samples are known (black dots). Applying GP regression leads to a posterior GP fully described by the posterior mean (solid blue line) and the posterior variance, from which a high-probability uncertainty set can be derived (shaded blue). Left: The ground truth is completely contained in the uncertainty set. Right: The ground truth violates the uncertainty bound around $x = 1$.

Furthermore, consider GP regression with a prior $\mathcal{GP}_D(m, k)$, then $\mu_t - m \in H_k^{\text{pre}}$, where μ_t is the posterior mean corresponding to a finite data set with t points. In particular, the posterior mean for a zero mean prior GP is always in the pre RKHS corresponding to the covariance function. As is customary in machine learning with GPs (Rasmussen & Williams, 2006), and also in many BO scenarios (Garnett, 2023), in the following we will use a zero mean GP prior, $m \equiv 0$, without loss of generality.

2.3 Frequentist uncertainty bounds

An important ingredient in SafeOpt-type algorithms are upper and lower bounds on the unknown target function, and these bounds have to hold uniformly in both time and input space. If we adopt a stochastic setup, then this can be formalized by finding upper and lower bounds such that for a given user-specified confidence $\delta \in (0, 1)$, we have

$$\mathbb{P}\left[\ell_t(x) \leq f(x) \leq u_t(x),\, \forall x \in D, t \geq 0\right] \geq 1 - \delta,$$

where the probability is with respect to the data-generating process, rather than to the target function f, which is fixed. In GP regression, the posterior mean μ_t can be interpreted as a nominal estimate of f, and the posterior variance σ_t^2 as a measure of uncertainty of this estimate. However, using the posterior variance to build upper and lower bounds in the SafeOpt setting is not straightforward. Firstly, the posterior variance is a *pointwise* measure of uncertainty about the ground truth, but the upper and lower bounds have to hold *uniformly* over the input set, and also uniformly in time. Secondly, GP regression is by its nature a Bayesian method. However, the SafeOpt setting is a typical frequentist setup – we have a fixed, yet unknown ground truth, about which we receive noisy information. In particular, any stochasticity arises only through the data-generating process (e.g., via random noise on the function values), and not through epistemic uncertainty, as is the case in the Bayesian setup. This difficulty is well-known, see (Fiedler et al., 2021a), and is particularly relevant in the context of robust control and related areas (Fiedler et al., 2021b). We thus need bounds $(\nu_t)_{t\geq 1}$, such that for a user-specified $\delta \in (0, 1)$ we have

$$\mathbb{P}\left[|f(x) - \mu_t(x)| \leq \nu_t(x)\, \forall x \in D, t \geq 1\right] \geq 1 - \delta. \tag{3}$$

The bounding function ν_t must only depend on data collected up to time t, as well as reasonable prior knowledge about f and the data-generating process, e.g., about the noise statistics. These bounds are illustrated in Figure 1. All common bounds are of the form

$$\nu_t(x) = \beta_t \sigma_t(x), \tag{4}$$

where $\beta_t \in \mathbb{R}_{\geq 0}$ is some scaling factor.

Let k be the covariance function used in GP regression, and H_k the unique RKHS with k as its reproducing kernel. Assume that the ground truth is contained in this RKHS, i.e., $f \in H_k$. Let $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$ be a filtration defined on the underlying probability space, and assume that the sequence of inputs $(x_t)_{t\geq 0}$ chosen by the algorithm is adapted to $\mathbb{F}$.[3]

The first bound of the form (4) was introduced in the seminal work (Srinivas et al., 2010), cf. their Theorem 6, which holds in the case of bounded noise. This is also the bound that was used in the original SafeOpt paper Sui et al. (2015). At the moment, the most commonly used uncertainty bound in the analysis of SafeOpt-type algorithms is (Chowdhury & Gopalan, 2017, Theorem 2). Assume that $(\epsilon_t)_t$ is a martingale difference sequence that is conditionally R-subgaussian w.r.t. $\mathbb{F}$ for some $R \in \mathbb{R}_{\geq 0}$, i.e., for all $t \geq 1$

$$\mathbb{E}\left[\exp(\nu\epsilon_t) \mid \mathcal{F}_t\right] \leq \exp\left(\frac{R^2\nu^2}{2}\right) \ \mathbb{P}\text{-a.s.} \quad \forall \nu \in \mathbb{R}.$$

Additionally, assume that $\lambda > 1$, or $\lambda \geq 1$ and the covariance function k is positive definite, where λ is the nominal noise variance used in GP regression. Under these conditions, the uncertainty bound (3) holds with

$$\beta_t = \|f\|_k + R\sqrt{2(\gamma_{t-1} + 1 + \log(1/\delta))} \tag{5}$$

in (4), where γ_t is the *maximum information gain* after t rounds, cf. (Srinivas et al., 2010) for a thorough discussion of this quantity. In contrast to (Srinivas et al., 2010, Theorem 6), this bound allows subgaussian noise (including bounded noise and normal noise) and involves only fairly small numerical constants. However, it still requires the maximum information gain or an upper bound thereof, which can be difficult to work with in practice, and it introduces some conservatism.

Motivated by these shortcomings, Fiedler et al. (2021a) proposed a data-dependent scaling factor in (4), based on (Chowdhury & Gopalan, 2017, Theorem 2). Assume the same setting as this latter result, and also assume that the covariance function k is positive definite, then we can set

$$\beta_t = \|f\|_k + \frac{R}{\sqrt{\lambda}}\sqrt{\ln\left(\det(\bar{\lambda}/\lambda \mathbf{K}_t + \bar{\lambda}\mathbf{I}_t)\right) - 2\ln(\delta)}, \tag{6}$$

where we defined $\bar{\lambda} = \max\{1, \lambda\}$, and λ is again the nominal noise variance used in GP regression, corresponding to the regularization parameter in kernel ridge regression. This bound no longer involves the maximum information gain, and numerical experiments demonstrate that the resulting uncertainty bounds are rarely significantly larger than common heuristics, see (Fiedler et al., 2021a). In fact, the bounds are sufficiently small to use in algorithms, e.g. (Fiedler et al., 2021b).

Finally, from the results in the doctoral thesis (Abbasi-Yadkori, 2013), which was published in 2012, an uncertain bound can be deduced that is superior to (Chowdhury & Gopalan, 2017, Theorem 2), and therefore also improves over (6). Consider the same setting as introduced above. Combining Theorem 3.11 with Remark 3.13 in (Abbasi-Yadkori, 2013), we find that for all $\delta \in (0, 1)$ we can set

$$\beta_t = \|f\|_k + \frac{R}{\sqrt{\lambda}}\sqrt{2\ln\left(\frac{1}{\delta}\det\left(\mathbf{I}_t + \frac{1}{\lambda}\mathbf{K}_t\right)\right)} \tag{7}$$

in (4). Like (6), this bound can easily be evaluated, though it seems that (7) has not appeared explicitly before. Interestingly, this result appears to have been infrequently used in the machine learning community, though it has recently been rediscovered in for example (Whitehouse et al., 2024). Observe that for $0 < \lambda < 1$ and k positive definite,

$$\begin{aligned}\frac{R}{\sqrt{\lambda}}\sqrt{2\ln\left(\frac{1}{\delta}\det\left(\mathbf{I}_t + \frac{1}{\lambda}\mathbf{K}_t\right)\right)} &= \frac{R}{\sqrt{\lambda}}\sqrt{\ln\left(\det(1/\lambda\mathbf{K}_t + \mathbf{I}_t)\right) - 2\ln(\delta)} \\ &= \frac{R}{\sqrt{\lambda}}\sqrt{\ln\left(\det(\bar{\lambda}/\lambda\mathbf{K}_t + \bar{\lambda}\mathbf{I}_t)\right) - 2\ln(\delta)},\end{aligned}$$

[3]Of course, this requires that D is a measurable space, which is not a problem in practice.

Algorithm 1 Generic SafeOpt-type algorithm

Initialize model $\mathcal{M}_0$
for $t = 1, 2, \ldots$ **do**
  Compute current model $\mathcal{M}_t$
  Compute S_t from $\mathcal{M}_t$
  Compute G_t from S_t and $\mathcal{M}_t$
  Compute M_t from S_t and $\mathcal{M}_t$
  $x_t \leftarrow \operatorname{argmax}_{x \in G_t \cup M_t} w_t(x)$
  Query function with x_t, receive $y_t = f(x_t) + \epsilon_t$
  Update model with (x_t, y_t)
end for

so in this case (Fiedler et al., 2021a, Theorem 1) reproduces the result from (Abbasi-Yadkori, 2013). Since the only difference between (6) and (7) happens inside $\sqrt{\ln(\cdot)}$, any noticeable difference between the two bounds will happen for $\lambda >> 1$, so any difference will be negligible in practice.

3 Problem setting and objectives

We now formalize our problem setting, describe SafeOpt-type algorithms in detail, and specify our objectives for the remainder of this work.

We work in the setting introduced by the seminal paper (Sui et al., 2015). Consider a nonempty set D, the *input set*, and a fixed, but unknown function $f : D \to \mathbb{R}$, the *target function* or *ground truth*. We are interested in an algorithm that finds the maximum of f by iteratively querying the function. At time step $t \in \mathbb{N}_0$, such an algorithm chooses an input $x_t \in D$ and receives a noisy function evaluation $y_t = f(x_t) + \epsilon_t$, where ϵ_t is additive measurement noise. As a safety constraint, all chosen inputs must have a function value above a given safety threshold $h \in \mathbb{R}$, i.e., $f(x_t) \geq h$ for all t. Furthermore, the algorithm should be sample-efficient, i.e., use as few function queries as possible to find an input with a high function value. To make progress on this problem, it is clear that some restriction on the function f must be posed. Central to our developments is the next assumption.

Assumption 1. *D is equipped with a metric $d : D \times D \to \mathbb{R}_{\geq 0}$. Additionally, f is L-Lipschitz continuous, where $L \in \mathbb{R}_{\geq 0}$ is a known Lipschitz constant.*

The second assumption means that for all $x, x' \in D$, we have $|f(x) - f(x')| \leq L\, d(x, x')$. From now on, we work under Assumption 1. Furthermore, we assume that we have access to a non-empty set of known safe inputs $S_0 \subseteq D$, i.e., for all $x \in S_0$ we have $f(x) \geq h$. The SafeOpt algorithm and its derivatives use an iteratively updated model $\mathcal{M}_t$ that provides estimated upper and lower bounds u_t and ℓ_t on f, i.e., with a certain confidence it holds that $\ell_t(x) \leq f(x) \leq u_t(x)$ for all $x \in D$ and all $t \geq 1$. These bounds are also used to provide a measure of uncertainty defined as $w_t(x) = u_t(x) - \ell_t(x)$. In each step $t \geq 1$, the previous model $\mathcal{M}_{t-1}$ together with the Lipschitz assumption is used to determine a new set $S_t \subseteq D$ of safe inputs, starting from the initial safe set S_0. Subsequently, a set $M_t \subseteq S_t$ of potential maximizers of the target function, and a set $G_t \subseteq S_t$ of potential expanders is computed. The latter contains inputs that are likely to lead to new safe inputs upon query. Finally, the target function is queried at the input $x_t = \operatorname{argmax}_{x \in G_t \cup M_t} w_t(x)$, a noisy function value y_t is received, and the model $\mathcal{M}_{t-1}$ is updated with the data point (x_t, y_t). Pseudocode for a generic version of SafeOpt is provided by Algorithm 1. Different variants of SafeOpt result from different choices of models and computations of S_t, M_t and G_t.

To the best of our knowledge, in all SafeOpt-type algorithms, the unknown ground truth f is modeled as a GP. To compute appropriate upper and lower bounds, it is assumed that an appropriate scaling factor β_t is available, cf. (4). For each time step t, define $C_t(x) = C_{t-1} \cap Q_t(x)$, where $Q_t(x) = [\mu_{t-1}(x) \pm \beta_t\, \sigma_{t-1}(x)]$, and, starting with $Q_0(x) = \mathbb{R}$ for all $x \in D$, $C_0(x) = [h, \infty)$ for all $x \in S_0$ and $C_0(x) = \mathbb{R}$ for $x \in D \setminus S_0$. The corresponding estimated upper and lower bounds are given by $u_t(x) = \max C_t(x)$ and $\ell_t(x) = \min C_t(x)$, respectively. In the original SafeOpt algorithm from (Sui et al., 2015), for each step $t \geq 1$, the new safe sets

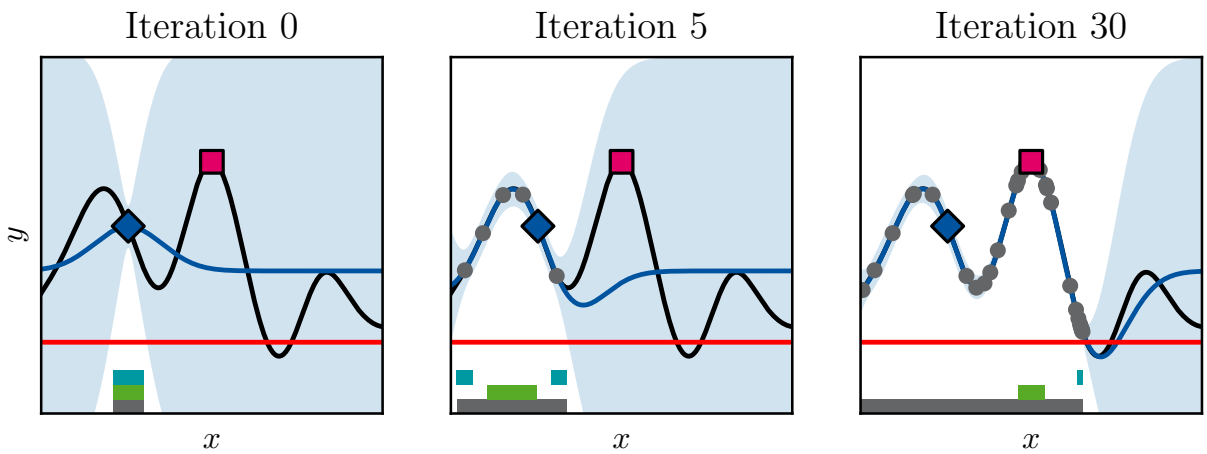

Figure 2: Illustration of SafeOpt. The safe set (gray bar), expanders (green bar), and maximizers (blue bar), derived from the current GP model (with solid blue line the posterior mean, shaded blue areas the uncertainty sets), are used to find the safely reachable optimum (red box). In each iteration, the next input is chosen from the union of the current expanders and maximizers (a subset of the safe set) by maximizing the acquisition function.

are calculated by

$$S_t = \bigcup_{x \in S_{t-1}} \{x' \in D \,|\, \ell_t(x) - L\, d(x, x') \geq h\}, \tag{8}$$

the potential maximizers are given by

$$M_t = \{x \in S_t \mid u_t(x) \geq \max_{x_S \in S_t} \ell_t(x_S)\}, \tag{9}$$

and the potential expanders by

$$G_t = \{x_S \in S_t \mid \exists x \in D \setminus S_t : \; u_t(x_S) - Ld(x_S, x) \geq h\}. \tag{10}$$

The resulting algorithm is illustrated in Figure 2. A formal description of the algorithm using pseudocode is provided by Algorithm 2. In some popular variants of the SafeOpt algorithm, no Lipschitz bound is used in the computation of the safe sets (Berkenkamp et al., 2016b). However, since knowledge of such a Lipschitz bound is additional knowledge that should be used by the algorithm, and we strongly rely on this assumption from Section 6 onwards, we do not consider these algorithmic variants in the present work.

Our primary objective is to investigate and improve practically relevant safety aspects of SafeOpt-type algorithms. We will consider three specific objectives.

(O1) While SafeOpt and related algorithms come with precise theoretical safety guarantees, all known implementations to date use heuristics (usually by setting β_t to some constant) instead of theoretically sound uncertainty bounds. These heuristics generally invalidate all theoretical safety guarantees, raising the question of whether their use leads to safety problems in practice.

(O2) Without heuristics, safety guarantees for SafeOpt-type algorithms rely on the appropriateness of the assumptions used in the algorithmic setup. Since SafeOpt and its variants are adopted for scenarios with stringent safety requirements where no failure should occur with high probability and no tuning phase is allowed, it is important that all underpinning assumptions are reasonable and verifiable by users. The most delicate of these assumptions is knowledge of an upper bound of the RKHS norm of the target function, raising the question of whether the RKHS norm bound assumption is reasonable, and if not, how it can be replaced.

Algorithm 2 SafeOpt

Require: Lipschitz constant L, algorithm to compute β_t, initial safe set S_0, safety threshold h

$Q_0(x) \leftarrow \mathbb{R}$ for all $x \in D$ ▷ Initialization of uncertainty sets
$C_0(x) \leftarrow [h, \infty)$ for all $x \in S_0$
$C_0(x) \leftarrow \mathbb{R}$ for all $x \in D \setminus S_0$
for $t = 1, 2, \ldots$ **do**
  $C_t(x) \leftarrow C_{t-1}(x) \cap Q_{t-1}(x)$ for all $x \in D$ ▷ Compute upper and lower bounds for current iteration
  $\ell_t(x) \leftarrow \min C_t(x)$, $u_t(x) \leftarrow \max C_t(x)$ for all $x \in D$
  if $t > 1$ **then** ▷ Compute new safe set
    $S_t = S_{t-1} \cup \{x \in D \mid \exists x_s \in S_{t-1} : \ell_t(x_s) - Ld(x_s, x) \geq h\}$
  else
    $S_1 = S_0$
  end if
  $G_t \leftarrow \{x \in S_t \mid \exists x' \in D \setminus S_t : u_t(x) - Ld(x, x') \geq h\}$ ▷ Compute set of potential expanders
  $M_t = \{x \in S_t \mid u_t(x) \geq \max_{x_S \in S_t} \ell_t(x_S)\}$ ▷ Compute set of potential maximizers
  $x_t \leftarrow \operatorname{argmax}_{x \in G_t \cup M_t} w_t(x)$ ▷ Determine next input
  Query function with x_t, receive $y_t = f(x_t) + \epsilon_t$
  Update GP with new data point (x_t, y_t), resulting in mean μ_t and σ_t
  Compute updated β_t
  $Q_t(x) = [\mu_t(x) - \beta_t \, \sigma_t(x), \mu_t(x) + \beta_t \, \sigma_t(x)]$ for all $x \in D$
end for

(O3) In many relevant applications of safe BO, the input space is a continuous domain in moderately high dimensions. Since SafeOpt-type algorithms rely on a discrete input space, they cannot be used in these settings. It is therefore necessary to devise alternatives that work in moderately high dimensions, while retaining safety guarantees.

These three aspects will be explored in Sections 5, 6 and 7, respectively.

4 Related work

In the following, related work will be reviewed. We first provide an overview of safe BO methods, particularly those closely related to the present setting. As Lipschitz bounds play a central role from Section 6 onwards, we also review previous work on methods that rely on such an assumption.

4.1 Safety in Bayesian optimization

We first consider safety in the context of Bayesian optimization (BO) (Shahriari et al., 2015; Garnett, 2023). The type of safety constraint considered in this work was first introduced in the seminal paper (Sui et al., 2015), which also proposed and analyzed the original SafeOpt algorithm. A safety constraint of this type appears in many practically relevant application scenarios, including controller tuning (Berkenkamp et al., 2016b; 2023), automatic tuning of medical therapies (Sarikhani et al., 2021), or safe robot learning (Baumann et al., 2021). Several variations to the original algorithm have been proposed, with a SafeOpt-variant that does not use a Lipschitz constant (Berkenkamp et al., 2016b) proving particularly popular in the robotics community. While the original SafeOpt algorithm from Sui et al. (2015) interleaves safe exploration and function optimization over certified safe sets, a two-stage variant was introduced and analyzed in (Sui et al., 2018). Since SafeOpt relies on a discrete input space, and hence requires discretization in the case of a continuous optimization problem, applying this algorithm to continuous problems in even moderate dimensions can be very challenging, cf. also Section 7. This motivated the introduction of a variant based on swarm-optimization (Duivenvoorden et al., 2017), albeit with heuristic safety guarantees, as well as a variant tailored to dynamical problems in high dimensions (Sukhija et al., 2023). A method based on random projections was also proposed and analyzed in (Kirschner et al., 2019a), and applied to tuning of a particle accelerator (Kirschner et al., 2019b). In terms of problem formulations, instead of just one

safety constraint for the input function, multiple safety constraints can also be enforced, each encoded by an unknown constraint function (Berkenkamp et al., 2016a; Sui et al., 2018; Berkenkamp et al., 2023). In many applications of SafeOpt-type algorithms, properties of a dynamical system need to be optimized under safety constraints. For example, in controller tuning with BO, the inputs correspond to parameters of a controller, and the performance of the controller is the target function to be optimized. The dynamic aspect can be included in the optimization algorithm and its safety mechanisms, for example, in order to use a backup controller (Baumann et al., 2021). Furthermore, while formally SafeOpt-type algorithms are BO algorithms having input constraints, this is different from *constrained BO* as considered for example in (Hernandez et al., 2016). In the latter type of BO, the focus is on finding good inputs (i.e., corresponding to a high objective function value) that fulfill the (usually unknown) constraints, and violation of the constraints during the optimization process is not consideredproblematic (though of course it can be advantageous to avoid this). By contrast, safe BO aims to avoid constraint violations during the optimization process since these are considered problematic or even harmful. For an in-depth discussion of this difference, and further connections between the two problem settings, we refer to the excellent survey (Kim et al., 2020).

SafeOpt-type algorithms are motivated by problems where safety violations are considered to be very costly. In particular, *any* safety violations should be avoided with very high probability. For example, in controller tuning, a safety violation might correspond to damage to the controlled plant, while in robot policy tuning it could mean a literal robot crash, potentially resulting in its destruction. In a medical context, a safety violation corresponds to patient harm and therefore has to be avoided with very high probability. In short, SafeOpt-type algorithms are particularly interesting for *hard safety requirements*, i.e., avoiding any safety constraint violation (with high probability). This contrasts with a whole spectrum of related, but different safe BO (and reinforcement learning) settings. For example, in robot policy optimization, a robot crash might not lead to damage, but rather correspond to an unsuccessful experimental run. A small number of crashes then becomes acceptable, since it might allow more aggressive exploration, which in turn leads to better policies after optimization. SafeOpt-type algorithms, tailored to avoiding *any* safety violations, are thus not an optimal choice in this scenario, and a crash-aware BO algorithm would be preferable (Marco et al., 2021). Similarly, in the context of bandit algorithms (Slivkins et al., 2019; Lattimore & Szepesvári, 2020), instead of avoiding a bad option at all costs, one might instead want to be *conservative*, carefully exploring alternatives, starting from a default strategy. This setting is formalized in the context of *conservative bandits* (Wu et al., 2016), and again, is related to, but distinct from, the SafeOpt setting. Corresponding bandit formulations in the context of hard safety are also available (Amani et al., 2019), where the focus is on regret bounds under safety constraints. Finally, parallel to the situation of conservative bandits, cautious variants of BO (Fröhlich et al., 2021) can be considered. In these variants, safety violations are not avoided at all costs, with the aim instead being for exploration to proceed cautiously.

4.2 Lipschitz-based methods

In order to address safety-related issues uncovered and discussed in Sections 5.1 and 6.1, we will introduce algorithms based on a Lipschitz assumption. In particular, it will be assumed that the target function is Lipschitz-continuous with a known upper bound on the Lipschitz constant. While regularity properties like Lipschitz (and the more general Hölder) continuity play an important role in the theory of statistical learning, especially in nonparametric statistical estimation (Tsybakov, 2009), learning algorithms based on Lipschitz assumptions have received relatively little attention in the machine learning community. Relevant exceptions to this are Lipschitz bandits (Kleinberg et al., 2019) and the original SafeOpt algorithm from Sui et al. (2015). The situation is considerably different in the field of global optimization and the systems and control community, respectively.

In the former, Lipschitz continuity with a specific Lipschitz constant is a standard assumption, used in a variety of algorithms for (certified) global optimization, cf. (Hansen et al., 1992; Pintér, 1995), though usually a noise-free setting is assumed in this literature. Global optimization of Lipschitz functions has recently also received attention from the machine learning community (Malherbe & Vayatis, 2017). Similarly, a specified Lipschitz constant is also used in the context of Lipschitz interpolation (Beliakov, 2006). Furthermore, closely related to our approach taken in Section 6, a deterministic variant of SafeOpt has been considered in

Sergeyev et al. (2020). However, the latter reference only works with functions on a compact interval, and does not use any BO technique.

In the systems and control community, Lipschitz assumptions have long been used, especially in the context of systems identification, where they were explicitly introduced and popularized by Milanese & Novara (2004), though similar methods had been used before, e.g., (Cooper, 1995). A known bound on the Lipschitz constant and on the size of additive noise is commonly used to derive uncertainty bounds in the context of regression. This approach has been further popularized and extended to the case of Hölder continuous functions by Calliess (2014), widely known as *kinky inference* in the systems and control community. A central assumption in this context is knowledge of a *concrete, numerical* upper bound on the Lipschitz constant of the target function. This assumption has a clear geometric and practical interpretation, namely a bounded rate of change of the target quantity. As such, it is related to the well-established field of sensitivity analysis (Da Veiga et al., 2021), for example.

Approaches to estimate the Lipschitz constant of an unknown function have been proposed both in the context of global optimization (Strongin, 1973) and in Lipschitz-based regression methods, particularly in the context of systems identification (Milanese & Novara, 2004; Novara et al., 2013; Calliess et al., 2020), see (Huang et al., 2023) for an overview and very recent sample-complexity results. We would like to stress that these approaches are not suitable for the present setting of hard safety constraints, since the estimation of a Lipschitz constant bound requires queries to the target function, which in turn already need to be safe, see also the discussion in Section 6.1.

The developments in Sections 6 and 7 combine kernel-based methods (here GP regression) with a Lipschitz assumption to overcome the requirement of a known bound on the RKHS norm of the target function, see Section 6.1 for details. The problematic nature of an RKHS norm bound in the context of learning-based control has been recognized for some time (Lederer et al., 2019; Fiedler et al., 2022). In (Lederer et al., 2019), using probabilistic Lipschitz bounds together with a space discretization was suggested to derive GP uncertainty bounds. However, this approach relies on a probabilistic setting, and is therefore not suitable in the context of SafeOpt-type algorithms. The work (Fiedler et al., 2022) proposes the usage of geometric constraints as prior knowledge in the context of uncertainty sets for kernel-based regression, with Lipschitz constant bounds as a special case. The resulting kernel machines, which provide nominal predictions and smoothed uncertainty bounds adhering to the geometric constraints, are not necessary in our setting, though using more general geometric constraints than Lipschitz constant bounds might be a promising avenue in the context of SafeOpt-type algorithms. Finally, combining kernel methods with Lipschitz assumptions is a natural approach as there is a close connection between regularity properties of a kernel and Lipschitz continuity of functions in the RKHS generated by the kernel. For a thorough discussion of this point, we refer to (Fiedler, 2023).

5 Frequentist uncertainty bounds and practical safety issues in SafeOpt

We now investigate practical safety implications of commonly used heuristics in the frequentist uncertainty bounds in SafeOpt-type algorithms, addressing objective (O1). In Section 5.1, we discuss why these heuristics are problematic and demonstrate safety issues using numerical experiments. To overcome these issues, in Section 5.2 we propose using state-of-the-art frequentist uncertainty bounds in the actual algorithm.

5.1 Practical safety issues in SafeOpt

Safety in SafeOpt-type algorithms is ensured by restricting query inputs to safe sets, which are computed using frequentist uncertainty bounds, typically in the form (4) using (5). However, these bounds are often too conservative for algorithmic use, which has led to implementations adopting heuristic choices for β_t, for example, $\beta_t \equiv 2$ in (Berkenkamp et al., 2016b; Turchetta et al., 2016), $\beta_t \equiv 3$ in (Helwa et al., 2019; Baumann et al., 2021), or $\beta_t \equiv 4$ in (Sukhija et al., 2022). Using such heuristics instead of evaluating β_t **invalidates all safety guarantees**. In practice, choosing some β_t can be a useful heuristic , as demonstrated by the reported success of SafeOpt-type algorithms (Berkenkamp et al., 2016b; Baumann et al., 2021; Sukhija et al., 2023). However, it should be stressed that in the setting of SafeOpt as outlined in Section 3, the

learning algorithm has to fulfill a **hard safety constraint** – namely, that no unsafe inputs are queried by the algorithm (potentially only with high probability). In particular, no burn-in or tuning phase for β_t is allowed. Such heuristics not only invalidate the theoretical safety guarantees, but can also actually lead to safety violations.

First, we demonstrate empirically that a simple heuristic like setting $\beta_t \equiv 2$ can lead to a significant proportion of bound violations. To do so, we follow the general approach adopted in (Fiedler et al., 2021a). We randomly generate 100 RKHS functions on $[0, 1]$ with RKHS norm 10, using the squared exponential kernel with length scale $0.2/\sqrt{2}$. Here, we utlize the orthonormal basis (ONB) of the corresponding RKHS as described in (Steinwart & Christmann, 2008, Section 4.4), by selecting some of these basis functions, and combining them in a weighted sum using randomly generated weights. For each of the resulting functions, we generate 10000 independent data sets, by uniformly sampling 100 inputs from $[0, 1]$, evaluating the RKHS function on these inputs, and then adding i.i.d. normal noise with variance 0.01. For each data set, we apply GP regression with a zero mean prior, and the SE kernel as covariance function, using the same length scale as for the generation of the target functions. Finally, we use the uncertainty set (4) with $\beta_t \equiv 2$, and check on a fine grid on $[0, 1]$ whether the target function from the RKHS is completely contained within this uncertainty set. 2727±3882 (average ± SD) of these runs (all 10000 repetitions for all 100 functions) led to a bound violation, a sizeable proportion.

Second, we show that these bound violations can indeed lead to safety violations when running SafeOpt. We generate an RKHS function f (same approach as above), and define the safety threshold h by setting $h = \hat{\mu}(f) - 0.2\hat{SD}(f)$, where $\hat{\mu}(f)$ and $\hat{SD}(f)$ are the empirical mean and standard deviation of the test function f evaluated on a fine grid of the input space. We further evaluate $|f'|$ on a fine grid, take the maximum and multiply it by 1.1 to find a (slightly conservative) upper bound of the Lipschitz constant of f. We then run SafeOpt on this function 10000 times from a random safe initial state, again using i.i.d. additive normal noise with variance 0.01. This leads to 2862 (out of 10000) runs with safety violations (see Table 1), which is unacceptable for most application scenarios of SafeOpt-type algorithms.

These experiments illustrate that using heuristics in SafeOpt can be highly problematic. Even in the relative benign setting used above, both uncertainty bound and safety violations occur. Moreover, in application scenarios for SafeOpt-type algorithms, tuning the heuristic scaling factor is not possible, as it is the primary mechanism for safety. Dispensing with the need for such heuristics, and retaining safety guarantees both in theory and practice, is therefore the primary motivation for this work.

Remark 1. *The assumption $f \in H_k$ is standard in the BO literature (Abbasi-Yadkori, 2013; Srinivas et al., 2010; Chowdhury & Gopalan, 2017; Whitehouse et al., 2024), where k is a kernel that is used as the covariance function in GP regression. In general, if functions from H_k are to be used as models of the underlying target function, the RKHS H_k has to be sufficiently rich. In many cases, as a qualitative assumption, this is quite mild. For example, if D is a compact metric space, then a large variety of* universal kernels *are available – kernels with an RKHS that is dense (w.r.t. the supremum norm) in the set of all continuous functions on D (Steinwart & Christmann, 2008, Section 4.6). However, the assumption $f \in H_k$ is considerably more stringent and delicate. Specifically, even containment of the target function within an RKHS generated by a kernel from the same class as the one used as a covariance (e.g., a squared exponential or Matern kernel) may not be sufficient, because there can be a mismatch of hyperparameters. For the case of squared exponential kernels, a complete characterization of the inclusions of RKHSs w.r.t. different hyperparameters is described in (Steinwart & Christmann, 2008, Section 4.4), revealing that this situation can indeed occur. While there is work on covariance function misspecification in GP regression (Wynne et al., 2021)*[4] *and GP-based BO (Berkenkamp et al., 2019; Bogunovic & Krause, 2021), these works do not help in the present situation, because of the form of the resulting guarantees in the former case, and the potential for schemes sampling unsafe inputs in the latter case. The situation is further complicated by the fact that in practice the hyperparameters are adapted during BO (Garnett, 2023), theoretical guarantees of which are only slowly emerging (Teckentrup, 2020; Karvonen et al., 2020; Karvonen & Oates, 2023). In the following, we work (unless noted otherwise) with the assumption $f \in H_k$, and do not consider hyperparameter adaption during the optimization process. This is justified from two perspectives. First, it is a common approach in the BO literature, including safe BO (Sui et al., 2015; 2018). Second, and more importantly, the safety-*

[4]Even likelihood misspecification.

related issues that we are about to discuss are independent of this issue (though of course, it is an additional problematic factor).

5.2 Real-β-SafeOpt

As a first step, we propose using modern uncertainty bounds in SafeOpt that can be computed numerically, avoiding the replacement with unreliable heuristics. For this purpose, we investigate the original SafeOpt algorithm with β_t from (7), which is very close to (6). To clearly distinguish this variant of SafeOpt from previous work, we call it *Real-β-SafeOpt*, emphasizing that we use a *theoretically sound choice of* β_t. The resulting algorithm is again described by Algorithm 2, using (7) to compute β_t. The bound (7) requires the determinant to be computed, which can be computationally expensive, but typical applications of SafeOpt and related algorithms allow few evaluations, so this does not pose a problem. Furthermore, the additive noise needs to be a conditionally R-subgaussian (martingale-difference) sequence with a (known upper bound on) R. This assumption is standard, and in many cases harmless. It also has a clear interpretation. Finally, for a frequentist uncertainty bound, we also need the next assumption.

Assumption 2. *Some $B \in \mathbb{R}_{\geq 0}$ is known with $\|f\|_k \leq B$.*

While Assumption 2 is standard in the literature on safe BO, to the best of our knowledge, it has only been used for theoretical analyses, but not in the actual algorithms.

Combining these ingredients allows us to compute the bound (7), so Real-β-SafeOpt can actually be implemented. To illustrate the advantages of Real-β-SafeOpt, we run it on the same function f as before (cf. Section 5.1), and set $\delta = 0.01$, as well as the true RKHS norm in (7). Running this experiment results in 0 failures (cf. Table 1), evidently well within the δ range required.

An obvious question arising at this point is how Real-β-SafeOpt compares in terms of performance with previous SafeOpt-variants relying on heuristics. Unfortunately, a meaningful comparison is impossible since the two algorithmic variants address different problems. If for the latter the heuristic constant β is determined by trial-and-error, and SafeOpt is then run with this constant that leads to no or almost no safety violation, then we essentially end up with a different algorithm overall. If the heuristic constant β is chosen arbitrarily or based on experience with previous usages of SafeOpt-type algorithms, then the original setting of hard safety constraints is abandoned, and we instead end up with a form of cautious BO. By contrast, Real-β-SafeOpt tries to stay within the original setting of SafeOpt that requires hard safety constraints. To investigate how Real-β-SafeOpt typically behaves, we will perform a careful empirical evaluation of this algorithm variant in Section 6.3.

6 Lipschitz-only safe Bayesian optimization (LoSBO)

As discussed above, the safety of SafeOpt-type algorithms should only depend on reasonable assumptions that can be verified and interpreted by the algorithm's user. In this section, we address objective (O2). In particular, we investigate the central Assumption 2 of a known upper bound on the RKHS norm of the target function, finding that this is a problematic assumption in practice. To overcome this issue, we propose Lipschitz-only Safe Bayesian Optimization (LoSBO) as a solution to this problem, and perform extensive numerical experiments, comparing LoSBO with Real-β-SafeOpt.

6.1 Practical problems with the RKHS norm bound

A central ingredient in the Real-β-SafeOpt algorithm is an upper bound on the RKHS norm of the target function. In particular, the safety and exploration guarantees inherited from (Sui et al., 2015) hinge on the knowledge of such an upper bound. Unfortunately, while the RKHS norm is very well understood from a theoretical point of view, it is unclear how to derive such a bound in practice. In the following, we provide an overview of known characterizations and representations of the RKHS norm, and discuss why these results seem unsuitable for this task.

For an arbitrary kernel, one can use discretization-based variational characterizations of the RKHS norm (and RKHS functions), for example, by maximization over a family of lower bounds on the RKHS norm

(Fiedler et al., 2024, Section B), (Atteia, 1992, Chapter I), by minimization over certain bounds on function values at finitely many inputs (Okutmuştur, 2005, Theorem A.2.6), by minimization over finite interpolation problems (Paulsen & Raghupathi, 2016, Theorem 3.11), or by minimization over certain matrix inequalities (Paulsen & Raghupathi, 2016, Theorem 3.11). For separable RKHSs, the RKHS norm can be expressed using a sampling expansion (Korezlioglu, 1968), or as the limit of norms of RKHSs over finite inputs (Lukić & Beder, 2001, Lemma 4.6). On the one hand, all of these variational problems have an explicit form and they work for *any* kernel (any kernel with separable RKHS, respectively). However, it is not at all clear how to relate these representations to common properties of functions that might be used as reliable prior knowledge to derive upper bounds on the RKHS norm. Furthermore, these variational problems generally cannot be used in numerical methods to estimate upper bounds on the RKHS norm, but only lower bounds, though they may be used for estimating bounds in heuristics (Tokmak et al., 2023). Since these characterizations are based on discretizations of a given RKHS function, in particular, using the exact function values, they are not suitable in the present setting of unknown target functions only accessible through noisy evaluations.

If one considers more specific classes of kernels, other characterizations of the RKHS norm become available. For example, continuous kernels on a compact metric space equipped with a measure having full support (often called a Mercer kernel in this context) allow a description of the RKHS norm as a weighted ℓ_2-norm (Steinwart & Christmann, 2008, Section 4.5), based on Mercer's theorem. This has a clear interpretation in the context of kernel methods, in particular, giving insight into the regularization behavior of the RKHS norm in optimization problems in kernel machines (Hastie et al., 2009, Section 5.8), which in turn can be used to derive learning rates for various statistical learning problems (Steinwart et al., 2009). More general forms of Mercer's theorem are available (Steinwart & Scovel, 2012), which in turn lead to improved learning theory results (Fischer & Steinwart, 2020). While the RKHS norm representation for Mercer kernels is an important tool for statistical learning theory and provides intuition about the regularization behavior, it is again unclear how it can be used to derive *quantitative* RKHS norm bounds. Expressing the RKHS norm for Mercer kernels as a weighted ℓ_2-norm provides valuable *qualitative* intuition about the corresponding RKHS norm, but we are not aware of any practically relevant example where this has been used to translate realistic prior knowledge into a concrete upper bound on the RKHS norm.

Similarly, for sufficiently regular translation-invariant kernels, the RKHS norm can be expressed as a weighted integral over the Fourier transform of RKHS functions (Wendland, 2004, Theorem 10.12). This formulation allows an intuitive interpretation of the RKHS norm as a generalized energy, penalizing high-frequency behavior of RKHS functions (as determined by the Fourier transforms of the kernel). Several important function spaces are related to RKHSs, for example certain Sobolev spaces (Wendland, 2004, Chapter 10) or Fock spaces (Steinwart & Christmann, 2008, Section 4.4), which again have their own representations of the RKHS norm (potentially after some embedding). Again, all of these representations offer insights into the RKHS norm, and are important theoretical tools, but how this can be used to derive practically useful quantitative upper bounds on the RKHS norm remains unclear.

To summarize, while an extensive body of work on characterization and representation results for the RKHS norm is available, these results appear to be unsuitable to derive upper bounds on this norm. In particular, to the best of our knowledge, it is not currently possible to derive concrete numerical upper bounds on the RKHS norm from realistic assumptions in non-trivial cases. The central difficulty here is that a concrete *numerical upper bound* on the RKHS norm is needed for algorithms like Real-β-SafeOpt. This issue is known in e.g. the learning-based control community (Fiedler et al., 2022), but has not yet been addressed in the context of safe BO. A possible explanation for this current lack is that this problem does not appear in many other kernel-based learning scenarios. For example, to derive learning rates for SVMs and related kernel machines, membership of the target function[5] in an appropriate RKHS is enough (or in a function space that in a suitable sense can be approximated by an RKHS), and (an upper bound of) the RKHS norm of the target function is *not* needed by the kernel-based learning algorithm.

The discussion above has immediate consequences for SafeOpt-type algorithms. These algorithms are intended for hard safety settings, i.e., scenarios where *any* safety violation is very costly and must be avoided.

[5]In the standard statistical learning theory setup, there is no notion of a target function as in the context of safe BO. Instead, the learning problem is described using a loss function. However, for example in the context of regression with the squared-error loss, the conditional expectation (the regression function) takes the role of our target function in the explanation above.

In these scenarios it is not possible to have a tuning phase before the actual optimization run where algorithmic parameters are set - the safety requirements hold from the start. For SafeOpt-type algorithms this means that all algorithmic parameters have to be set beforehand, and these parameters need to ensure the safety guarantees and lead to satisfying exploration behaviour. However, this entails an upper bound on the RKHS norm of the target function, which currently appears to be impossible to derive from reasonable prior knowledge in practically relevant scenarios.

Furthermore, using an invalid RKHS upper norm bound can indeed easily lead to safety violations. In order to illustrate this, we run SafeOpt on the same function f as before (cf. Section 5.1), and set $\delta = 0.01$, but this time, we use a misspecified RKHS norm of 2.5 in (7). Running this experiment now results in 1338 failure runs out of 10000, cf. Table 1, which is much more than what would be expected from a safety probability of $1 - \delta = 0.99$. For a summary of this experiment, see again Table 1.

Finally, simply using a very conservative upper bound on the RKHS norm is not a viable strategy to overcome this problem. A severe overestimation of the RKHS norm leads to very large and overly conservative uncertainty bounds, which in turn leads to performance issues. In particular, since the uncertainty bounds are used to determine the safety sets in SafeOpt-type algorithms, a supposed RKHS upper norm bound that is too conservative can result in the algorithm "getting stuck", i.e., no more exploration is possible. It should also be noted that it is not even clear what "very conservative" means in the present context. Recalling the discussion of the RKHS norm from above, while an extensive body of theory and strong qualitative intuitions on these objects are available, the lack of concrete, quantitative bounds amenable to numerical evaluation makes it very difficult for users of BO to judge what a conservative estimate of the RKHS norm in a given situation could be.

We argue that as a consequence, any SafeOpt-like algorithm *should not depend on the knowledge of an upper bound on the RKHS norm of the target function for safety*, at least in the setting of hard safety requirements where *any* failure should be avoided (with high probability). More generally, in order to guarantee safety, we should only use assumptions that are judged as reliable and reasonable by practitioners. In particular, all assumptions that are used for safety should have a clear interpretation for practitioners and a clear connection to established prior knowledge in the application area of the safe BO algorithm. In the end, it is up to the user of safe BO to decide which assumptions used in the safety mechanism can be considered as reliable.

6.2 Describing LoSBO and its safety guarantees

Motivated by the popularity of SafeOpt, which combines GPs with a Lipschitz assumption, and the extensive experience of the systems and control community with Lipschitz bounds and bounded noise (cf. Section 4), we propose to use *an upper bound on the Lipschitz constant of the target function and a known noise bound* as the ingredients for safety in BO. More precisely, we propose Lipschitz continuity of the target function and a known upper bound on the Lipschitz constant, together with bounded measurement noise and a known upper bound on the magnitude of the noise, as core assumptions for safety. Both of these assumptions fulfill the desiderata outlined above. First, they have a clear interpretation: A known upper bound on the Lipschitz constant corresponds to a slope constraint on the target function, and bounded noise is self-explanatory. Second, they are easily related to established prior knowledge: A known upper bound on the Lipschitz constant corresponds to an a priori bound on the rate of change of a function, i.e., it is related to the sensitivity of the underlying problem, and a known bound on the magnitude of the noise means that the strength of the noise generating mechanism is known.

The key idea is to ensure the safety mechanism works reliably independent of the (statistical) exploration mechanism. In a generic SafeOpt algorithm, safety is guaranteed by ensuring that the safe sets S_t contain only safe inputs (potentially only up to high probability), i.e., requiring that $f(x) \geq h$ for all $x \in S_t$. Once this property is fulfilled, the rest of the algorithm can no longer violate the safety constraints anymore. Based on the earlier discussion, the construction of the safe set should only rely on the Lipschitz and noise bound. As is well-known, these two assumption allow the construction of lower bounds on the function (Milanese & Novara, 2004), and the corresponding safe sets should therefore be defined for all $t \geq 1$ as

$$S_t = S_{t-1} \cup \{x \in D \mid y_{t-1} - E - Ld(x_{t-1}, x) \geq h\}, \tag{11}$$

where $L \in \mathbb{R}_{\geq 0}$ is a bound on the Lipschitz constant of the unknown target function, $E \in \mathbb{R}_{\geq 0}$ a bound on the magnitude of the noise, and S_0 is the initial safe set. We propose using this variant of the safe set, and leaving the rest of the generic SafeOpt algorithm unchanged, which leads to an algorithm we call Lipschitz-only Safe Bayesian Optimization (LoSBO). The algorithm fulfills the following safety guarantee.

Proposition 1. *Let $f : D \to \mathbb{R}$ be an L-Lipschitz function. Assume that $|\epsilon_t| \leq E$ for all $t \geq 1$ and let $\emptyset \neq S_0 \subseteq D$ such that $f(x) \geq h$ for all $x \in S_0$. For any choice of the scaling factors $\beta_t > 0$, running the LoSBO algorithm leads to a sequence of only safe inputs, i.e., we have $f(x_t) \geq h$ for all $t \geq 1$.*

Proof. It is enough to show that $\forall t \geq 0$ and $x \in S_t$, we have $f(x) \geq h$. Induction on t: For $t = 0$, this follows by assumption. For $t \geq 1$, let $x \in S_t = S_{t-1} \cup \{x \in D \mid y_{t-1} - E - Ld(x_{t-1}, x) \geq h\}$. If $x \in S_{t-1}$, then $f(x) \geq h$ follows from the induction hypothesis. Otherwise we have

$$f(x) = f(x_t) + \epsilon_t - \epsilon_t + f(x) - f(x_t) \geq y_t - E - Ld(x_t, x) \geq h,$$

where we used the L-Lipschitz continuity of f and the noise bound $|\epsilon_t| \leq E$ in the first inequality, and the definition of S_t in the second inequality. □

The argument in the proof above is well-known in e.g. the systems identification literature, and the result bounds even fulfill certain optimality properties (Milanese & Novara, 2004). We would like to stress that the safety guarantee of LoSBO, as formalized in Proposition 1, is *deterministic*, i.e., it always holds and not only with high probability. This type of safety is often preferred in the context of control and robotics (Hewing et al., 2020; Brunke et al., 2022).

Remark 2. *Inspecting the proof of the preceding result reveals that considerably more general (and weaker) assumptions can be used with the same argument.*

1. *Instead of a fixed noise bound E, one can mutatis mutandis use asymmetric, time-varying and heteroscedastic noise. Formally, one can assume that two functions $E_\ell, E_u : D \times \mathbb{N}_0 \to \mathbb{R}_{\geq 0}$ exist, such that for all $t \geq 1$ and $x \in D$, it holds that $E_\ell(x, t) \leq \epsilon_t \leq E_u(x, t)$, if x is the input used at time t.*

2. *Instead of Lipschitz continuity, one can assume that there exists a continuous and strictly increasing function $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $\phi(0) = 0$, such that for all $x, x' \in D$ it holds that $f(x') \geq f(x) - \phi(d_D(x, x'))$. This includes the case of Hölder continuity, which has previously been used in a similar context (Calliess, 2014).*

To keep the presentation focused and easy to follow, we do not use this additional flexibility in the present work, but everything that follows immediately applies to these more general cases.

Our proposed modification applies to *any* algorithm instantiating the generic SafeOpt strategy outlined in Section 3. For concreteness, we focus in the following on the original SafeOpt algorithm from (Sui et al., 2015). The resulting variant of LoSBO is described in detail in Algorithm 3.

While LoSBO arises from a rather minor modification of the generic SafeOpt algorithm class (by changing the computation of the safe sets S_t), on a conceptual level significant differences arise. Inspecting the proof of Proposition 1 shows that the safety guarantee of LoSBO is *independent* of the underlying model sequence $(\mathcal{M}_t)_t$. As an important consequence, the choice of the uncertainty sets used in the optimization of the acquisition function cannot jeopardize safety. One consequence is that the assumption that the target function f is contained in the RKHS of the covariance function used in GP regression is not necessary anymore. In particular, *in order to ensure safety, we need only Assumption 1* together with a noise bound, *and not Assumption 2 anymore*. Similarly, hyperparameter tuning is not an issue for safety, cf. also to our discussion in Section 2.3. Of course, an appropriate function model is important for good exploration performance, but this issue is now independent of the safety aspect. As another, even more important consequence, in the context of the concrete LoSBO variant described in Algorithm 3, the scaling parameter β_t is now a proper tuning parameter. Modifying them, even online, in order to improve exploration no longer interferes with the safety requirements. This differs from previous variants (and practical usage) of SafeOpt,

Algorithm 3 LoSBO

Require: Lipschitz constant L, algorithm to compute β_t, noise bound E, initial safe set S_0, safety threshold h

$Q_0(x) \leftarrow \mathbb{R}$ for all $x \in D$ ▷ Initialization of uncertainty sets
$C_0(x) \leftarrow [h, \infty)$ for all $x \in S_0$
$C_0(x) \leftarrow \mathbb{R}$ for all $x \in D \setminus S_0$
for $t = 1, 2, \ldots$ **do**
  $C_t(x) \leftarrow C_{t-1}(x) \cap Q_{t-1}(x)$ for all $x \in D$ ▷ Compute upper and lower bounds for current iteration
  $\ell_t(x) \leftarrow \min C_t(x)$, $u_t(x) \leftarrow \max C_t(x)$ for all $x \in D$
  if $t > 1$ **then** ▷ Compute new safe set
    $S_t = S_{t-1} \cup \{x \in D \mid y_{t-1} - E - Ld(x_{t-1}, x) \geq h\}$
  else
    $S_1 = S_0$
  end if
  $G_t \leftarrow \{x \in S_t \mid \exists x' \in D \setminus S_t : \; u_t(x) - Ld(x, x') \geq h\}$ ▷ Compute set of potential expanders
  $M_t = \{x \in S_t \mid u_t(x) \geq \max_{x_S \in S_t} \ell_t(x_S)\}$ ▷ Compute set of potential maximizers
  $x_t \leftarrow \operatorname{argmax}_{x \in G_t \cup M_t} w_t(x)$ ▷ Determine next input
  Query function with x_t, receive $y_t = f(x_t) + \epsilon_t$
  Update GP with new data point (x_t, y_t), resulting in mean μ_t and σ_t
  Compute updated β_t
  $Q_t(x) = [\mu_t(x) - \beta_t \, \sigma_t(x), \mu_t(x) + \beta_t \, \sigma_t(x)]$ for all $x \in D$
end for

where the scaling factors β_t *cannot* be freely tuned since they are central for the safety mechanism of these algorithms. This aspect is illustrated in Figure 3. In the situation depicted there, the uncertainty bounds do not hold uniformly, i.e., the target function is not completely covered by them, and deriving safety sets from these uncertainty bounds, regardless of whether to include the additional knowledge of the Lipschitz bound (Sui et al., 2015) or not (Berkenkamp et al., 2016a), results in potential safety violations. However, since in LoSBO these bounds are ignored for the safe sets, this problem does not occur.

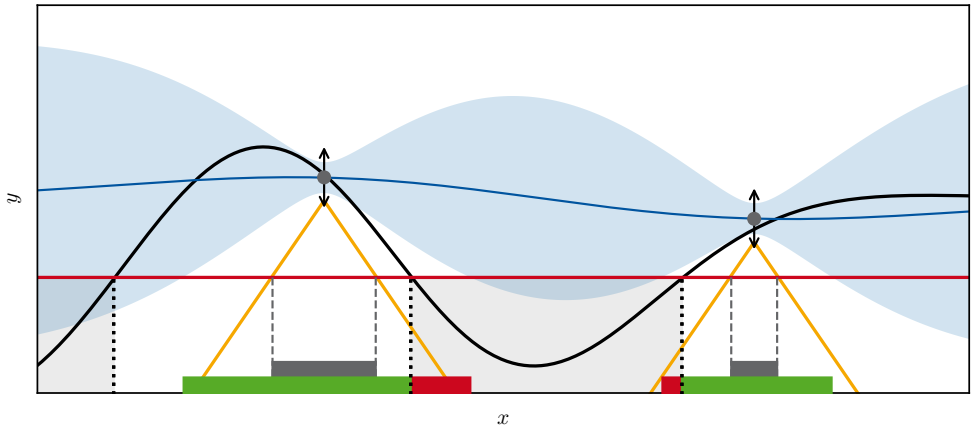

Figure 3: Illustration of LosBO being safe, while a safe set based on invalid uncertainty bounds leads to potential safety violations. The safe set of LoSBO (gray set) is determined by the constant E (gray arrow) and the Lipschitz cone (orange). The GP mean and the confidence bounds are illustrated in blue. The points in the safe set given by the lower confidence bound are green if they are safe and red if they are unsafe.

The original SafeOpt algorithm comes with conditional[6] exploration guarantees. Since our modification leading to LoSBO essentially separates the safety and exploration mechanisms, the exploration guarantees from SafeOpt are rendered inapplicable in the present context. An inspection of the proof of (Sui et al., 2015, Theorem 1) shows that it cannot easily be modified to apply to LoSBO again, as the argument used there relies on the GP model interacting with the safety mechanism[7]. Furthermore, we suspect that pathological situations exist where LoSBO fails to properly explore. However, we have not observed such a situation in our extensive experiments. While providing (again conditional) exploration guarantees for LoSBO is an interesting focus for future work, we argue that the present lack of such theoretical guarantees does not diminish the relevance and usefulness of this algorithm. First, LoSBO shows excellent exploration performance, as demonstrated in the experiments described in the next section. Second, since the scaling parameters β_t (which have an important influence on the exploration performance) are proper tuning parameters in LoSBO, unsatisfying performance of the algorithm can be overcome by using this tuning knob. We would like to stress again that in the previous variants of SafeOpt, this option is not available as the scaling parameters need to lead to valid uncertainty sets.

Remark 3. *Proposition 1 states that if $E \in \mathbb{R}_{>0}$ is a bound on the noise magnitude, then LoSBO is safe. Suppose we know that $|\epsilon_t| \leq B_\epsilon$ for some constant $B_\epsilon \in \mathbb{R}_{>0}$, then we could set $E = B_\epsilon$, and assume that the bound B_ϵ is sharp. For example, we might have $\epsilon_t \sim \frac{1}{2}\delta_{B_\epsilon} + \frac{1}{2}\delta_{-B_\epsilon}$, where δ_x is the Dirac distribution with atom on x. If we choose $E = B_\epsilon$, then LoSBO is indeed safe according to Proposition 1, i.e., for all inputs $x_t \in D$ that are queried by the algorithm, we have $f(x_t) \geq h$. However, if we additionally assume that there are sizeable parts of the input space D where $f \approx h$, and since the border of the safe sets will be in such a region, it is likely that inputs from this area will be queried. This means that it is likely that measurements y_t with $y_t < h$ will be received - this happens if $f(x_t) \approx h$ and $\epsilon_t \approx -B_\epsilon$. While according to the formal model such an input x_t is safe, to the user it looks as if a safety violation occurred. Since in practice a detected (not necessarily real) safety violation might lead to some costly action (e.g., emergency stop), such a situation is undesirable. To avoid this, we can set $E = 2B_\epsilon$. While this introduces some conservatism, it avoids the described apparent safety violations - essentially, this option mitigates false alarms. Whether to choose in the present situation $E = B_\epsilon$ or $E = 2B_\epsilon$ (or even an option in between) is ultimately a practical question to be addressed by the practitioner using the algorithm.*

6.3 Experimental evaluation

As discussed in Section 5.2, a meaningful comparison with SafeOpt implementations relying on heuristics is impossible, since the latter essentially address a different problem setting. For this reason, we will compare LoSBO with Real-β-SafeOpt, which precisely adheres to the original SafeOpt setting.

Experimental setup For the empirical evaluations a frequentist approach will be used as this is the most natural setting for SafeOpt-like algorithms, see (Srinivas et al., 2010; Fiedler et al., 2021a;b) for further discussion on this. A target function is therefore fixed, and the algorithms are run multiple times on this same function with independent noise realizations. To enable the performance of the algorithms to be clearly evaluated, synthetic target functions will be used, and since we want to compare LoSBO with Real-β-SafeOpt - the latter requiring a target function from an RKHS and with a known RKHS norm upper bound - we generate target functions from an RKHS. The frequentist setup is inherently worst-case, but for numerical experiments it is necessary to restrict ourselves to finitely many RKHS functions. Nevertheless, the RKHS functions used should be somewhat representative to give a meaningful indication of the algorithmic performance. Specifically, any bias due to the function generating method should be minimized. In the following experiments, we sample functions from the pre RKHS, i.e., given a kernel k, we randomly choose some $M \in \mathbb{N}_{>0}$, $\alpha_1, \ldots, \alpha_M \in \mathbb{R}$ and $x_1, \ldots, x_M \in D$ and then use $f = \sum_{i=1}^{M} \alpha_i k(\cdot, x_i) \in H_k$ as a target function, which works for *any* kernel, cf. Section 2.2. In the case of the squared exponential kernel, we also utilize the ONB described in Steinwart & Christmann (2008, Section 4.4), which we have already used for

[6]By this we mean that the exploration guarantee given in (Sui et al., 2015, Theorem 1) is conditional on the choice of the kernel. Inspecting the expression for the time t^* in this latter result, we find that this result requires appropriate growth behavior of the maximum information gain γ_t in order to lead to non-vacuous exploration guarantees.

[7]More precisely, with LoSBO we cannot ensure that the inequality in the last display in the proof of (Sui et al., 2015, Lemma 7) holds.

some of the experiments in Sections 5.1 and 6.1. Generating RKHS functions with more than one method ensures more variety of the considered RKHS functions. Moreover, with both approaches, the exact RKHS norm is available (and can be set by normalization), and the generated functions can be evaluated at arbitrary inputs. Unless noted otherwise, we generate RKHS functions with an RKHS norm of 10, i.e., we consider target functions $f \in H_k$ with $\|f\|_k = 10$. For a more thorough discussion of generating RKHS functions and subtle biases due to the chosen method, we refer to (Fiedler et al., 2021a).

LoSBO and Real-β-SafeOpt work on arbitrary metric spaces, as long as a kernel can be defined on that space. Following the previous safe BO literature, we restrict ourselves to compact subsets of $\mathbb{R}^d$, and in this section for simplicity we further restrict ourselves to $d = 1$. To run LoSBO and Real-β-SafeOpt, we need a bound on the Lipschitz constant of the target function, as well as an initial safe set. For the former, we restrict ourselves to kernels inducing continuously differentiable RKHS functions, since the latter are Lipschitz continuous due to the compact domain. To determine a bound on the Lipschitz constant, we evaluate the target function on a fine discretization of the input domain, numerically compute an approximation of the Lipschitz constant, and multiply the result by 1.1 to counterbalance the discretization error. Since the target functions are randomly generated, we compute an appropriate safety threshold for each function so that some portions of the input space are safe, and some are unsafe. This avoids trivial situations for safe BO. More precisely, for a given target function f, we compute its empirical mean $\hat{\mu}(f)$ and empirical standard deviation $\hat{SD}(f)$ on a fine grid, and then set $h = \hat{\mu}(f) - 0.2\hat{SD}(f)$. Next, for each target function f and safety threshold h, we need to generate an initial safe set. Similar to the choice of the safety threshold, trivial situations should be avoided, particularly cases where no safe exploration is possible. To achieve this goal, we first determine some $x_0 \in \operatorname{argmax}_{x \in D} f(x)$, then consider the set $D \cap I_{x_0}$, where I_{x_0} is the largest interval such that $x_0 \in I_{x_0}$ and $f|_{I_{x_0}} \geq h + E$. Finally, one input is randomly selected from this set, and the singleton set containing this latter input is then selected as the initial safe set. Using a singleton initial safe set is common in the literature on SafeOpt-type algorithms, see (Berkenkamp et al., 2016a).

The typical application scenario for SafeOpt-type algorithms is the optimization of some performance measure by interacting with a physical system. In particular, each function query is relatively expensive, hence in these scenarios only a few function values are sampled. Motivated by this, in all of the following experiments, for each target function, LoSBO and Real-β-SafeOpt are run for 20 iterations, starting from the same safe set. For each target function, this is repeated 10000 times to allow a frequentist evaluation of the behavior. Finally, each type of experiment is run with 100 different randomly generated target functions. To make runs with different target functions comparable, we evaluate the performance in a given run of a target function f by

$$\hat{f}_t^* = \frac{f\left(\operatorname{argmax}_{x \in S_t} \mu_t(x)\right) - h}{f^* - h}, \tag{12}$$

where $\mu_t(x)$ is the predictive mean (in LoSBO and Real-β-SafeOpt the posterior mean) at time $t \geq 1$, evaluated at input x, and f^* is the maximum of f. This metric will be averaged over all runs for a given f, and over all target functions, respectively, in the following plots.

For simplicity, independent additive noise, uniformly sampled from $[-B_\epsilon, B_\epsilon]$, is used in all of the following experiments. As is well-known, bounded random variables are subgaussian, and we can set $R = B_\epsilon$ in Real-β-SafeOpt. Additionally, we choose $\delta = 0.01$ and the true RKHS norm as the RKHS norm upper bound in Real-β-SafeOpt, unless noted otherwise. We further set the nominal noise variance equal to R in both LoSBO and Real-β-SafeOpt. Following the discussion in Remark 3, we choose $E = 2B_\epsilon$ in LoSBO. Finally, we must specify a strategy to compute β_t in LoSBO. Recall from Section 6.2 that these scaling factors are now proper tuning parameters. In all of the following experiments, we use $\beta \equiv 2$ in LoSBO, as this is a common choice in the literature on SafeOpt and GP-UCB. Choosing such a simple rule also simplifies the experimental evaluation, as no additional tuning parameters or further algorithmic choices are introduced. Unless noted otherwise, in all of the following experiments $B_\epsilon = 0.01$ is used. For convenience, the experimental results are concisely summarized in Table 1.

Well-specified setting We start by comparing LoSBO and Real-β-SafeOpt in a well-specified setting. This means that all the algorithmic parameters are set correctly, in particular, the covariance function used in GP regression is the kernel generating the RKHS from which the target functions are sampled. In Figure

Algorithm	SafeOpt	Real-β-SafeOpt	Real-β-SafeOpt	Real-β-SafeOpt	LosBO
Experimental setting	$\beta = 2$ (heuristic)	$B = 2.5$ (underestimate)	$B = 10$ (correct)	$B = 20$ (overestimate)	$\beta = 2$
Not started %	1.93	3.16	30.40	68.34	0.018
Safety violations %	3.95	0.859	0	0	0
Safety violations worst case %	28.62	13.38	0	0	0
Final performance %	88.75	88.76	82.45	76.69	90.90

Table 1: Safety-performance tradeoff in SafeOpt. We evaluated 100 functions sampled from an SE-kernel with $B = 10$. On each function we ran each algorithm 10000 times, starting from two initial safe points.

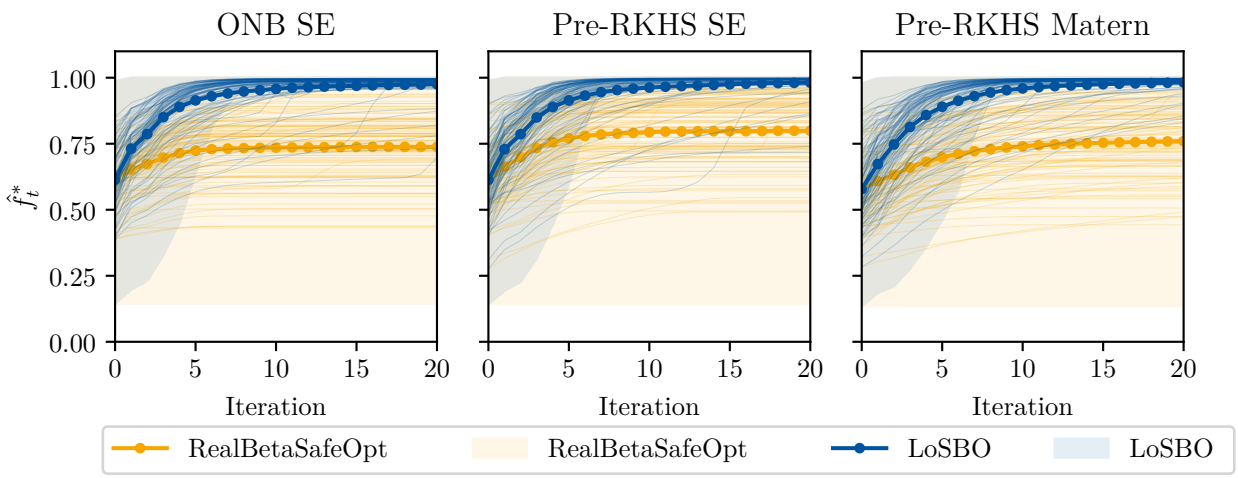

Figure 4: Comparison of LosBO and Real-β-SafeOpt in a well-specified setting. Thick solid lines are the means over all functions and repetitions, thin solid lines are the means over all repetitions for each individual function, shaded area corresponds to one standard deviation over all runs.

4, the results for this setting are presented. The thick solid lines are the means over all 100 functions and all of their repetitions, the fine lines are the means for each of the individual 100 target functions (over 10000 repetitions for each function), and the shaded area shows an interval of width one standard deviation around the mean, again over all functions. Figure 4, top left, displays the results for functions from the Squared Exponential RKHS, sampled using the ONB approach. Interestingly, LoSBO exhibits superior performance compared to Real-β-SafeOpt, despite providing the latter algorithm with the correct ingredients (Lipschitz bound, kernel, RKHS norm bound, noise variance). Figure 4, top right, displays the results for functions sampled from the Squared Exponential pre RKHS. While no real difference in performance is noticable for LoSBO, Real-β-SafeOpt appears to perform slightly better compared to target functions generated using the ONB approach. A potential explanation lies in the shapes of functions that typically arise in the two different sampling methods. As observed in (Fiedler et al., 2021a), functions sampled from the ONB look more "bumpy" compared to pre RKHS functions, and appear to be more challenging for the uncertainty bounds. Since Real-β-SafeOpt needs to precisely adhere to these bounds, its exploration performance is diminished. By contrast, LoSBO behaves overly optimistically as $\beta_t \equiv 2$ is used, but the underlying RKHS functions have RKHS norm 10, see also the evaluations in (Fiedler et al., 2021a). It appears that this over-optimism leads to better performance, and since for safety LoSBO *does not* rely on scaling factors β_t that correspond to valid frequentist uncertainty sets, this over-optimism does not jeopardize safety. Finally, Figure 4, lower left, shows the results for RKHS functions corresponding to a Matern-3/2 kernel and sampled

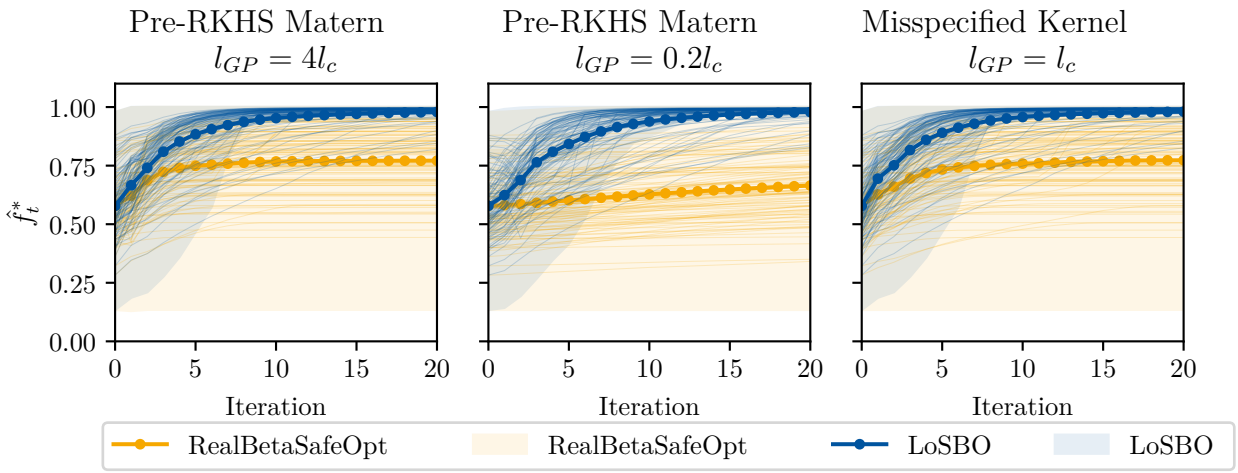

Figure 5: Comparison of LosBO and Real-β-SafeOpt in misspecified settings.

with the pre RKHS approach. Qualitatively, we see the same picture, though the performance of both LoSBO and Real-β-SafeOpt appear to be slightly weaker compared to the previous setting. Intuitively, this is to be expected as Matern RKHS functions are generically less smooth than Squared Exponential RKHS functions, and both LoSBO and Real-β-SafeOpt rely on a Lipschitz bound, which is in turn related to regularity of functions.

Misspecified setting We turn to misspecified settings, where the algorithmic parameters do not match the true setting of the target function. This is particularly interesting in the present situation, since the underlying GP model does not impact the safety of LoSBO, and therefore becomes amenable to tuning. The results of the experiments are displayed in Figure 5, where once more the thick solid lines are the means over all 100 functions, the fine lines are the means for each of the individual 100 target functions, and the shaded areas show an interval of width one standard deviation around the mean, again over all functions. We start with the length scale used in the kernel, which is arguably the most important type of hyperparameter in practice. In Figure 5, top left, we show the results for overestimating the length scale in GP regression, using Matern kernels. More precisely, a Matern kernel is used both as the kernel for generating the target functions and as a covariance function in GP regression, but the length scale of the covariance function in GP regression is 4 times the ength scale used in the kernel to generate the target function. The qualitative picture remains the same, though it appears that the performance of Real-β-SafeOpt suffers from the misspecification more than LoSBO. More importantly, in this setting safety violations occur in 12.57 % of all runs. Moreover, for the worst-behaving target function 943 out of 10000 runs lead to safety violations, which is unacceptable in a real-world use case. Figure 5, top right, shows the complementary situation of underestimating the length scale in GP regression, again using Matern kernels. The length scale of the covariance function used in GP regression is 0.2 times the length scale of the kernel that is used to generate the target function. Again, the qualitative picture remains the same, but the performance degradation is worse for both algorithms in this case. Finally, consider the case where a different kernel is used to generate the target functions than the covariance function in the GP regression. We use a Matern-3/2 kernel to generate the target functions, and a Squared Exponential kernel as covariance function in the GP regression, with the same length scale for both. The results are displayed in Figure 5, lower left. Interestingly, essentially no qualitative difference can be noticed compared to the well-specified Matern case (Figure 4, lower left). We suspect that this is due to the correct specification of the length scale, which in the present setting is more important than the kernel misspecification.

7 LoS-GP-UCB

In SafeOpt-type algorithms, including our variant LoSBO (Algorithm 3), the central computational step is the optimization of the acquisition function over the expander and maximizer sets, see Algorithm 1. Inspecting the definition of the latter two sets, see Section 3, makes it clear that computing these sets requires a discrete input set D. For many typical application scenarios, at least parts of the input set will be continuous (e.g., if the optimization variables include a physical parameter that can vary continuously). This means that some form of discretization is necessary before a SafeOpt-type algorithm can be applicable. Typically, equidistant gridding of the (continuous parts of the) input set is used as a discretization, see (Berkenkamp et al., 2016a) for a typical example. As a result, SafeOpt-type algorithms become impractical for even moderate dimensions, e.g., $D \subseteq \mathbb{R}^d$ for $d > 3$ (Kirschner et al., 2019a; Sukhija et al., 2023), and as a member of this class, LoSBO inherits this limitation. In this section, we present and investigate an approach to overcome this issue. Instead of adapting existing solution approaches like (Duivenvoorden et al., 2017; Kirschner et al., 2019a; Sukhija et al., 2023), we suggest a pragmatic and straightforward variant that is motivated by three observations.

First, safety in SafeOpt-type algorithms is ensured by restricting the optimization of the acquisition function to (subsets of) safe sets, i.e., sets $S \subseteq D$ such that $f|_S \geq h$, where f is the unknown target function. In other words, as long as we ensure that the acquisition function optimization is restricted to such sets S, the resulting algorithm will be safe, no matter how this optimization is performed or whether an additional restriction is added (as in SafeOpt, where the optimization is only over expander and maximizer sets). In the case of LoSBO, these safe sets are of a particularly simple form, as they are the union of closed spheres in a metric space, see Figure 6, left, for an illustration for the case $D \subseteq \mathbb{R}^2$. In typical application scenarios of SafeOpt-type algorithms, the number of input queries is relatively low, and hence the aforementioned union is only over relatively few sets.

Second, a discrete input set for SafeOpt-type algorithms is necessary due to the involved definition of the expander and maximizer sets, which in turn are defined to guarantee proper exploration in the original SafeOpt setting (Sui et al., 2015). However, for concrete practical problems, such an underexploration might not pose a severe challenge, and an existing BO algorithm can simply be made safe by restricting the optimization of the acquisition function to a safe set S as described above. In fact, the original SafeOpt paper (Sui et al., 2015) already discussed a safe variant of GP-UCB (Srinivas et al., 2010). This indicates that it might be possible to avoid the complicated sets involved in SafeOpt-type algorithms, and still attain practically useful exploration performance.

Third, in the current practice of BO, moderately high dimensions of the input space are not problematic for modern BO algorithms[8] Typically, acquisition functions are optimized by running a local optimization method (usually gradient-based) from several initial guesses (which might be random, or based on heuristics). In particular, no gridding is necessary, and this strategy can even deal with moderately high dimensions since local search methods behave well despite increasing dimensionality. In fact, state-of-the-art BO libraries like BOTorch (Balandat et al., 2020) implement exactly this approach as the default option.

Based on these three observations, we now propose a straightforward safe BO algorithm that works even in moderate input dimensions, is compatible with modern BO libraries, and retains all the safety guarantees from LoSBO. In Section 7.1 we describe this algorithm in detail and provide some discussion. In Section 7.2 we evaluate the algorithm empirically and compare it to LoSBO.

7.1 Algorithm

We now introduce a practical safe BO algorithm that retains the favorable properties of LoSBO, but also works in moderately high dimensions. Consider the setting of LoSBO as described in Section 6.2.

Motivated by the preceding discussion, we start with a standard GP-based BO algorithm that does not need expander and maximizer sets (or similar complicated sets requiring discretization). Due to its relation to

[8]Here we are referring to handling the dimensionality within the algorithm, specifically optimization of the acquisition function, and not the exploration performance. Of course, the latter poses challenges, especially if not enough structural or qualitative prior knowledge is encoded in the BO function model.

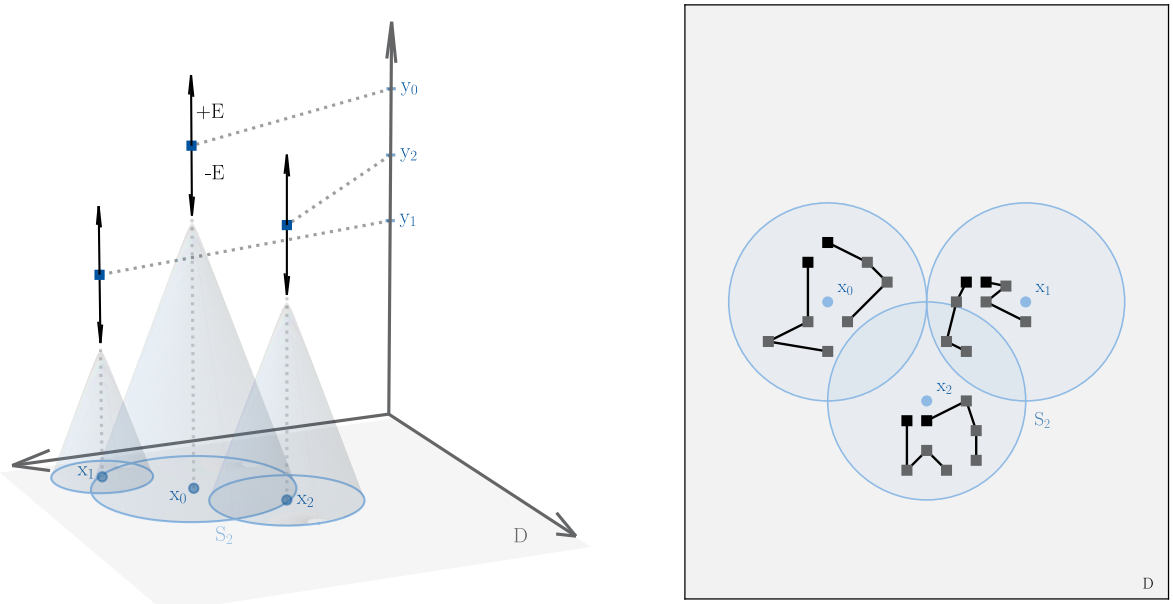

Figure 6: Illustration of safe sets for a 2-dimensional input set. Left: Safe set in LoSBO resulting from three function evaluations. Right: Illustration of the acquisition function optimization in LoS-GP-UCB over a safe set resulting from four function evaluations. Two initial guesses for the local optimization are used for each of the balls that span the safe set according to (15).

SafeOpt, we choose GP-UCB (Srinivas et al., 2010) for this task, so at step $t \geq 1$, the next input is

$$x_{t+1} = \text{argmax}_{x \in D} \mu_t(x) + \beta_t \sigma_t(x), \tag{13}$$

for an appropriate scaling factor $\beta_t \in \mathbb{R}_{>0}$. As usual, ties are broken arbitrarily. Using the (scaled) posterior variance as the acquisition function would be even closer to SafeOpt, but numerical experiments indicate that GP-UCB performs slightly better in this context. Next, we restrict the acquisition function optimization to the safe sets S_t as defined for LoSBO in (11),

$$x_{t+1} = \text{argmax}_{x \in S_t} \mu_t(x) + \beta_t \sigma_t(x). \tag{14}$$

Observe now that

$$S_t = \bigcup_{j=1}^{N_t} \bar{B}_{r_j}(z_j) \tag{15}$$

for some $N_t \in \mathbb{N}_{>0}$, $r_1, \ldots, r_{N_t} \in \mathbb{R}_{>0}$ and $z_1, \ldots, z_{N_t} \in D$, and $\bar{B}_r(z) = \{x \in D \mid d_D(z, x) \leq r\}$ is the closed ball with radius $r \in \mathbb{R}_{>0}$ and center $z \in D$ in the metric space D. For example, if the initial safe set has only one element x_0 and no input is repeatedly sampled, then $N_t = t + 1$, $z_1 = x_0$ and $z_j = x_{j-1}$ for $j = 2, \ldots, t + 1$. Using the decomposition (15), we now have

$$x_{t+1} = \text{argmax}_{j=1,\ldots,N_t} \text{argmax}_{x \in \bar{B}_{r_j}(z_j)} \mu_t(x) + \beta_t \sigma_t(x). \tag{16}$$

Each of the inner optimization problems $\max_{x \in \bar{B}_{r_j}(z_j)} \mu_t(x) + \beta_t \sigma_t(x)$, $j = 1, \ldots, N_t$, is a maximization problem over the convex sets $\bar{B}_{r_j}(z_j)$, and each of these inner problems are independent. In particular, these

Algorithm 4 LoS-GP-UCB

Require: Lipschitz constant L, algorithm to compute β_t, noise bound E, initial safe set S_0, safety threshold h

Compute N_0, $z_1, \ldots, z_{N_0}$, $r_1, \ldots, r_{N_0}$, β_0
for $t = 1, 2, \ldots$ **do**
  $x_t = \text{argmax}_{j=1,\ldots,N_{t-1}} \text{argmax}_{x \in \bar{B}_{r_j}(z_j)} \mu_{t-1}(x) + \beta_{t-1}\sigma_{t-1}(x)$. ▷ Determine next input
  Query function with x_t, receive $y_t = f(x_t) + \epsilon_t$
  Update GP with new data point (x_t, y_t), resulting in mean μ_t and σ_t
  Compute updated β_t
  Compute N_t and add new z_j, r_j
end for

optimizations can be trivially parallelized. In practice, one usually has $D \subseteq \mathbb{R}^d$ (often with a simple geometry) and a differentiable covariance function k, so it is possible to use a gradient-based local optimization method started from multiple initial guesses. This is illustrated in Figure 6, right. All of these multistarts are independent, and can therefore also be parallelized. We thus arrive at Algorithm 4, which we call Lipschitz-only Safe Gaussian Process Upper Confidence Bound (LoS-GP-UCB) algorithm in the following. LoS-GP-UCB can easily be implemented with state-of-the-art BO libraries. For the numerical experiments described in the next section, we have chosen BoTorch (Balandat et al., 2020), which allows an easy parallel implementation of the acquisition function optimization.

Finally, LoS-GP-UCB retains the safety guarantees from LoSBO. In particular, the scaling factors $(\beta_t)_t$ remain tuning factors, and the safety of LoS-GP-UCB is independent of their choice. Furthermore, safety of LoS-GP-UCB *does not require Assumption 2*. Similarly, Remarks 2 and 3 also apply to LoS-GP-UCB.

7.2 Experimental evaluation

Analogously to the case of Real-β-SafeOpt and LoSBO, a frequentist setup will be used, i.e., the algorithm will be run on a fixed target function for many independent noise realizations. In the experiments two aspects will be investigated. First, LoS-GP-UCB will be compared with Real-β-SafeOpt and LoSBO. Second, we apply LoS-GP-UCB to several benchmark functions with moderate input dimensions.

We start with the comparison to Real-β-SafeOpt and LoSBO. Since these two algorithms rely on a discrete input space, this comparison necessarily has to be performed on functions with a low dimensional input. We choose essentially the same experimental settings as in Section 7.2 and consider only the well-specified case. As for LoSBO, we use $\beta_t \equiv 2$ in LoS-GP-UCB, and we consider one-dimensional RKHS functions. The algorithms are evaluated on 100 target functions sampled from the pre RKHS corresponding to a Matern-3/2 kernel, and for each function, we run the algorithms 1000 times[9]. The results of this experiment are shown in 7, where we again use the evaluation metric (12). Thick solid lines are the means over all functions and repetitions and shaded areas correspond to one standard deviation from the mean (again over all functions and realizations). To avoid clutter, the means for each individual function are plotted only for LoS-GP-UCB (thin lines). It is clear from Figure 7 that the performance of Los-GP-UCB is only slightly inferior to the original LosBO algorithm, yet still superior to Real-β-SafeOpt. This outcome indicates that LoS-GP-UCB is not severely affected by the under-exploration problem described for a safe variant of GP-UCB in (Sui et al., 2015). We suspect that, similar to LoSBO, this is due to the overoptimism resulting from setting $\beta_t \equiv 2$, corresponding to moderately aggressive exploration.

Let us turn to the evaluation of LoS-GP-UCB on several benchmark functions with moderate to high input dimensions. As test functions, we use the standard benchmarks Camelback (2d) and Hartmann (6d). Similar to (Kirschner et al., 2019a), we also use a Gaussian function $f(x) = \exp(-4\|x\|_2^2)$ in ten dimensions (10d) as a benchmark. For the Camelback and Hartmann functions, we choose a random initial point in the safe set. For the Gaussian function, we choose a random safe set from the level $f(x_0) = 0.4$. We assume uniformly

[9]The qualitative picture is already clear for 1000 repetitions, hence we choose to save computational resources and do not use 10000 repetitions as in Section 7.2.

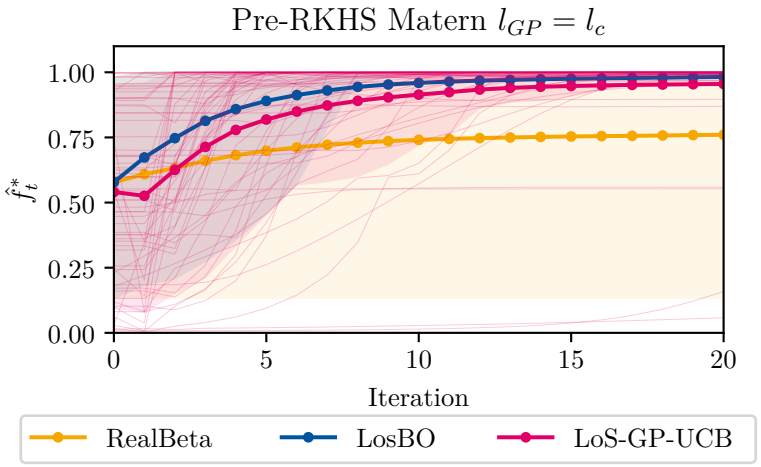

Figure 7: Comparison of LoS-GP-UCB to Real-β-SafeOpt and LoSBO.

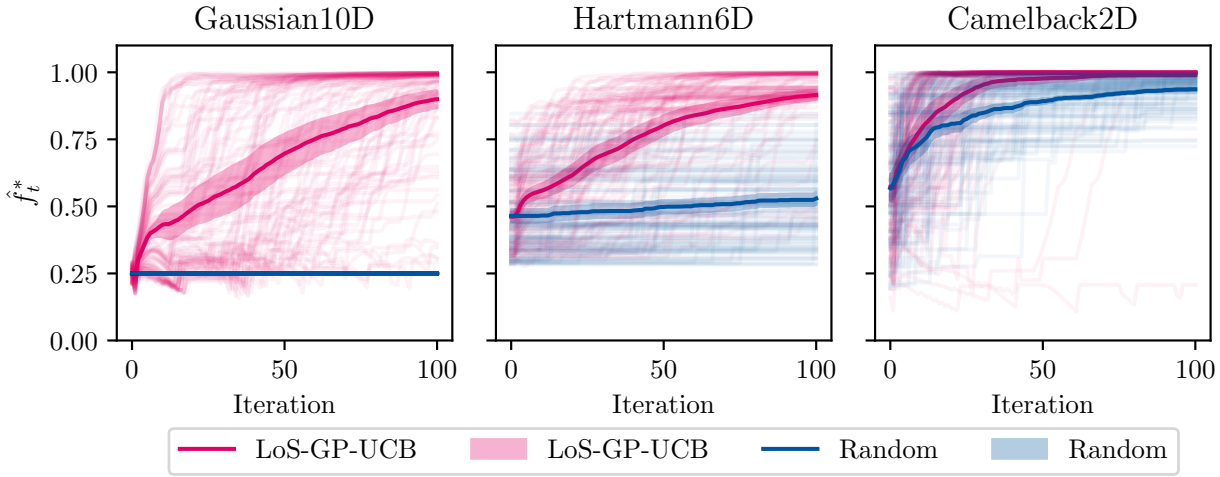

Figure 8: Evaluation of LoS-GP-UCB (pink) on three benchmark functions and comparison to random search (blue). Thick lines are the means over all runs, thin lines are individual runs.

bounded noise with a noise level of 0.01. In this synthetic setting the Lipschitz constant L is determined by evaluating the function on a fine grid. As a model we use a Squared Exponential kernel with output variance set to 1 and length scale set to $1/L$. For the prior mean we choose 0.5 as the function values are between 0 and 1. Finally, in all these settings, we compare LoS-GP-UCB to random search, and run both algorithms from the same initial safe set for 100 iterations, repeating this 100 times (for different random choices of the initial safe set).

8 Conclusion

In this work, we are concerned with practically relevant safety aspects of the important class of SafeOpt-type algorithms. We identified the use of heuristics to derive uncertainty bounds as a potential source of safety violations in the practical application of these algorithms. This prompted us to use recent, rigorous uncer-

tainty bounds in SafeOpt, which allowed us to numerically investigate the safety behavior of this algorithm. We further identified the knowledge of an upper bound on the RKHS norm of the target function as a serious obstacle to reliable real-world applicability of SafeOpt-type algorithms. To overcome this obstacle, we proposed LoSBO, a BO algorithm class relying only on a Lipschitz bound and noise bound to guarantee safety. Numerical experiments demonstrated that this algorithm is not only safe, but also exhibits superior performance.

Ongoing work is concerned with implementing the presented algorithms for safe learning in an automotive context, as well as providing exploration guarantees for LoSBO. We expect that the approach outlined in Section 6 applies to most SafeOpt variants. The derivation, implementation, and evaluation of the corresponding LoSBO-type algorithms for these variants is thus another interesting direction for future work. Our findings in combination with evidence in the literature that SafeOpt and related algorithms have been successfully used in various applications indicate that this algorithm class does not ensure hard safety constraints (in practice), but instead yields "cautious" behavior. The precise connection to conservative bandits and existing cautious BO approaches is another interesting topic for further investigations.

Broader Impact Statement

This work is concerned with safety issues of a popular BO algorithm that has already found numerous applications in real-world scenarios. Henceforth we contribute to the improved safety and reliability of machine learning methods for real-world applications. Furthermore, we expect no adverse societal impact of our work.

References

Yasin Abbasi-Yadkori. Online learning for linearly parametrized control problems. 2013.

Sanae Amani, Mahnoosh Alizadeh, and Christos Thrampoulidis. Linear stochastic bandits under safety constraints. *Advances in Neural Information Processing Systems*, 32, 2019.

Marc Atteia. *Hilbertian kernels and spline functions*. Elsevier, 1992.

Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

Dominik Baumann, Alonso Marco, Matteo Turchetta, and Sebastian Trimpe. Gosafe: Globally optimal safe robot learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4452–4458. IEEE, 2021.

Gleb Beliakov. Interpolation of Lipschitz functions. *Journal of computational and applied mathematics*, 196 (1):20–44, 2006.

Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016a.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 491–496. IEEE, 2016b.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. No-regret Bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 20:1–24, 2019.

Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *Machine Learning*, 112(10):3713–3747, 2023.

Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2004.

Ilija Bogunovic and Andreas Krause. Misspecified Gaussian process bandit optimization. *Advances in Neural Information Processing Systems*, 34:3004–3015, 2021.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Jan-Peter Calliess. *Conservative decision-making and inference in uncertain dynamical systems*. PhD thesis, Oxford University, UK, 2014.

Jan-Peter Calliess, Stephen J Roberts, Carl Edward Rasmussen, and Jan Maciejowski. Lazily adapted constant kinky inference for nonparametric regression and model-reference adaptive control. *Automatica*, 122:109216, 2020.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. *34th International Conference on Machine Learning, ICML 2017*, 2:1397–1422, 2017.

Duane A Cooper. Learning Lipschitz functions. *International journal of computer mathematics*, 59(1-2): 15–26, 1995.

Sébastien Da Veiga, Fabrice Gamboa, Bertrand Iooss, and Clémentine Prieur. *Basics and trends in sensitivity analysis: Theory and practice in R*. SIAM, 2021.

Rikky RPR Duivenvoorden, Felix Berkenkamp, Nicolas Carion, Andreas Krause, and Angela P Schoellig. Constrained Bayesian optimization with particle swarms for safe adaptive controller tuning. *IFAC-PapersOnLine*, 50(1):11800–11807, 2017.

Christian Fiedler. Lipschitz and Hölder continuity in reproducing kernel Hilbert spaces. *arXiv preprint arXiv:2310.18078*, 2023.

Christian Fiedler, Carsten W. Scherer, and Sebastian Trimpe. Practical and rigorous uncertainty bounds for Gaussian process regression. *Proceedings of the AAAI conference on artificial intelligence*, 35, 2021a.

Christian Fiedler, Carsten W. Scherer, and Sebastian Trimpe. Learning-enhanced robust controller synthesis. *60th IEEE Conference on Decision and Control (CDC)*, 2021b.

Christian Fiedler, Carsten W Scherer, and Sebastian Trimpe. Learning functions and uncertainty sets using geometrically constrained kernel regression. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 2141–2146. IEEE, 2022.

Christian Fiedler, Michael Herty, and Sebastian Trimpe. On kernel-based statistical learning in the mean field limit. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

Simon Fischer and Ingo Steinwart. Sobolev norm learning rates for regularized least-squares algorithms. *The Journal of Machine Learning Research*, 21(1):8464–8501, 2020.

Lukas P Fröhlich, Melanie N Zeilinger, and Edgar D Klenske. Cautious Bayesian optimization for efficient and scalable policy search. In *Learning for Dynamics and Control*, pp. 227–240. PMLR, 2021.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Pierre Hansen, Brigitte Jaumard, and Shi-Hui Lu. Global optimization of univariate Lipschitz functions: I. survey and properties. *Mathematical programming*, 55(1-3):251–272, 1992.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Mohamed K. Helwa, Adam Heins, and Angela P. Schoellig. Provably robust learning-based approach for high-accuracy tracking control of Lagrangian systems. *IEEE Robotics and Automation Letters*, 4(2):1587–1594, 2019. ISSN 23773766. doi: 10.1109/LRA.2019.2896728.

José Miguel Hernandez, Michael A Gelbart, Ryan P Adams, Matthew W Hoffman, Zoubin Ghahramani, et al. A general framework for constrained bayesian optimization using information-based search. *Journal of Machine Learning Research*, 17(160):1–53, 2016.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Julien Walden Huang, Stephen J Roberts, and Jan-Peter Calliess. On the sample complexity of Lipschitz constant estimation. *Transactions on Machine Learning Research*, 2023.

Toni Karvonen and Chris J Oates. Maximum likelihood estimation in Gaussian process regression is ill-posed. *Journal of Machine Learning Research*, 24(120):1–47, 2023.

Toni Karvonen, George Wynne, Filip Tronarp, Chris Oates, and Simo Sarkka. Maximum likelihood estimation and uncertainty quantification for Gaussian process approximation of deterministic functions. *SIAM/ASA Journal on Uncertainty Quantification*, 8(3):926–958, 2020.

Youngmin Kim, Richard Allmendinger, and Manuel López-Ibáñez. Safe learning and optimization techniques: Towards a survey of the state of the art. In *International Workshop on the Foundations of Trustworthy AI Integrating Learning, Optimization and Reasoning*, pp. 123–139. Springer, 2020.

Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning*, pp. 3429–3438. PMLR, 2019a.

Johannes Kirschner, Manuel Nonnenmacher, Mojmír Mutnỳ, Andreas Krause, Nicole Hiller, Rasmus Ischebeck, and Andreas Adelmann. Bayesian optimisation for fast and safe parameter tuning of SwissFEL. In *FEL2019, Proceedings of the 39th International Free-Electron Laser Conference*, pp. 707–710. JACoW Publishing, 2019b.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Bandits and experts in metric spaces. *Journal of the ACM (JACM)*, 66(4):1–77, 2019.

Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. Learning-based model predictive control for safe exploration. *Proceedings of the IEEE Conference on Decision and Control*, 2018-Decem: 6059–6066, 2019. ISSN 07431546. doi: 10.1109/CDC.2018.8619572.

Hayri Korezlioglu. Reproducing kernels in separable Hilbert spaces. *Pacific Journal of Mathematics*, 25(2): 305–314, 1968.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Armin Lederer, Jonas Umlauft, and Sandra Hirche. Uniform error bounds for Gaussian process regression with application to safe control. *Advances in Neural Information Processing Systems*, 32, 2019.

Milan Lukić and Jay Beder. Stochastic processes with sample paths in reproducing kernel Hilbert spaces. *Transactions of the American Mathematical Society*, 353(10):3945–3969, 2001.

Cédric Malherbe and Nicolas Vayatis. Global optimization of Lipschitz functions. In *International Conference on Machine Learning*, pp. 2314–2323. PMLR, 2017.

Alonso Marco, Dominik Baumann, Majid Khadiv, Philipp Hennig, Ludovic Righetti, and Sebastian Trimpe. Robot learning with crash constraints. *IEEE Robotics and Automation Letters*, 6(2):1439–1446, 2021.

Mario Milanese and Carlo Novara. Set membership identification of nonlinear systems. *Automatica*, 40(6): 957–975, 2004.

Carlo Novara, Lorenzo Fagiano, and Mario Milanese. Direct feedback control design for nonlinear systems. *Automatica*, 49(4):849–860, 2013.

Baver Okutmuştur. *Reproducing kernel Hilbert spaces.* PhD thesis, Bilkent Universitesi (Turkey), 2005.

Vern I Paulsen and Mrinal Raghupathi. *An introduction to the theory of reproducing kernel Hilbert spaces*, volume 152. Cambridge university press, 2016.

János D Pintér. *Global optimization in action: continuous and Lipschitz optimization. Algorithms, implementations and applications*, volume 6. Springer Science & Business Media, 1995.

C E Rasmussen and C K I Williams. *Gaussian Processes for Machine Learning.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, 2006.

Parisa Sarikhani, Benjamin Ferleger, Jeffrey Herron, Babak Mahmoudi, and Svjetlana Miocinovic. Automated deep brain stimulation programing with safety constraints for tremor suppression. *Brain Stimulation*, 14(6):1699–1700, 2021. ISSN 1935861X. doi: 10.1016/j.brs.2021.10.357. URL https://doi.org/10.1016/j.brs.2021.10.357.

Yaroslav D Sergeyev, Antonio Candelieri, Dmitri E Kvasov, and Riccardo Perego. Safe global optimization of expensive noisy black-box functions in the δ-Lipschitz framework. *Soft Computing*, 24(23):17715–17735, 2020.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Aleksandrs Slivkins et al. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning*, 12(1-2):1–286, 2019.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pp. 1015–1022, 2010. doi: 10.1109/TIT.2011.2182033.

Ingo Steinwart and Andreas Christmann. *Support vector machines.* Springer Science & Business Media, 2008.

Ingo Steinwart and Clint Scovel. Mercer's theorem on general domains: On the interaction between measures, kernels, and RKHSs. *Constructive Approximation*, 35:363–417, 2012.

Ingo Steinwart, Don R Hush, Clint Scovel, et al. Optimal rates for regularized least squares regression. In *COLT*, pp. 79–93, 2009.

RG Strongin. On the convergence of an algorithm for finding a global extremum. *Eng. Cybernetics*, 11: 549–555, 1973.

Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with Gaussian processes. In *International conference on machine learning*, pp. 997–1005. PMLR, 2015.

Yanan Sui, Vincent Zhuang, Joel Burdick, and Yisong Yue. Stagewise safe Bayesian optimization with Gaussian processes. In *International conference on machine learning*, pp. 4781–4789. PMLR, 2018.

Bhavya Sukhija, Matteo Turchetta, David Lindner, Andreas Krause, Sebastian Trimpe, and Dominik Baumann. Scalable Safe Exploration for Global Optimization of Dynamical Systems. 2022. URL http://arxiv.org/abs/2201.09562.

Bhavya Sukhija, Matteo Turchetta, David Lindner, Andreas Krause, Sebastian Trimpe, and Dominik Baumann. Gosafeopt: Scalable safe exploration for global optimization of dynamical systems. *Artificial Intelligence*, 320:103922, 2023.

Aretha L Teckentrup. Convergence of Gaussian process regression with estimated hyper-parameters and applications in Bayesian inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, 8(4):1310–1337, 2020.

Abdullah Tokmak, Christian Fiedler, Melanie N Zeilinger, Sebastian Trimpe, and Johannes Köhler. Automatic nonlinear MPC approximation with closed-loop guarantees. *arXiv preprint arXiv:2312.10199*, 2023.

Alexandre Tsybakov. *Introduction to Nonparametric Estimation*. Springer New York, NY, 2009.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite Markov decision processes with Gaussian processes. *Advances in Neural Information Processing Systems*, (Nips):4312–4320, 2016. ISSN 10495258.

Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

Justin Whitehouse, Aaditya Ramdas, and Steven Z Wu. On the sublinear regret of GP-UCB. *Advances in Neural Information Processing Systems*, 36, 2024.

Yifan Wu, Roshan Shariff, Tor Lattimore, and Csaba Szepesvári. Conservative bandits. In *International Conference on Machine Learning*, pp. 1254–1262. PMLR, 2016.

George Wynne, François-Xavier Briol, and Mark Girolami. Convergence guarantees for Gaussian process means with misspecified likelihoods and smoothness. *The Journal of Machine Learning Research*, 22(1): 5468–5507, 2021.