Unstructured High-Dimensional Bayesian Optimization

Hu Hanyang
Supervisor: Jonathan Scarlett

Department of Mathematics
National University of Singapore
Advanced UROPS in Mathematics for AY2023/2024, Special Term

Abstract

High-dimensional Bayesian optimization (BO) methods typically work by restricting the search space or considering low-dimensional structures. It has recently been suggested that vanilla BO methods could be performant in high dimensions with minor adjustments, such as imposing a low-complexity assumption on the objective function via a dimensionality-scaled lengthscale prior (DSP). However, as a consequence of the stationary kernel used in vanilla BO assuming the function to vary at a consistent rate over the domain, the posterior lengthscales based on measurements could be misspecified when the ground truth objective function does not meet this assumption. In case the estimated lengthscales are erroneously large (especially under the low-complexity assumption), the ground truth objective function might not be considered probable. Hence, DSP may risk the algorithm overlooking regions containing the global optimum. Drawing inspiration from BO methods with unknown hyperparameters in low-dimensional settings, we explore various lengthscale evolution strategies. In addition, we propose soft Winsorization, an easy-to-implement method that attempts to adaptively reduce the objective function's complexity, aligning with the low-complexity assumption made by DSP. Our empirical analysis showcases the effectiveness of soft Winsorization in managing extreme observation outliers. Furthermore, we evaluate the lengthscale evolution strategies and soft Winsorization on several benchmark problems and investigate their respective behaviors.

1 Introduction

Bayesian optimization (BO) is a popular method for optimizing expensive black-box functions with noisy measurements, as it naturally provides a measure of uncertainty and could be sample-efficient by incorporating prior knowledge in the surrogate model. However, BO suffers from the curse of dimensionality [12], with severe performance degradation when scaling to high-dimensional settings. As the dimensionality of the objective function increases, a substantially higher number of queries is needed to explore the exponentially larger high-variance regions in the search space and make reasonable inferences about additional hyperparameters in the surrogate model [4].

Consequently, many approaches proposed for high-dimensional BO attempt to (1) restrict the search space in a local region [8, 16]; or (2) adhere to low-dimensional structural assumptions, such as additive kernels [11] and low-dimensional active subspaces [7, 24]. Recently, Hvarfner et al. have proposed an enhanced lengthscale prior that enables standard Bayesian optimization to outperform existing methods in high-dimensional tasks, without imposing any structural assumptions [9]. Instead, they are assuming that the objective function is simple enough to optimize globally since the Gaussian process surrogate model with larger lengthscales considers less complex candidate functions [5].

However, we have noticed a potential issue with this approach in the aspect of estimating unknown hyperparameters (i.e. lengthscales). Although the stationary kernel used in the Gaussian process surrogate model of vanilla BO enjoys sample efficiency by assuming the objective function changes at a consistent rate over the input space, when such an assumption is violated, it could lead to an erroneous estimate of lengthscales and often causes the algorithm to converge to poor local optima [3]. Especially under a low-complexity assumption, the algorithm might be more prone to overlooking the global optima outside the Gaussian process's confidence interval.

In this report, we aim to explore minor modifications to vanilla BO that could potentially improve performance without imposing further structural assumptions. We investigate various lengthscale evolution strategies inspired by BO methods in low-dimensional settings to alleviate the issue of unknown hyperparameters [3, 22, 23]. Furthermore, we propose soft Winsorization, aiming to adaptively simplify the standardized observations to align with low-complexity assumptions on the objective function. These methods are evaluated and empirically investigated across multiple synthetic and real-world problems.

2 Background

2.1 Problem Statement

We consider the following expensive-to-evaluate black-box function optimization problem

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$$

over the D-dimensional input space $\mathcal{X} = [0,1]^D$, assuming that f can only be observed point-wise and the measurements are subject to a Gaussian noise perturbation $w_n \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, i.e. $y_n = f(\mathbf{x}_n) + w_n$.

2.2 Gaussian Processes

The Gaussian process (GP) [18] provides a distribution over functions $\hat{f} \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$ specified by the mean function $m(\cdot)$ and the covariance function $k(\cdot,\cdot)$. Conditioned on the input $\mathbf{X}_n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and observations $\mathbf{y}_n = (y_1, \ldots, y_n)$, the posterior distribution is $\mathcal{GP}(\mu_n(\cdot), k_n(\cdot,\cdot))$, i.e. at a given location $\mathbf{x} \in \mathcal{X}$, the value of the function $\hat{f}(\mathbf{x})$ is normally distributed with a closed-form solution (involving $m(\cdot)$ and $k(\cdot,\cdot)$) for the mean $\mu_n(\mathbf{x})$ and variance $\sigma_n^2(\mathbf{x}) = k_n(\mathbf{x},\mathbf{x})$.

More specifically, at a given location $\mathbf{x} \in \mathcal{X}$, take the covariance matrix $\mathbf{K}_n \in \mathbb{R}^{t\times t}$ such that $[\mathbf{K}_n]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j \in \{1, \ldots, n\}$ and the vector $\mathbf{k}_n(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_n)]$, we have

$$\mu_n(\mathbf{x}) = m(\mathbf{x}) + \mathbf{k}_n(\mathbf{x})(\mathbf{K}_n + \sigma_\varepsilon^2\mathbf{I})^{-1}(\mathbf{y}_n - m(\mathbf{x}))$$
$$k_n(\mathbf{x},\mathbf{x}') = k(\mathbf{x},\mathbf{x}') - \mathbf{k}_n(\mathbf{x})(\mathbf{K}_n + \sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{k}_n^T(\mathbf{x})$$

In practice, we take $m \equiv 0$ so that the dynamics are fully modeled by $k(\cdot,\cdot)$. We also normalize the inputs $\mathbf{X}_n$ (to make sure the domain is $\mathcal{X} = [0,1]^D$) and standardize the observations $\mathbf{y}_n$, since these settings work best in the BoTorch framework [2].

We focus on stationary kernels such that $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, i.e. the correlations are dependent only on the weighted distance between inputs. The lengthscales $l \in \mathbb{R}^D_{>0}$ are hyperparameters that determine the relative importance of different dimensions, which is called Automatic Relevance Determination (ARD) [27]. Notice that a larger lengthscale indicates that the corresponding dimension is less active. We use the Radial Basis Function (RBF) kernel [10] with ARD lengthscales as in the setting of vanilla BO algorithms [9]:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{1}{2}\sum_{i=1}^{d}\frac{(\mathbf{x}_i - \mathbf{x}'_i)^2}{l_i^2}\right\}$$

Conventionally, l is found through maximum likelihood or maximum a posteriori (MAP) estimation.

2.3 Bayesian Optimization

Bayesian optimization uses a surrogate model (GP, in most cases) together with an acquisition function $\alpha(\mathbf{x} \,|\, \mathbf{X}_n, \mathbf{y}_n)$ to guide the search for the optimum. In the n-th iteration, the candidate point is chosen as

$$\mathbf{x}_* = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x} \,|\, \mathbf{X}_n, \mathbf{y}_n)$$

which is then augmented (together with the observation) to the data to perform the next iteration.

The most common acquisition function is the expected improvement (EI) [15], defined by

$$\alpha_{\mathrm{EI}}(\mathbf{x} \,|\, \mathbf{X}_n, \mathbf{y}_n) = \mathbb{E}_{f(\mathbf{x})}\left[[f(\mathbf{x}) - y_{\max}]_+\right] = \sigma_n(\mathbf{x}) h\left(\frac{\mu_n(\mathbf{x}) - y_{\max}}{\sigma_n(\mathbf{x})}\right)$$

where $y_{\max} = \max_{1\le i\le n} y_i$ is the incumbent and $h(z) = \phi(z) + z\Phi(z)$.[1] The candidate point chosen by EI is in the Pareto-optimal set trading-off between exploitation (mean) and exploration (variance) [6].

3 Related Works

Several works have identified limitations of vanilla BO (especially in high-dimensional settings) and proposed corresponding solutions, which we outline below.

3.1 Locality Issue

The expected distance between uniformly sampled points in a D-dimensional unit hypercube is $\Theta(\sqrt{D})$, i.e. it grows asymptotically as fast as $\sqrt{D}$ [25]. Consequently, as the dimensionality of the problem increases, the distance between data points tends to get larger, which leads to lower correlation and higher model complexity. For kernels such as RBF and Matern$-\frac{5}{2}$, the covariance would decrease exponentially with the squared distance weighted by lengthscales l. In the extreme case, we would have $\mathbf{K} \approx \mathbf{I}$, i.e. the model is uninformed.

Hvarfner et al. [9] have characterized the behavior of vanilla BO in such cases:

Theorem 3.1.1. (Lower Bound on EI Correlation). Assume that $y_{\max} > m(\mathbf{x}_*)$, $\mathbf{K} = \sigma_f \mathbf{I}$ and that the candidate query $\mathbf{x}_* = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha_{\mathrm{EI}}(\mathbf{x})$ correlates with at most one observation. Then the correlation $\rho^* = \sigma_f^{-2} k(\mathbf{x}_*, \mathbf{x}_{\mathrm{inc}})$ between the next query $\mathbf{x}_*$ and the best-observed point $\mathbf{x}_{\mathrm{inc}}$ satisfies

$$\rho^* \sqrt{\frac{1+\rho^*}{1-\rho^*}} \ge \frac{y_{\max} - m(\mathbf{x}_*)}{\sigma_f}$$

where $m(\cdot)$ is the GP mean function.

Notice that a lower bound on correlation is equivalent to an upper bound on the weighted distance for stationary kernels of which the covariance decreases with the distance. That is to say, the algorithm would query at regions exceedingly close to the best-observed point repeatedly when the model is uninformed, despite the existence of other high-variance regions, exhibiting local search behaviors.

[1] $\phi(\cdot)$ and $\Phi(\cdot)$ denote the pdf and cdf of the standard normal distribution respectively.

To overcome this locality issue, they propose a dimensionality-scaled lengthscale prior (DSP), which shifts the location parameter μ of the log-normal prior of lengthscales by $\frac{\log D}{2}$ and hence scales up the mode and mean of the prior distribution by a factor of $\sqrt{D}$:

$$l_n \sim \mathcal{LN}\left(\mu_0 + \frac{\log D}{2}, \sigma_0\right)$$

where (μ_0, σ_0) are suitable parameters for a one-dimensional objective.

3.2 Vanishing Gradient

Intuitively, as more data is observed, the likelihood of improving over the incumbent rapidly decreases. Although EI is never analytically zero under a Gaussian process posterior, it (and its gradient) often vanishes numerically, making gradient-based optimization exceptionally difficult [1]. Especially under high-dimensional settings, the values and gradients of EI are often diminutive in large regions of the domain, and random initialization for optimization likely falls in these regions. [17].

Ament et al. [1] have derived a bound on the probability of encountering numerically vanishing normalized predicted improvement $(\mu_n(\mathbf{x}) - y_{\max})/\sigma_n(\mathbf{x})$ [6] using samples from the distribution $P_{\mathbf{x}}$ to initialize the optimization of the acquisition function.

Theorem 3.2.1. Suppose f is drawn from a Gaussian process prior P_f, $f_{\max}$ is the global optimum of the ground truth function, $y_{\max} \leq f_{\max}$, μ_n, σ_n are the mean and standard deviation of the posterior $P_f(f \mid \mathcal{X}_n)$ and $B \in \mathbb{R}$. Then with probability $1 - \delta$,

$$P_{\mathbf{x}}\left(\frac{\mu_n(\mathbf{x}) - y_{\max}}{\sigma_n(\mathbf{x})} < B\right) \geq P_{\mathbf{x}}(f(\mathbf{x}) < f_{\max} - \varepsilon_n)$$

where $\varepsilon_n = (f_{\max} - y_{\max}) + (\sqrt{-2\log 2\delta} - B)\max_{\mathbf{x}} \sigma_n(\mathbf{x})$.

Consequently, as a BO algorithm closes the simple regret $f_{\max} - y_{\max}$, the RHS increases, and hence $(\mu_n(\mathbf{x}) - y_{\max})/\sigma_n(\mathbf{x})$ is more likely to be small so that EI exhibit numerically vanishing gradients. Furthermore, if the inputs that give rise to high objective values ($\approx f_{\max}$) are concentrated, the RHS will drop slowly as ε_n increases, remaining a high lower bound, such as the Ackley function. They claim that this issue might worsen as the dimensionality grows. To overcome the issue of vanishing gradient, they have introduced the `LogEI` family of acquisition functions that have either identical or approximately equal optima as their canonical counterparts but behave much better numerically.

On another note, Rana et al. [17] have proposed the elastic GP method to deal with the vanishing gradient issue in high-dimensional settings specifically. They have shown that (1) gradients of the acquisition functions become significant for sufficiently large lengthscales; and (2) the extrema of acquisition functions change smoothly as the lengthscales change. Consequently, they start with large lengthscales that do not lead to vanishing gradients and gradually shrink to the learned lengthscales, using the extremum obtained with the previous lengthscales as the initial point to optimize the acquisition function with the shrunk lengthscales.

3.3 Unknown Gaussian Process Hyperparameters

Previous subsections have discussed how having a preference for large lengthscales could alleviate the locality issue and vanishing gradient in the high-dimensional setting. However, directly applying such methods would implicitly assume that the objective has low complexity [9]. In case such an assumption is false, the model would tend to underfit with the observations and get stuck in local optima [3]. A typical strategy to deal with this issue of unknown hyperparameters is to cool down lengthscales along the optimization process.

Wabersich et al. [22] have proposed the alpha ratio (AR) cool down strategy which uses the acquisition function values to determine whether to shrink the lengthscales. They choose a constant lower bound on lengthscales $\bar{l}$. For each iteration, let l_{n-1} be the lengthscales used in the previous iteration and $\alpha_n^*(l) := \max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x} \mid \mathbf{X}_{n-1}, \mathbf{y}_{n-1}, l)$ be the optimal acquisition function value given observations $(\mathbf{X}_{n-1}, \mathbf{y}_{n-1})$ and lengthscales l. Put $l'_n = \max(\bar{l}, l_{n-1}/2)$, if the alpha ratio $\alpha_{r,n} := \alpha_n^*(l'_n)/\alpha_n^*(l_{n-1})$ is larger than a certain threshold (e.g. $\alpha_{r,n} > 1.5$), take $l_n = l'_n$; otherwise, keep $l_n = l_{n-1}$.

Berkenkamp et al. [3] have proposed adaptive GP-UCB (A-GP-UCB) which scales down the lengthscales and scales up the RKHS norm bound (which expands the GP's confidence intervals) according to schedulers that may depend on the data collected so far at each iteration. In doing so, A-GP-UCB gradually expands the space of candidate functions for consideration.

Notice that both methods mentioned above start with an initial guess of lengthscales l_0 and gradually shrink them without altering the model's weighting of different dimensions' relative significance (characterized by the proportion of lengthscales). In contrast to directly shrinking the estimated lengthscales, Wang et al. [23] have proposed to reduce the upper bound on lengthscales and restrict lengthscales estimation within the evolving bounds, with the assumption that an upper bound on the lengthscales exists. They lower the upper bound of lengthscales when the algorithm repeatedly queries points of low posterior variance in comparison with the noise variance.

4 Methods

4.1 Lengthscales Cool Down

We maintain a base length $L \in [\bar{L}, 1]$ ($\bar{L} > 0$) and evolve them with different strategies (e.g. AR cool down). We consider the following meta-strategies, which affect lengthscales via L in different manners:

Option 1. We could directly follow the practice in low-dimensional settings [3, 22], i.e. keep the proportion of lengthscales fixed during the entire optimization process. In this case, given the initial estimation l_0, the lengthscales at the n-th iteration should be $l_n = Ll_0$. Notice that this practice in low-dimensional settings may not be directly applicable to higher dimensions when a larger number of hyperparameters need to be estimated, although the algorithm could sample more points at the initializing stage for problems of higher dimensions. We have observed that the initial guess l_0 may not reflect the importance of different dimensions in synthetic functions embedded in high-dimensional space, as shown in Figure 1. Consequently, this meta-strategy is not implemented.

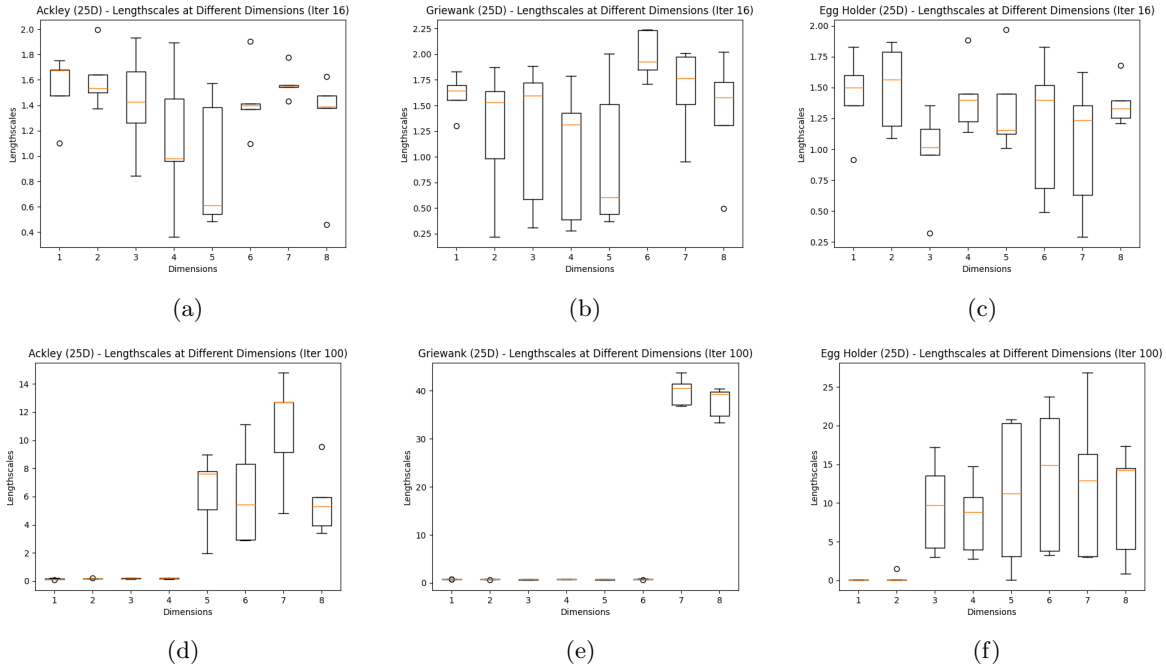


Figure 1: Box plots indicating distributions of lengthscales for the first 8 dimensions estimated for vanilla BO running on sparse synthetic test functions embedded in 25-dimensional space over 5 repetitions, at the 16th and 100th iterations respectively. During the initialization stage, $3\sqrt{D} = 15$ points are generated from the Sobol engine. However, the model initially fails to identify the active dimensions, which are only recognized later in the optimization process.

Option 2. Alternatively, we could modify the log-normal prior in Section 3.1 to

$$l_n \sim \mathcal{LN}\left(\mu_0 + \frac{\log D}{2} + \log L, \sigma_0\right)$$

When $L \leq 1/\sqrt{D}$, the model resumes the high-complexity assumption.

Option 3. Even if the model assumes high complexity on the objective (i.e. the prior is suggestive of short lengthscales), the posterior probability mass may still be concentrated around lengthscales that are erroneously larger since stationary kernels do not consider functions of varying rates of change to be likely [3]. Consequently, the third option is to scale the posterior lengthscales by the base length L directly, i.e.

$$\begin{aligned} l'_n &\sim \mathcal{LN}\left(\mu_0 + \frac{\log D}{2}, \sigma_0\right) \\ l_n &= \max(Ll'_n, \bar{l}) \end{aligned}$$

Lengthscales Cool Down Option 1 (not implemented)

init $\mathbf{X}_0 \subseteq \mathcal{X}$, $\mathbf{y}_0 \subseteq \mathbb{R}$ ▷ generate initial inputs, the size depends on dimension
init $l_0 \in \mathbb{R}^D_{>0}$ ▷ E.g., via ML/MAP estimation
$L \leftarrow 1$, $n \leftarrow 1$
for $n \leq N$ **do**
  $\mathbf{y}'_{n-1} \leftarrow (\mathbf{y}_{n-1} - \text{avg}(\mathbf{y}_{n-1}))/\text{std}(\mathbf{y}_{n-1})$ ▷ standardize the observations
  if should_shrink$(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, L, n)$ **then**
    $L \leftarrow \max(L/2, \bar{L})$
  end if
  $l_n \leftarrow \max(Ll_0, \bar{l})$ ▷ proportions of lengthscales on different dimensions are fixed
  $\mathbf{x}_n \leftarrow \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x} \mid \mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, l_n)$ ▷ optimize acquisition function value
  $\mathbf{X}_n \leftarrow \mathbf{X}_{n-1} \cup \{\mathbf{x}_n\}$
  $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} \cup \{f(\mathbf{x}_n) + w_n\}$ ▷ query new sample (subject to random noise)
  $n \leftarrow n + 1$
end for

Lengthscales Cool Down Option 2

init $\mathbf{X}_0 \subseteq \mathcal{X}$, $\mathbf{y}_0 \subseteq \mathbb{R}$, and $l_0 \in \mathbb{R}^D_{>0}$
$L \leftarrow 1$, $n \leftarrow 1$
for $n \leq N$ **do**
  $\mathbf{y}'_{n-1} \leftarrow (\mathbf{y}_{n-1} - \text{avg}(\mathbf{y}_{n-1}))/\text{std}(\mathbf{y}_{n-1})$
  if should_shrink$(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, L, n)$ **then**
    $L \leftarrow \max(L/2, \bar{L})$
  end if
  $l_n \leftarrow \text{MAP}(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, L)$
  $\mathbf{x}_n \leftarrow \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x} \mid \mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, l_n)$
  $\mathbf{X}_n \leftarrow \mathbf{X}_{n-1} \cup \{\mathbf{x}_n\}$
  $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} \cup \{f(\mathbf{x}_n) + w_n\}$
  $n \leftarrow n + 1$
end for

Lengthscales Cool Down Option 3

init $\mathbf{X}_0 \subseteq \mathcal{X}$, $\mathbf{y}_0 \subseteq \mathbb{R}$, and $l_0 \in \mathbb{R}^D_{>0}$
$L \leftarrow 1$, $n \leftarrow 1$
for $n \leq N$ **do**
  $\mathbf{y}'_{n-1} \leftarrow (\mathbf{y}'_{n-1} - \text{avg}(\mathbf{y}_{n-1}))/\text{std}(\mathbf{y}_{n-1})$
  if should_shrink$(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, L, n)$ **then**
    $L \leftarrow \max(L/2, \bar{L})$
  end if
  $l'_n \leftarrow \text{MAP}(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, 1)$
  $l_n \leftarrow \max(Ll'_n, \bar{l})$
  $\mathbf{x}_n \leftarrow \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x} \mid \mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, l_n)$
  $\mathbf{X}_n \leftarrow \mathbf{X}_{n-1} \cup \{\mathbf{x}_n\}$
  $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} \cup \{f(\mathbf{x}_n) + w_n\}$
  $n \leftarrow n + 1$
end for

On particular strategies to evolve the base length L, we may try out alternative methods (i.e., modifying the `should_shrink` method) other than AR cool down, such as a fixed lengthscale scheduler that gradually shrinks lengthscales according to a pre-defined schedule (potentially depending on the objective's dimensionality).

We could also use a success/failure counter inspired by TuRBO [8] to evolve the base length L, i.e. L expands after τ_{succ} consecutive successes and shrinks after τ_{fail} consecutive failures, where "success"

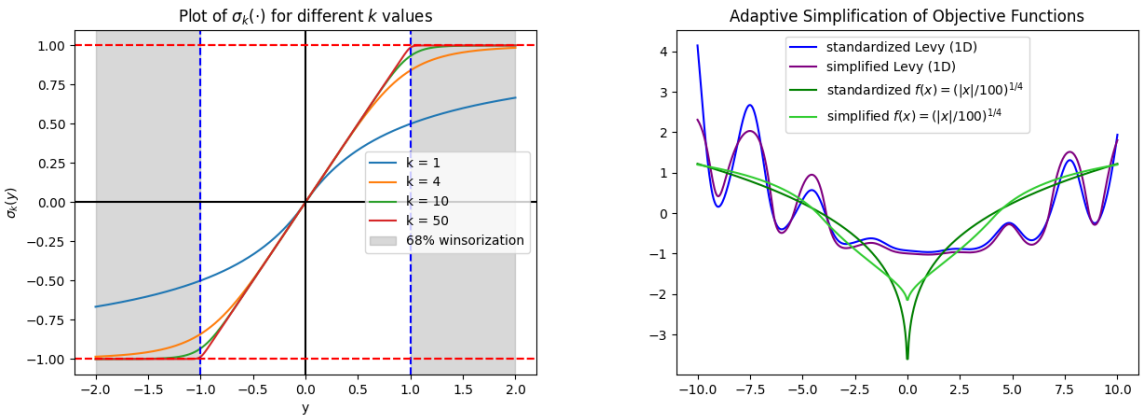
and "failure" are regarding whether the candidate has improved upon the previous best observation. The potential benefit of this TuRBO-like approach is that there are chances to restore the preference for larger lengthscales. Moreover, it might implicitly incorporate the mechanism suggested by Wang et al. [23] mentioned in Section 3.3. When the algorithm repeatedly samples points of low posterior variance, the sampled points tend to have low probabilities of improvement (PI), i.e. it becomes more likely to fail in making improvements to the incumbent. This is because PI at the queried point $\mathbf{x}_*$ decreases as the posterior variance $\sigma_n^2(\mathbf{x}_*)$ decreases when $y_{\max} > \mu_n(\mathbf{x}_*)$ [6].

4.2 Soft Winsorization

4.2.1 Adaptive Simplification of Observations

Sections 3.1 and 3.2 suggest that a "simpler" assumption on the objective makes the algorithm less likely to suffer from the locality issue and vanishing gradient. Section 3.3 also discusses the concern of unknown hyperparameters, particularly when the objective function does not align with the model's assumptions. Consequently, it is natural to ponder: since we made a low-complexity assumption on the objective, instead of gradually making more complex assumptions (e.g. lengthscales cool down), can we simply transform the objective (or observed values) to make it appear "simpler"?

For a "simpler" objective, it could have lower $y_{\max}$ and less concentrated regions of high values, or be more likely to be contained in the GP's confidence interval under a low-complexity assumption. Directly scaling by a constant factor $c < 1$ would not help in practice since the observed values are standardized before being fitted by the GP surrogate. Consequently, we apply a sigmoid function to the standardized observed values, conjecturing that it can implicitly simplify the objective (Alg. 4). We choose $\sigma_k(y) = y(\sqrt[k]{1+|y|^k})^{-1}$ where k is a positive integer. Notice that $\sigma_k(\cdot)$ is strictly monotonic, hence the optima after simplification should be identical to that of the original objective function. Furthermore, this transformation can be interpreted as applying "soft" Winsorization (i.e. a smooth approximation of Winsorization, as shown in Figure 2(a)) to the observed data, since it is equivalent to a 68% Winsorization[2] when $k \to \infty$. This percentage can be changed by scaling the sigmoid function $\sigma_k(\cdot)$, i.e. we can choose a constant $C > 0$ and put $\sigma_{k,C} = C\sigma_k(y/C)$. When $C \to \infty$, we have $\sigma_{k,C}(y) \to y$ for all $y \in \mathbb{R}$.

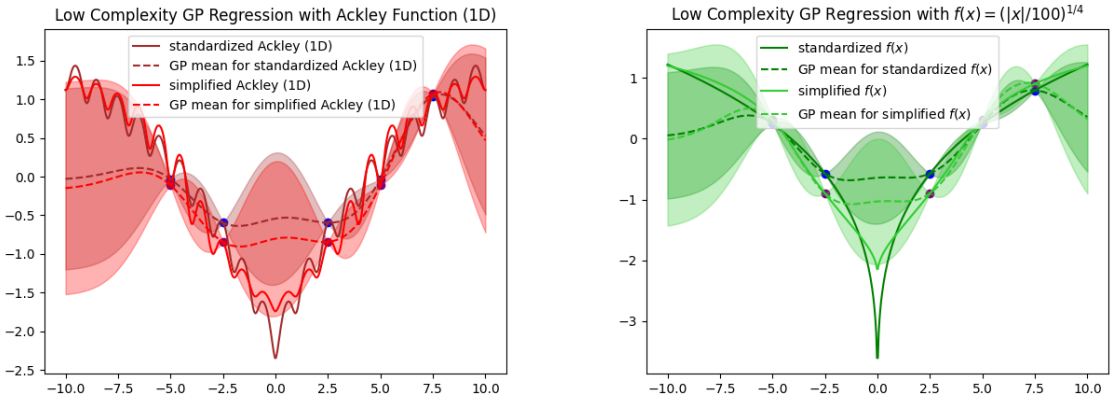


(a) (b)

Figure 2: (a) As $k \to \infty$, $\sigma_k(\cdot)$ would take values below -1 as -1 and values above 1 as 1. Since the data is standardized, this is equivalent to applying 68% Winsorization to the observed values. (b) The standardized 1D Levy function and a standardized spiky function $f(x) = (|x|/100)^{1/4}$ before and after applying the sigmoid function $\sigma_{k,C}(y)$ $(k = 1, C = 1.5)$ and being re-standardized. Notice that standardized values that are far from the origin are scaled adaptively. The 1D Levy function is not changed too much since it is simple already, yet $f(x) = (|x|/100)^{1/4}$ is simplified with a less spiky shape.

[2]Winsorization is the transformation of statistics to alleviate the effect of outliers [26]. A 68% Winsorization would set all data below the 16th percentile to the 16th percentile, and data above the 84th percentile to the 84th percentile, i.e. data that are not within 1 standard deviation from the mean are considered outliers and are restricted.

Figure 2(b) and 3 demonstrate how soft Winsorization might be useful in the context of Gaussian process regression with low-complexity assumption. Notably, it could simplify objective functions adaptively, with a minor impact on simple objectives such as the Levy function, and more obvious twists for hard objectives with spiky regions, such as the Ackley function or $f(x) = (|x|/100)^{1/4}$.



(a) (b)

Figure 3: In the context of GP regression, the simplified functions (for the 1D Ackley function and $f(x) = (|x|/100)^{1/4}$ respectively) are more likely to be contained in the GP's 95% confidence intervals with a low-complexity assumption ($l = 2.0$).

However, if the objective has a highly oscillatory landscape, such as the Griewank function, soft Winsorization may not alleviate these oscillations. In addition, the inputs would not be evenly distributed in the BO context. For example, when the BO algorithm turns toward more exploitation, the input would concentrate on regions that give rise to higher values. Hence, the mean and standard deviation of the observations would be biased, and it could be difficult to interpret the effect of soft Winsorization in such a situation.

Alternatively, instead of applying soft Winsorization to standardized observations directly, we may utilize a bootstrapping approach in practice - we fit a separate GP surrogate model with the standardized observations, sample $N = 1024$ points from this model, and use them to estimate the mean and standard deviation (Alg. 5). By applying bootstrapping, the issue of concentrated inputs could be alleviated. However, we have observed performance degradation for this approach in most of our experiments.

Soft Winsorization

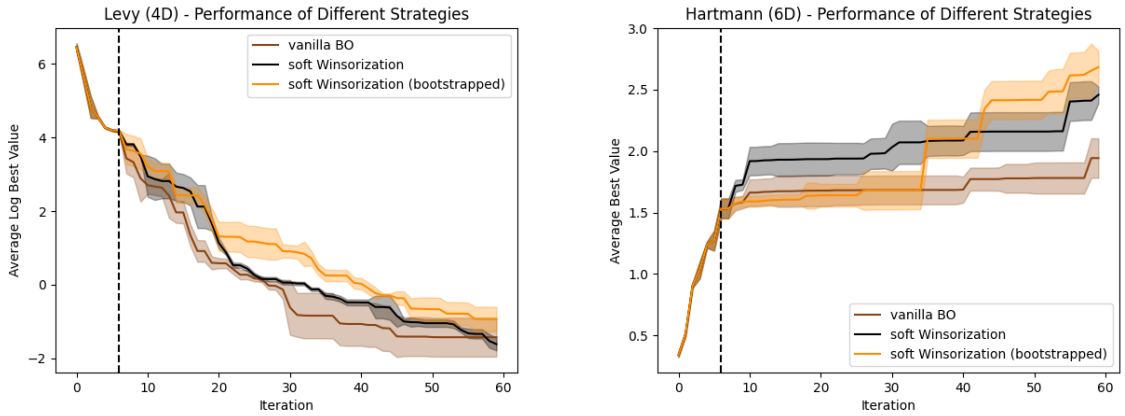
init $k \in \mathbf{N}$, $C \in \mathbb{R}_{>0}$
init $\mathbf{X}_0 \subseteq \mathcal{X}$, $\mathbf{y}_0 \subseteq \mathbb{R}$, and $l_0 \in \mathbb{R}^D_{>0}$
$L \leftarrow 1$, $n \leftarrow 1$
for $n \leq N$ **do**
  $\mathbf{y}'_{n-1} \leftarrow (\mathbf{y}_{n-1} - \text{avg}(\mathbf{y}_{n-1}))/\text{std}(\mathbf{y}_{n-1})$
  $\mathbf{y}'_{n-1} \leftarrow \sigma_{k,C}(\mathbf{y}'_{n-1})$
  $\mathbf{y}'_{n-1} \leftarrow (\mathbf{y}'_{n-1} - \text{avg}(\mathbf{y}'_{n-1}))/\text{std}(\mathbf{y}'_{n-1})$
  $l_n \leftarrow \text{MAP}(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, 1)$
  $\mathbf{x}_n \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x} \,|\, \mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, l_n)$
  $\mathbf{X}_n \leftarrow \mathbf{X}_{n-1} \cup \{\mathbf{x}_n\}$
  $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} \cup \{f(\mathbf{x}_n) + w_n\}$
  $n \leftarrow n + 1$
end for

Soft Winsorization (Bootstrapped)

init $k \in \mathbf{N}$, $C \in \mathbb{R}_{>0}$, $N \in \mathbf{N}$
init $\mathbf{X}_0 \subseteq \mathcal{X}$, $\mathbf{y}_0 \subseteq \mathbb{R}$, and $l_0 \in \mathbb{R}^D_{>0}$
$L \leftarrow 1$, $n \leftarrow 1$
for $n \leq N$ **do**
  $\mathbf{y}'_{n-1} \leftarrow (\mathbf{y}_{n-1} - \text{avg}(\mathbf{y}_{n-1}))/\text{std}(\mathbf{y}_{n-1})$
  $l'_n \leftarrow \text{MAP}(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1}, 1)$
  Fit GP with $(\mathbf{X}_{n-1}, \mathbf{y}'_{n-1})$ and l'_n
  Sample N points $(\mathbf{X}', \mathbf{y}')$ from GP posterior
  $\mathbf{y}''_{n-1} \leftarrow \sigma_{k,C}((\mathbf{y}'_{n-1} - \text{avg}(\mathbf{y}'))/\text{std}(\mathbf{y}'))$
  $\mathbf{y}''_{n-1} \leftarrow (\mathbf{y}''_{n-1} - \text{avg}(\mathbf{y}''_{n-1}))/\text{std}(\mathbf{y}''_{n-1})$
  $l_n \leftarrow \text{MAP}(\mathbf{X}_{n-1}, \mathbf{y}''_{n-1}, 1)$
  $\mathbf{x}_n \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x} \,|\, \mathbf{X}_{n-1}, \mathbf{y}''_{n-1}, l_n)$
  $\mathbf{X}_n \leftarrow \mathbf{X}_{n-1} \cup \{\mathbf{x}_n\}$
  $\mathbf{y}_n \leftarrow \mathbf{y}_{n-1} \cup \{f(\mathbf{x}_n) + w_n\}$
  $n \leftarrow n + 1$
end for

4.2.2 Optimization in the Presence of Outliers

Gaussian processes and Bayesian optimization are generally disadvantaged when the observations are subject to severe variability, i.e. populated by outliers [14, 19]. Since Winsorization is known as a technique to overcome outliers, we may hope it could mitigate the issue of extremely outlying observations. Figure 4 has shown that soft Winsorization outperforms vanilla BO on synthetic tasks in the setting with large random noise (i.i.d. sampled from Uniform$(-60, 60)$, injected to 16.7% of the observations), even in low dimensions.



(a) (b)

Figure 4: Synthetic Test Functions with Large Uniform Noise. To demonstrate the ability of these methods to optimize robustly, we have recorded the best-so-far re-evaluations of the queried points without injecting large random noises. Each strategy runs with 10 repetitions. The consistency of each method's performance is measured by the median absolute deviation (MAD) of the best values over repeated runs. (a) Average log best values of the points queried by different strategies for the Levy function in 4-dimensional space, soft Winsorization seems to perform stabler. (b) Average best values of the points queried by different strategies for the Hartmann function in 6-dimensional space. Notably, soft Winsorization could improve performance while vanilla BO rarely queried better points after the random initialization.

We have also compared the performance between vanilla BO and soft Winsorization on an environment in OpenAI Gymnasium [21], since simulations in such environments are generally sensitive to even tiny perturbations. We have taken the lunar lander task as an example. This is a classic rocket trajectory optimization problem to determine whether to fire engines of one direction (left, right, or up) or do nothing based on the input providing the coordinates, linear/angular velocities, and other sensor information [21]. Similar to the experiment settings in TuRBO [8], we use a heuristic controller with 12 learnable parameters and set the `INITIAL_RANDOM` variable to 1500 to give more uncertainty to the initial position of the lunar lander. We run each method for 5 repeated trials. For each trial, $n = 200$ points are queried. We put batch size $q = 1$ instead of $q = 50$ in TuRBO's setting. Instead of plotting the best values observed in each iteration for each method, we extract the policy after the optimization process and compare their performance across 100 simulations. More specifically, we select parameters with the highest lower confidence bound $\mathbf{x}_* = \arg\max_{\mathbf{x}\in\mathbf{X}} \mu_n(\mathbf{x}) - \beta\sigma_n(\mathbf{x})$ from the GP posterior (where we take $\beta = 1$). The results are demonstrated in Table 1.

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
vanilla BO	229.73 ± 27.34	-67.38 ± 97.84	$\mathbf{41.65} \pm 65.33$	37.93 ± 105.82	70.20 ± 145.72
soft Winsorization	$\mathbf{245.46} \pm 24.53$	$\mathbf{18.02} \pm 21.05$	-15.90 ± 40.58	$\mathbf{253.39} \pm 48.16$	$\mathbf{259.26} \pm 17.46$

Table 1: Lunar Lander. The mean and standard deviation of the cumulative rewards in 100 simulations for the parameters selected by vanilla BO and soft Winsorization in each trial. Parameters selected by soft Winsorization tend to perform better (except in Trial 3) and behave more consistently.

5 Results

We compare vanilla BO (with DSP), soft Winsorization ($k = 1$, $C = 1.5$) with or without bootstrapping, AR cool down (thresholding at 1.0), success/failure counter (with $\tau_{\text{succ}} = 10$ and $\tau_{\text{fail}} = \max(4, D)$ depending on the objective's dimension D), and a fixed scheduler that shrinks the base length L by 0.7 for every $10\sqrt{D}$ iterations, taking the average of performance over 5 repetitions. For each lengthscales cool down strategy, we have implemented them using option 2 and option 3 (labeled as `opt2` and `opt3`) respectively as discussed in Section 4.1. Following the practice in Hvarfner et al. [9], we set $\mu_0 = \sqrt{2}$ and $\sigma_0 = \sqrt{3}$ for the DSP, and use `qLogNEI` (with $q = 1$) from the Noisy EI family [13] instead of the analytical EI described in Section 2.3. The major difference between `qLogNEI` and `qLogEI` is the choice of incumbent [1].

On another note, the base length L of `AR cool down opt2` dropped rapidly as its threshold was triggered easier than that of its counterpart using option 3. Hence the method raised errors indicating that the optimizer cannot find a MAP estimation numerically. We omit this approach in our experiments due to its numerical instability.

Generally, we observe that all lengthscales cool down methods using option 2 have delivered relatively similar performance to the vanilla BO baseline (especially obvious in the experiment on the egg holder function), suggesting that modifying parameters in the prior of lengthscales (option 2) has less effect on the performance compared with directly scaling the posterior lengthscales (option 3). This could also be observed from the comparative plot for analytical EI values of the fixed scheduler method implemented in options 2 and 3 respectively.[3] The method using option 2 is usually less sensitive to the shrinkage of the base length L, as shown in the figures.

In addition, we observe that the success/failure counter method (option 3) could recover from over-exploration. This is demonstrated in the experiments on the Ackley function and the Griewank function, both of which have their global optimum at the origin [2].

5.1 Synthetic Test Functions

We use sparse synthetic test functions embedded in 25-dimensional space implemented in the BoTorch library [2] to compare the performance of lengthscales cool down strategies and the soft Winsorization method to that of the vanilla BO baseline. We deliberately choose synthetic functions that are considered to be "hard" by having multiple local optima and/or concentrated regions that give rise to high values, e.g. the Ackley function, Griewank function, and egg holder function. Furthermore, we perturb observations with random noise of standard deviation $\sigma_\varepsilon = 10^{-2}$.

5.1.1 Ackley Function

We evaluate different strategies on `ackley4_25`, i.e. 4-dimensional Ackley function embedded in 25-dimensional space. Figure 5(a) demonstrates the average log best values for each method at each iteration. Although we do not present the consistency of each method by their MAD, We have observed that the AR cool down method is relatively unstable, while the fixed scheduler method (option 3) outperformed the rest and delivered consistent performance. Figure 5(b) demonstrates the expected improvement of the point to query for the fixed scheduler strategy (with `seed=43`). Observe that the EI values for `fixed scheduler opt3` have lifted when the base length L shrinks. On the other hand, such a pattern appears less obvious for `fixed scheduler opt2`.

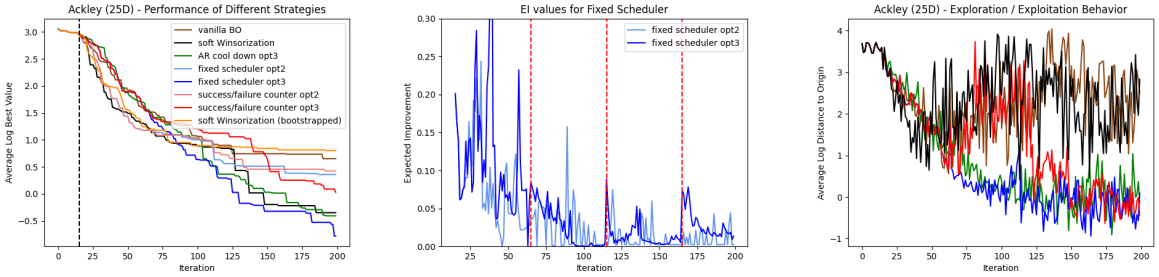
Since the global optimum of the Ackley function is at the origin [2], we plot the average log distances of the queried points to the origin for each method at each iteration in Figure 5(c) to understand the exploration/exploitation behavior of different approaches visually. We only use the coordinates of the important dimensions (i.e., the first 4 dimensions for `ackley4_25`) to compute the distance. The distance plots of soft Winsorization (bootstrapped), fixed scheduler (option 2), and success/failure counter (option 2) are omitted since they have demonstrated similar behavior to that of the vanilla BO baseline.

[3] The small discrepancies in EI values before the first base length shrinkage takes place are unexpected. We conjecture that these differences might be attributed to some numerical errors.

5.1.2 Griewank Function

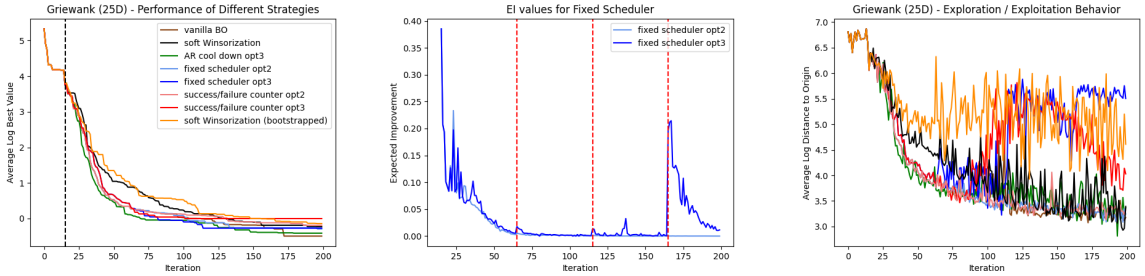
The Griewank function has widely spread local optima and a global optimum at the origin similar to the Ackley function [2], with a particularly oscillatory landscape. We evaluate different strategies on `griewank6_25`. The results are shown in Figure 6. Notice that all methods fail to surpass the vanilla BO baseline in this experiment.

Similarly, we plot the average log distances of the queried points to the origin for each method at each iteration in Figure 6(c), with only the first 6 important dimensions of the sparse synthetic function taken into consideration.



(a) (b) (c)

Figure 5: Ackley Function. (a) Soft Winsorization ($k = 1, C = 1.5$) without bootstrapping and the methods that shrink lengthscales via option 3 surpassed the performance of vanilla BO and their counterpart using option 2 (if any, which also surpassed the baseline). (b) For the fixed scheduler method, EI values lift at iterations 65, 115, and 165 when the base length L shrinks. (c) Vanilla BO, fixed scheduler (option 2), success/failure counter (option 2), and soft Winsorization (w/wo bootstrapping) tend to be more explorative, whilst AR cool down and fixed scheduler (option 3) tend to be more exploitative. The success/failure counter (option 3) could recover from exploration to exploitation.

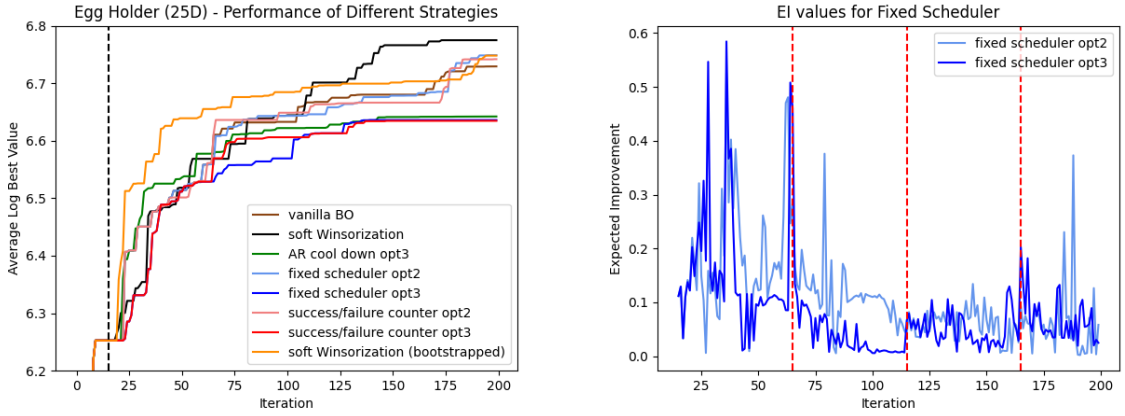


(a) (b) (c)

Figure 6: Griewank Function. (a) Vanilla BO performs better than soft Winsorization ($k = 1, C = 1.5$) w/wo bootstrapping and all lengthscales shrinking methods on average. (b) For the fixed scheduler method (option 3), EI values lift at iteration 165. For its counterpart using option 2, such behavior is not observed. (c) AR cool down (option 3) has reached closest to the origin on average, whilst lengthscales cool down methods using option 3 (except AR cool down) have started exploring regions far from the origin. The success/failure counter (option 3) could recover from exploration to exploitation.

5.1.3 Egg Holder Function

We also test those strategies on `eggholder2_25` (which has many local optima) with similar settings. The results are shown in Figure 7, and the soft Winsorization ($k = 1$, $C = 1.5$) approach without bootstrapping performs the best among all tested methods. Notice that the performance of lengthscales cool down methods implemented in option 2 are generally similar to that of the vanilla BO baseline.



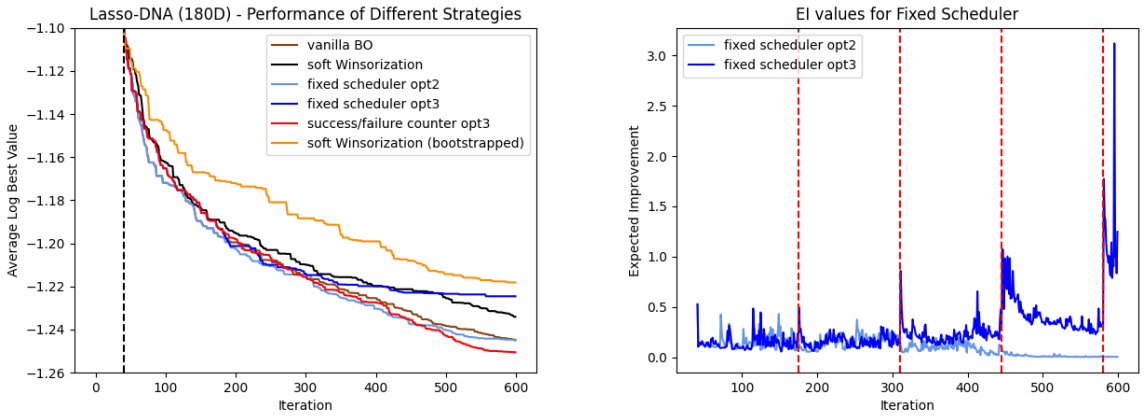
(a) (b)

Figure 7: Egg Holder Function. (a) Soft Winsorization ($k = 1, C = 1.5$) w/wo bootstrapping, fixed scheduler (option 2), and success/failure counter (option 2) surpassed the performance of vanilla BO on average. On the other hand, methods implemented using option 3 have delivered similar performance, below that of the baseline. (b) For the fixed scheduler, EI values lift at iterations 65, 115, and 165 when the base length L shrinks.

5.2 Real-World Tasks

5.2.1 Lasso-DNA

Lasso is a linear regression technique that minimizes the residual sum of squares subject to a hard upper bound on the ℓ^1 norm of the coefficients, which tends to produce a sparse model with only a few non-zero coefficients, favoring the typical situation when the number of observations is relatively small comparing to the number of features [20]. We run experiments on Lasso-DNA [28], which is a high-dimensional hyperparameter optimization (HD-HPO) problem to find separate constant constraints for different features, given 2000 data points with 180 features (approximately 43 active dimensions), the results are displayed in Figure 8. We observe that the base length L for the AR cool down approach rarely changes during the experiment.

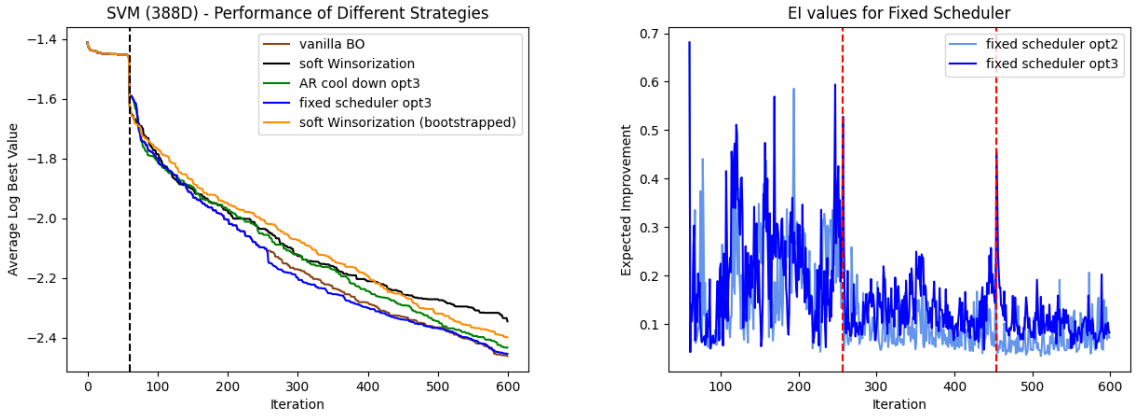


(a) (b)

Figure 8: Lasso-DNA Task. (a) The success/failure counter (option 3) outperforms other methods. The omitted performance of AR cool down (option 3) and success/failure counter (option 2) are almost the same as that of vanilla BO. (b) For fixed scheduler (option 3), the EI values lift at iteration 175, 310, 445, and 580. Such behavior is not observed for its counterpart using option 2.

5.2.2 Hyperparameter Tuning of an SVM

This is a hyperparameter tuning problem for training a kernel support vector machine (SVM) with 385-dimensional data and 3 regularization parameters, i.e. the problem has 388 dimensions in total with only a small number of them being important [7]. The results are shown in Figure 9.



(a) (b)

Figure 9: SVM. (a) Soft Winsorization ($k = 1$, $C = 1.5$) w/wo bootstrapping has performed worse compared to the baseline. The omitted performance of the success/failure counter (options 2 and 3) and the fixed scheduler (option 2) are almost identical to that of the vanilla BO baseline. (b) For the fixed scheduler method, EI values lift at iterations 257 and 454 when the base length L shrinks.

6 Discussion

6.1 The Behavior of Success/Failure Counter

We could not definitively conclude that the observed behavior of `success/failure counter opt3` in recovering from over-exploration to exploitation is attributed to its ability to restore the preference for larger lengthscales as aforementioned in Section 4.1. Note that our choice of the success tolerance, $\tau_{\text{succ}} = 10$, follows from the BoTorch tutorials [2] and differs from the original setting of $\tau_{\text{succ}} = 3$ for TuRBO-1 [8]. This higher success tolerance makes the base length L less likely to expand and restore the original value after shrinkage. However, we observe that the behavior of recovering from over-exploration appears more obvious with a larger success tolerance, as demonstrated in Figure 10.

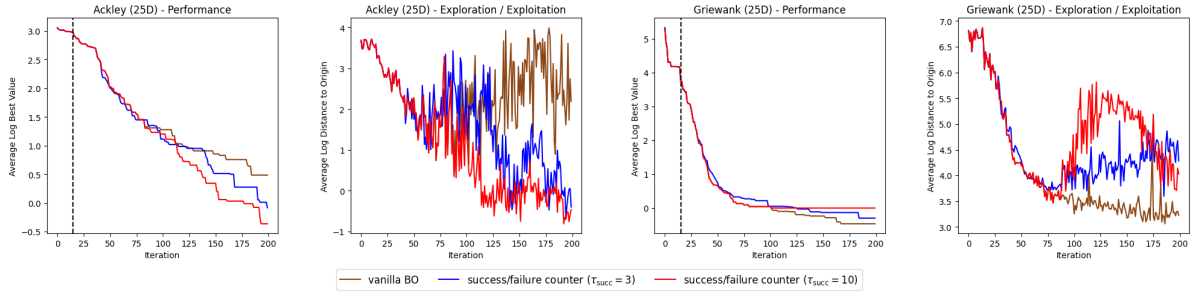


Figure 10: Comparison of performance and exploration/exploitation behavior of success/failure counter (option 3) with different success tolerances, on the Ackley function (left) and Griewank function (right). The plots do not demonstrate a clear advantage of choosing $\tau_{\text{succ}} = 3$ over $\tau_{\text{succ}} = 10$ in terms of performance. However, the ability to recover from over-exploration is more obvious for the latter.

6.2 Potential Future Refinements

We reiterate that none of the methods we have tested consistently outperforms the vanilla BO baseline with the specified settings in Section 5. This may necessitate more systematic and holistic investigations to identify a set of good hyperparameters across problems of varying dimensions.

Moreover, we attempt to identify opportunities to refine these methods in the future. For example, the lengthscales cool down methods explored in this report linearly scale the prior/posterior of lengthscales according to the evolving base length L, which might be somewhat arbitrary. We could potentially explore non-linear methods to shrink the lengthscales, with relatively more impacts on dimensions that are originally identified as inactive.

7 Conclusions

Hvarfner et al. [9] have shown that what essentially hinders vanilla BO in high-dimensional settings is the assumed complexity of the objective, instead of the dimensionality. Hence, they propose DSP which scales the lengthscales prior with increasing dimensionality, assuming the objective function is of low complexity and without tailoring to specific structural assumptions. Building upon this work, we seek to explore potential improvements, such as:

1. Evolving (mainly shrinking) the prior/posterior of lengthscales;
2. Simplifying observations, hence implicitly simplifying the objective.

Both directions attempt to adaptively make the model's assumption and the unknown ground truth objective function more aligned. However, these methods are not off-the-shelf solutions and introduce additional hyperparameters that might be critical to performance to be tuned, for example as the step length for the fixed scheduler, the success/failure tolerance $\tau_{\text{succ}}, \tau_{\text{fail}}$, and k, C for soft Winsorization.

Nevertheless, we still have some observations regarding certain behaviors of these methods, which might be useful in applications or worth further investigation:

1. Lengthscales evolution methods using option 3 generally have stronger impacts on the performance (either positive or negative) compared to methods using option 2;
2. The success/failure counter (option 3) is more likely able to recover from over-exploration to exploitation compared to other lengthscales cool down methods;
3. Soft Winsorization might be better at dealing with the existence of extreme outliers.

References

[1] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization, 2024.

[2] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization, 2020.

[3] Felix Berkenkamp, Angela P. Schoellig, and Andreas Krause. No-regret bayesian optimization with unknown hyperparameters. *Journal of Machine Learning Research*, 20(50):1–24, 2019.

[4] Mickael Binois and Nathan Wycoff. A survey on high-dimensional gaussian process modeling with application to bayesian optimization, 2022.

[5] Adam D. Bull. Convergence rates of efficient global optimization algorithms, 2011.

[6] George De Ath, Richard M. Everson, Alma A. M. Rahat, and Jonathan E. Fieldsend. Greed is good: Exploration and exploitation trade-offs in bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, 1(1):1–22, April 2021.

[7] David Eriksson and Martin Jankowiak. High-dimensional bayesian optimization with sparse axis-aligned subspaces, 2021.

[8] David Eriksson, Michael Pearce, Jacob R Gardner, Ryan Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization, 2020.

[9] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions, 2024.

[10] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001.

[11] Kirthevasan Kandasamy, Jeff Schneider, and Barnabas Poczos. High dimensional bayesian optimisation and bandits via additive models, 2016.

[12] Mario Köppen. The curse of dimensionality. *5th online world conference on soft computing in industrial applications (WSC5)*, 2000.

[13] Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. Constrained bayesian optimization with noisy experiments, 2018.

[14] Ruben Martinez-Cantin, Kevin Tee, and Michael McCourt. Practical bayesian optimization in the presence of outliers, 2017.

[15] J. Mockus, Vytautas Tiesis, and Antanas Zilinskas. *The application of Bayesian methods for seeking the extremum*, volume 2, pages 117–129. 09 2014.

[16] Sarah Müller, Alexander von Rohr, and Sebastian Trimpe. Local policy search with bayesian optimization, 2021.

[17] Santu Rana, Cheng Li, Sunil Gupta, Vu Nguyen, and Svetha Venkatesh. High dimensional Bayesian optimization with elastic Gaussian process. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2883–2891. PMLR, 06–11 Aug 2017.

[18] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005.

[19] Somya Sharma and Snigdhansu Chatterjee. Winsorization for robust bayesian neural networks. *Entropy (Basel)*, 23(11):1546, November 2021.

[20] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[21] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

[22] Kim Peter Wabersich and Marc Toussaint. Advancing bayesian optimization: The mixed-global-local (mgl) kernel and length-scale cool down, 2016.

[23] Ziyu Wang and Nando de Freitas. Theoretical analysis of bayesian optimisation with unknown gaussian process hyper-parameters, 2014.

[24] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings, 2016.

[25] Eric W. Weisstein. Hypercube line picking.

[26] Rand Wilcox. *Trimming and Winsorization*, volume 6. 07 2005.

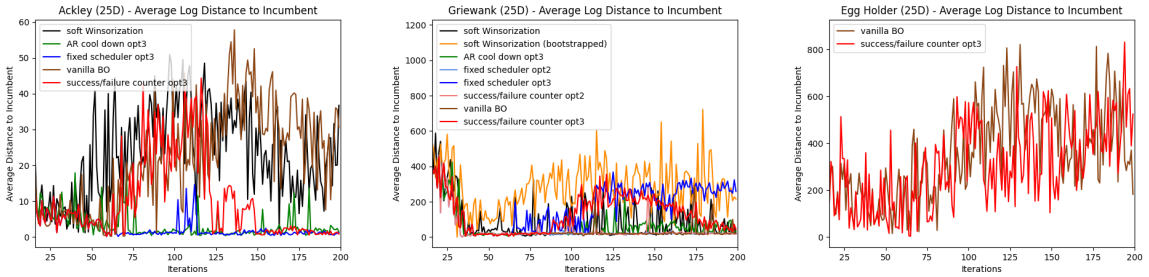
[27] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. In D. Touretzky, M.C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.

[28] Kenan Šehić, Alexandre Gramfort, Joseph Salmon, and Luigi Nardi. Lassobench: A high-dimensional hyperparameter optimization benchmark suite for lasso, 2022.

Appendix

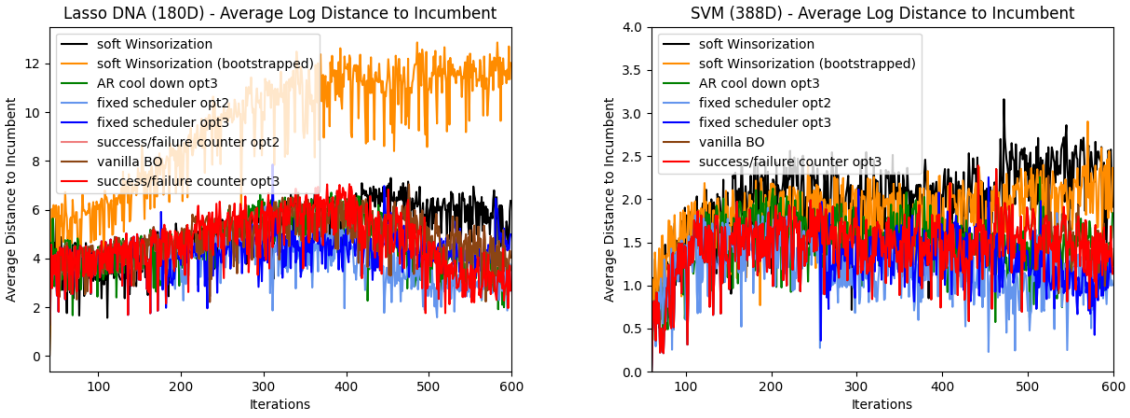
A.1 Exploration/Exploitation Behavior of Different Strategies

Instead of plotting the distance from the queried points to the origin (which only makes sense to the Ackley function and Griewank function), we plot the distance to the incumbent here.



(a) (b) (c)

Figure 11: Average distance of queried points to the incumbent for different methods on the synthetic test functions. The distance is only computed on the important dimensions. For clarity, we remove the plot for some methods.



(a) (b)

Figure 12: Average distance of queried points to the incumbent for different methods on the real-world tasks. The success/failure counter (option 3) has demonstrated the ability to recover from over-exploration in the Lasso DNA task.