Local policy search with Bayesian optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reinforcement learning (RL) aims to find an optimal policy by interaction with an environment. Consequently, learning complex behavior requires a vast number of samples, which can be prohibitive in practice. Nevertheless, instead of systematically reasoning and actively choosing informative samples, policy gradients for local search are often obtained from random perturbations. These random samples yield high variance estimates and hence are sub-optimal in terms of sample complexity. Actively selecting informative samples is at the core of Bayesian optimization, which constructs a probabilistic surrogate of the objective from past samples to reason about informative subsequent ones. In this paper, we propose to join both worlds. We develop an algorithm utilizing a probabilistic model of the objective function and its gradient. Based on the model, the algorithm decides where to query a noisy zeroth-order oracle to improve the gradient estimates. The resulting algorithm is a novel type of policy search method, which we compare to existing black-box algorithms. The comparison reveals improved sample complexity and reduced variance in extensive empirical evaluations on synthetic objectives. Further, we highlight the benefits of active sampling on popular RL benchmarks.

1 Introduction

Reinforcement learning (RL) is a notoriously data-hungry machine learning problem, where state-of-art methods easily require tens of thousands of data points to learn a given task [1]. For every data point, the agent has to carry out potentially complex interactions with its environment, either in simulation or in the physical world. This expensive data collection motivates the development of sample-efficient algorithms. Herein, we consider policy search problems, a type of RL technique where we directly optimize the parameters of a policy with respect to the cumulative reward over a finite episode. The collected data is utilized to estimate the direction of local policy improvement, enabling the use of powerful optimization techniques such as stochastic gradient descent. Policy gradient methods (e.g., [2–5]) usually rely on random perturbations for data generation, e.g., in the form of exploration noise in the action space or stochastic policies, and do not reason about uncertainty in their gradient estimation. However, innate in the RL setting is the ability to actively generate data, allowing the agent to decide on *informative*

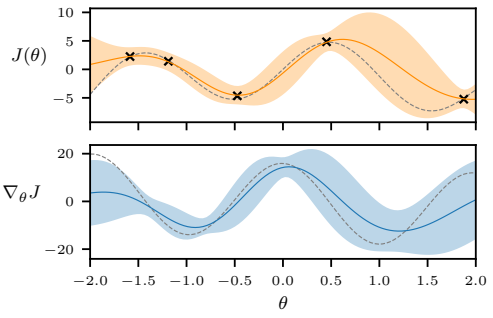

Figure 1: Estimation of a Jacobian GP model (bottom) of a 1-dimensional objective function (top). The model has observations (black crosses) from the function, but is able to form a posterior belief over the gradient. Uncertainty for the Jacobian model is reduced *between* samples. An active sample strategy can improve gradient estimates.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

queries, thereby potentially reducing the amount of data needed to find a (local) optimum. Active sampling has the potential to allow those algorithms to improve sample complexity, reducing the number of environment interactions.

In contrast to random sampling, Bayesian optimization (BO) [6] is a paradigm to optimize expensive-to-evaluate and noisy functions in a sample-efficient manner. At the core of BO is the question of how to query the objective function efficiently to maximize the information contained in each sample. By building a probabilistic model of the objective using past data and, critically, *prior knowledge*, the algorithm can reason about how to query a noisy oracle to solve the optimization task. Since RL can be framed as a black-box optimization problem, we can use BO to learn policies in a sample-efficient way. However, even though BO has been used to tackle RL, these approaches have been restricted to low-dimensional problems. One reason is that BO aims to find a *global* optimum; hence BO algorithms model and search the entire domain, which needs a lot of data and gets exponentially more difficult as the dimensionality increases. Additionally, as the amount of data grows so does the computational complexity of probabilistic models, which becomes a significant problem. However, the success of RL algorithms using policy gradient methods indicates that for many problems it is sufficient to find a locally optimal policy.

Our proposed algorithm combines the strength of gradient-based policy optimization with active sampling in the policy space using BO. We thereby improve the computational complexity of BO methods on the one hand, and the sample-inefficiency of gradient-based methods on the other hand, especially when proper prior knowledge is available. We achieve these improvements by explicitly learning a probabilistic model of the objective in the form of a Gaussian process (GP) [7]. From this model, we can jointly infer the objective and its gradients with a tractable probabilistic posterior. The resulting Jacobian estimate includes all data points, rendering data usage more efficient. Further, the algorithm infers informative queries from the uncertainty of the probabilistic model to improve the estimate of the local gradient. While in this paper we adapt the setting of [1] and assume access to zeroth-order information only, the algorithm extends straightforwardly to policy gradient algorithms where additional first-order information is available. In summary, the contribution of this paper is a local BO-like optimizer called *Gradient Information with BO* (GIBO). The queries of GIBO are chosen optimally to minimize uncertainty in the gradient estimation. GIBO can be used with existing policy search algorithms to improve gradient estimates. Using only zeroth order information, GIBO is able to

- significantly improves sample complexity in extensive within-model comparisons, i.e., when accurate prior knowledge is available;
- is able to solve RL benchmark problems in a sample efficient manner; and
- reduces variance in the results when compared to non-active sampling baselines.

1.1 Related Work

This section relates our contribution to the literature on BO for RL, on BO using gradient information, and how GIBO can be incorporated into existing policy gradient methods.

BO and RL. Bayesian optimization has been used as a global optimizer to solve RL tasks in prior work [8–12]. However, the mentioned methods usually search for a global optimum in 2- to 15-dimensional parameter spaces. Global BO for RL, exemplified by the mentioned literature, is limited to relatively low dimensional problems for two reasons: (i) the computational complexity of global probabilistic models does not scale well with the number of data points, (ii) global optimization of high-dimensional non-convex objectives is a challenging problem to solve in general. In contrast to these prior works on BO in RL, we improve sample complexity by leveraging gradient-based optimization. We reduce the problem to a local optimization problem, which entails that we only need a local GP model. A local model is computationally easier to handle and ships with the additional benefits we discuss in Sec. 3.3.

BO with first-order oracles. In general, a GP posterior can incorporate gradient information if the kernel is differentiable and a first-order oracle is available. Bayesian optimization methods that utilize gradient observations are known as first-order BO, and different approaches on how to include the derivative information in the model and acquisition functions have been proposed [13–16]. Since computing the joint posterior using first- and zeroth-order information is computationally expensive, [14] and [15] are using a single directional derivative instead of all partial derivatives. A first-order

BO approach for RL, where the gradient information is actively used to decide on the following query, is introduced by Prabuchandran et al. [16]. The method therein actively searches for local optima by querying points where the gradient is expected to be zero. In contrast, we use the probabilistic model of the Jacobian and a BO-style active decision making. GIBO queries the oracle to gain information about the gradient at the current location and afterwards a gradient-based optimizer decides on the next location. The work closest to ours was proposed recently by Shekhar and Javidi [17]. They propose an algorithm with access to zeroth and first-order oracles and derive improved regret bounds compared to the zeroth-order oracle case. Their algorithm is divided into two phases: In the first phase, it finds a near-optimal region via global optimization and proceeds to phase two, aiming towards the optimum with stochastic gradient descent. Contrary to GIBO, in the second phase the algorithm therein does not optimize the queries for gradient information. Instead it relies on repeated queries to the first-order oracle at the same location to reduce uncertainty. Additionally, these first-order BO methods all rely on gradient *observations*, while the proposed GIBO algorithm can work on zeroth-order queries alone, reducing computational complexity significantly.

Informative sampling in policy gradient methods. Policies that generate more informative samples have helped to improve model-free RL algorithms' performance during the past decade; we mention three examples here. Levine and Koltun [18] propose so-called guiding samples in high reward areas using differential dynamic programming and model knowledge. Soft actor-critic (SAC) methods [3] add the policy's entropy to the reward function to encourage exploration and improve the variance of gradient estimates. Based on SAC an optimistic actor-critic algorithm is introduced in [19] with a different exploration strategy that samples more informative actions. Differing from typical policy search methods exemplified above, we propose a probabilistic model of the objective function that enables active sampling and exploits all available information. It is possible to use GIBO as a layer between the policy gradient estimator such as SAC and a gradient-based optimizer, e.g., stochastic gradient ascent or Adam [20]. GIBO can utilize any policy gradient algorithm as an oracle for zeroth- or first-order information. Based on the posterior conditioned on all collected rewards, our algorithm can supply posterior gradient estimates and subsequent queries to evaluate.

To demonstrate the benefits of active sampling in a simple setup, we adopt the setting proposed by Mania et al. [1] where policy optimization is treated as a black-box problem. Augmented Random Search (ARS) [1] bases on finite-difference gradient estimation and has been shown to solve RL tasks, when no information about the gradient is available. We replace the random sampling strategy of ARS with active sampling and the gradient estimation with a GP model. These changes improve the sample complexity and variance of ARS, especially when prior knowledge about the objective function is available.

2 Preliminaries

This work presents a local optimizer with active sampling. The objective function and its derivative's joint distribution are modeled using a GP. Since we have developed the optimizer with the RL application in mind, we also introduce the RL problem. For the sake of brevity, we refer the reader to [7] and [21] for a GP and BO introduction, respectively.

2.1 Problem setting

In the following we phrase policy search as a black-box optimization problem. For a parameterized policy $\pi_\theta : \Theta \times \mathcal{S} \to \mathcal{A}$ that maps states $s \in \mathcal{S}$ and the static policy parameters $\theta \in \Theta$ to actions $a \in \mathcal{A}$, we use the same performance measure as in policy gradient methods for the episodic case. Hence, the objective function $J : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{i=0}^{I} r_i\right],$$

where $\mathbb{E}_{\pi_\theta}$ is the expectation under policy π_θ, r_i is the reward at time step i and I the length of the episode. A BO query is equivalent to the return of one rollout of the policy π_θ in the environment. The expected episodic reward is entirely determined by choice of policy parameters (and the initial conditions). Thus, the optimizer explores the reward function in the parameter space rather than in the action space. Since initial conditions might vary and the environment can be non-deterministic, reward evaluations are noisy.

Policy search herein is abstracted as a zeroth-order optimization problem of the form

$$\theta^* = \arg\max_{\theta \in \Theta} J(\theta), \tag{1}$$

where θ is the variable and $\Theta \subset \mathbb{R}^d$ a bounded set. To solve (1) an optimization algorithm can query an oracle for a noisy function evaluation $y = J(\theta) + \omega$. We assume an i.i.d. noise variable $\omega \in \mathbb{R}$ to follow a normal distribution $\omega \sim \mathcal{N}(0, \sigma^2)$ with variance σ^2. We do not assume access to gradient information or other higher-order oracles for conciseness. Albeit, GIBO requires that the following critical assumption is fulfilled:

Assumption 1. *The objective function J is a sample from a known GP prior $J \sim GP(m(\theta), k(\theta, \theta'))$, where the mean function is at least once differentiable and the covariance function k is at least twice differentiable, w.r.t. θ.*

This is the standard setting for BO with the addition that the mean and kernel need to be differentiable, which is satisfied by some of the most common kernels such as the squared exponential (SE) kernel. In the empirical section, we investigate the performance of the developed algorithm with and without Assumption 1 holding true.

2.2 Jacobian GP model

Since GPs are closed under linear operations, the derivative of a GP is again a GP [7]. This enables us to derive an analytical distribution for the objective's Jacobian, which we can use as a proxy for gradient estimates and enable gradient-based optimization.

Following Rasmussen and Williams [7], the joint distribution between a GP and its derivative at the point θ_* is

$$\begin{bmatrix} \bar{y} \\ \nabla_{\theta_*} J_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(X) \\ \nabla_{\theta_*} m(\theta_*) \end{bmatrix}, \begin{bmatrix} K(X,X) + \sigma^2 I & \nabla_{\theta_*} K(X, \theta_*) \\ \nabla_{\theta_*} K(\theta_*, X) & \nabla^2_{\theta_*} K(\theta_*, \theta_*) \end{bmatrix} \right), \tag{2}$$

where $\bar{y}$ are the n zeroth-order observations, $X \subset \Theta$ are the locations of these observations $X = [\theta_1, \dots, \theta_n]$, and K the covariance matrix given by the kernel function $k : \Theta \times \Theta \to \mathbb{R}$. The posterior can be derived by conditioning the joint Gaussian prior distribution on the observation [7]

$$\begin{aligned} p\left(\nabla_{\theta_*} J_* \middle| \theta_*, X, \bar{y}\right) &\sim \mathcal{N}(\mu'_*, \Sigma'_*) \\ \mu'_* &= \underbrace{\nabla_{\theta_*} m(\theta_*)}_{\in \mathbb{R}^d} + \underbrace{\nabla_{\theta_*} K(\theta_*, X)}_{\in \mathbb{R}^{d \times n}} \underbrace{\left(K(X,X) + \sigma^2 I\right)^{-1}}_{\in \mathbb{R}^{n \times n}} \underbrace{(\bar{y} - m(X))}_{\in \mathbb{R}^n} \in \mathbb{R}^d \\ \Sigma'_* &= \underbrace{\nabla^2_{x_*} K(\theta_*, \theta_*)}_{\in \mathbb{R}^{d \times d}} - \underbrace{\nabla_{\theta_*} K(\theta_*, X)}_{\in \mathbb{R}^{d \times n}} \underbrace{\left(K(X,X) + \sigma^2 I\right)^{-1}}_{\in \mathbb{R}^{n \times n}} \underbrace{\nabla_{\theta_*} K(X, \theta_*)}_{\in \mathbb{R}^{n \times d}} \in \mathbb{R}^{d \times d}. \end{aligned} \tag{3}$$

Remark 1. *Note that the term $[K(X,X) + \sigma^2 I]^{-1}$ with the highest computational cost ($\mathcal{O}(N^3)$) is the same term that is used to compute the posterior over f. Therefore, calculating the Jacobian does not add to the computational complexity once a GP posterior has been computed.*

Any twice differentiable kernel is sufficient for the presented framework, but we assume a SE kernel for the remainder of the paper. The derivatives of the SE kernel function are in the appendix Sec. A.1. For a visual example of function- and the Jacobian-posterior, refer to Fig. 1. The figure indicates that a zeroth-order oracle is enough to form a reasonable belief over the function's gradient. Moreover, Fig. 1 shows that the uncertainty about the Jacobian gets reduced *between* query points more so than at the query points themselves. To minimize uncertainty about the Jacobian at a specific point, it intuitively makes sense to space out query points in its immediate surrounding. Herein, we formalize this intuition and formulate an optimization problem that sequentially decides on query points that provide the most information about the Jacobian.

3 Gradient Informative Bayesian Optimization

In this section, we introduce the proposed method GIBO. First, we define an acquisition function to reduce uncertainty for the Jacobian. Second, we outline the basic GIBO algorithm, including extensions.

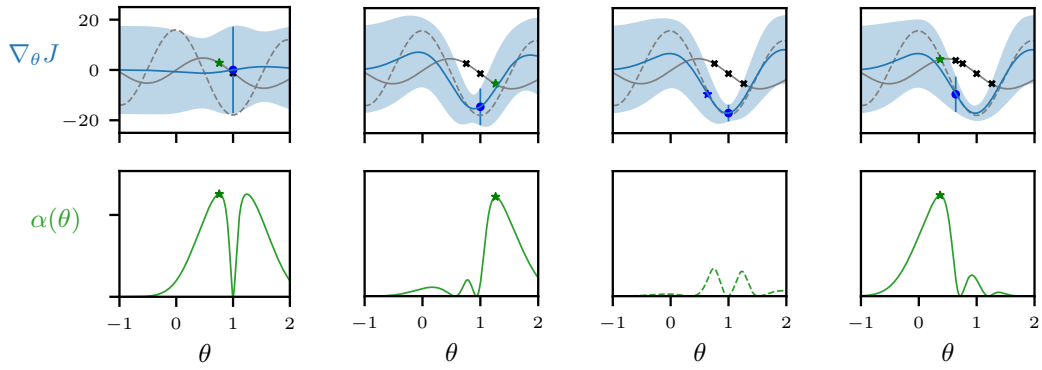

Figure 2: We visualize GIBO's active sampling process with a simple 1-dimensional function. The blue filled circle refers to the current parameter θ_t. In the first two images, the acquisition function α (solid green line) proposes two new query points (green stars) of the objective function J (solid light grey line). With the history of sampling points (black crosses), the model of the Jacobian $\nabla_\theta J$ (in blue with confidence intervals) is updated, reducing uncertainty around the analytic Jacobian (dashed light grey line). The last two images show a gradient ascent update step to θ_{t+1} (blue star).

3.1 Maximizing gradient information

We employ the BO framework to design a set of iterative queries maximizing gradient information. To this extend, we propose a novel acquisition function *Gradient Information (GI)* actively suggesting query points most informative for the gradient at the current parameters θ_t. Acquisition functions measure the expected utility of a sample point based on a surrogate model conditioned on the observed data. The utility $U : \mathbb{R}^d \to \mathbb{R}$ of our method depends on a Jacobian GP model, the objective's observation data $\mathcal{D}$, and the current parameter θ_t. It measures the decrease in the derivative's variance at θ_t when observing a new point θ of the objective function. Hence, we define the utility as the expected difference between the Jacobian's variance $\Sigma'(\theta_t|\mathcal{D})$ *before* and the Jacobian's variance $\Sigma'(\theta_t|\{\mathcal{D}, (\theta, y)\})$ *after* observing a new point (θ, y)

$$\alpha_{\text{GI}}(\theta|\theta_t, \mathcal{D}) = \mathbb{E}\left[U(\theta|\theta_t, \mathcal{D})\right] = \mathbb{E}\left[\text{Tr}\left(\Sigma'(\theta_t|\mathcal{D})\right) - \text{Tr}\left(\Sigma'\left(\theta_t|\left\{\mathcal{D}, (\theta, y)\right\}\right)\right)\right], \tag{4}$$

where Tr denotes the trace operator and $\Sigma'(\theta_t|\mathcal{D})$ is the variance of the Jacobian's GP model evaluated at θ_t

$$\nabla_\theta J\big|_{\theta=\theta_t} \sim \mathcal{GP}\left(\mu'(\theta_t|\mathcal{D}), \Sigma'(\theta_t|\mathcal{D})\right). \tag{5}$$

The Jacobian's variance $\Sigma'(\theta_t|\{\mathcal{D}, (\theta, y)\})$ depends on the extended dataset $\{\mathcal{D}, (\theta, y)\}$. A property of the Gaussian distribution is, that the covariance function is independent of the observed targets y as shown in Equation (3). Hence, we simplify the optimization over the expectation to (see Appendix A.2)

$$\arg\max_\theta \alpha_{\text{GI}}(\theta|\theta_t, \mathcal{D}) = \arg\min_\theta \text{Tr}\left(\Sigma'\left(\theta_t|\left[X, \theta\right]\right)\right), \tag{6}$$

where the variance only depends on a virtual data set $\hat{X} = [\theta_1, \ldots, \theta_n, \theta] =: [X, \theta]$. In conclusion, the most informative new parameter θ to query is only dependent on *where* we sample next and is independent of its outcome $f(\theta) = y$.

When we replace the Jacobian's variance in (6) with (3) and leave out constant factors we get

$$\theta^* = \arg\max_\theta \text{Tr}\left(\nabla_{\theta_t} K(\theta_t, \hat{X}) \left(K(\hat{X}, \hat{X}) + \sigma_n^2 I\right)^{-1} \left(\nabla_{\theta_t} K(\theta_t, \hat{X})\right)^T\right). \tag{7}$$

Since the acquisition function only depends on the virtual data set, the optimization of the acquisition function can be handled computationally efficient by performing the matrix inversion in (7) using Cholesky factor updates. This method is outlined in Appendix A.3.

3.2 The GIBO algorithm

The guided sequential search of the acquisition function for gradient estimates divides the resulting algorithm into two loops: An outer loop for iterative parameter updates and an inner loop where the acquisition function queries points to increase gradient information. The basic algorithm is given in Alg. 1.

Algorithm 1 GIBO

Hyperparameters: stepsize η, hyperpriors for GP hyperparameters, number of iterations N and M samples for a gradient estimate.
Initialize: place a GP prior on $J(\theta)$, set θ_0 and $\mathcal{D} = \{\}$.
for $t = 0, \dots, N$ **do** ▷ Parameter updates.
  Sample noisy objective function: $y_t = J(\theta_t) + \epsilon_t$
  Extend data set: $\mathcal{D} \leftarrow \{\mathcal{D}, (\theta_t, y_t)\}$
  GP hyperparameter optimization.
  for $m = 1, 2, \dots, M$ **do** ▷ Sample points for a gradient estimate.
    Get query point: $\hat{\theta} = \arg\max_{\hat{\theta}} \alpha_{\text{GI}}(\hat{\theta}|\theta_t, \mathcal{D})$.
    Sample noisy objective function: $\hat{y} = J(\hat{\theta}) + \omega$. ▷ Optionally: Use a policy gradient method for additional derivative observations.
    Extend data set: $\mathcal{D} \leftarrow \{\mathcal{D}, (\hat{\theta}, \hat{y})\}$.
    Update the posterior probability distribution of $\nabla_\theta J$.
  end for
  $\theta_{t+1} = \theta_t + \eta \cdot \mathbb{E}\left[\nabla_\theta J\big|_{\theta=\theta_t}\right]$ ▷ Gradient ascent, or any other gradient based optimizer.
end for

3.3 Extensions

In the following, algorithmic extensions are introduced that further improve the performance and computational efficiency of our method.

Local GP model. Sparse approximation of GPs can be applied on BO when the computational burden of exact inference is too big [22]. In our case, however, we are only interested in estimating the local Jacobian at the current parameter θ_t. We define a sparse approximation of the posterior at the current parameter θ_t heuristically with the last N_m sampled points. Estimating a local model has the additional benefit of making the model selection and hyperparameter optimization simpler. We can approximate non-stationary processes locally by dynamically adapting hyperparameters.

Local optimization of GI. Following similar reasoning as above, we do not have to optimize the GI acquisition function globally since we expect informative points to be relatively close to the current parameter θ_t when using a SE kernel. Hence, we define our search bounds locally as $[\theta_t - \delta_b, \theta_t + \delta_b]$.

Uncertainty threshold. We can stop sampling new points once our uncertainty about the gradient is small enough, and another point would not provide a significant information gain. If $\Sigma'(\theta_t) - \Sigma'(\hat{\theta}) < \epsilon_{\text{a}}$ we stop sampling, where ϵ_{a} is a hyperparameter of the algorithm.

Gradient normalization. The gradient is normalized with the Mahalanobis norm using the length-scales of the SE kernel.Hence, the stepsize η is adapted automatically to scale with the correlation between points, as explained in Appendix A.4.

State normalization. We apply state normalization to policy search for RL environments. This has the same effect as data whitening for regression tasks. In practice, this is beneficial to perform GP regression for unknown policy spaces. In case of a linear policy $\pi_\theta : \mathbb{R}^p \to \mathbb{R}^m$, $\pi_\theta(s) = As + b$ with bias $b \in \mathbb{R}^m$, states $s \in \mathbb{R}^p$, means of states $\mu_s \in \mathbb{R}^p$ and variances $\sigma_s \in \mathbb{R}^p$ of states, state normalization can be defined by $\pi_\theta\left(\frac{s-\mu_s}{\sigma_s}\right) = A\left(\frac{s-\mu_s}{\sigma_s}\right) + b = A \cdot \frac{1}{\sigma_s} s - A \cdot \frac{\mu_s}{\sigma_s} + b$. The state normalization is implemented in an efficient way that does not require the storage of all states. Also, we only keep track of the diagonal of the state's covariance matrix with Welford's online algorithm [23].

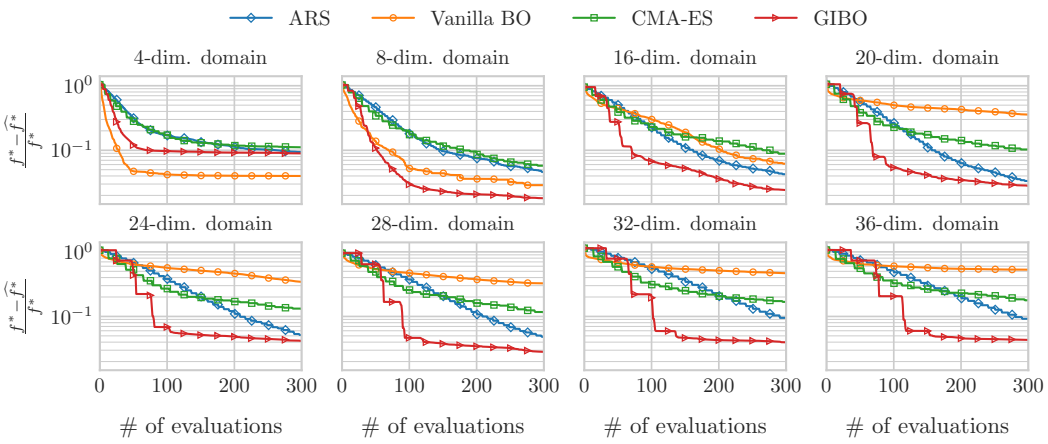

Figure 3: Within-model comparison: Normalized distance of function value at optimizers' best guesses from the true global maximum for eight different dimensional function domains. Logarithmic scale.

4 Empirical Results

We empirically evaluate the performance of GIBO in three types of experiments. In the first experiment, we compare our algorithm on several functions sampled from a GP prior so that Assumption 1 is satisfied. In these *within-model comparisons* [24], we can show that GIBO outperforms the benchmark methods in terms of sample complexity and variance of regret, especially in higher dimension. In a second experiment, we perform policy search for a linear quadratic regulator (LQR) problem proposed by Mania et al. [1]. Finally, in policy search for RL environments of Gym [25] and MuJoCo [26], we show that GIBO reaches acceptable rewards thresholds faster and with significantly less variance than ARS. All data and source code necessary to reproduce the results are published anonymized at `https://github.com/gibo-neurips-2021/GIBO`.

4.1 Within-model Comparison

We evaluate GIBOs performance as a general black-box optimizer on functions that satisfy Assumption 1. A straightforward way to guarantee this is by sampling the objective from a known GP prior. This approach has been called within-model comparison by Hennig and Schuler [24] but has likewise been used in other BO literature (e.g., [27, 28]). To show that GIBO scales particularly well to higher-dimensional search spaces, we analyze synthetic benchmarks for up to 36 dimensions.

The experiment was carried out over a d-dimensional unit domain $I = [0, 1]^d$. For each domain, we generate 40 different test functions. For each function, 1000 values were jointly sampled from a GP prior with a SE kernel and unit signal variance. To cover the space evenly, we used a quasi-random Sobol sampler. To perform experiments with comparable difficulty across different dimensional domains, we increase the lengthscales in higher dimensions by sampling them from the distribution $\ell(d)$, introduced in Appendix A.5. The resulting posterior mean was the objective function. All algorithms were started in the middle of the domain $x_0 = [0.5]^d$ and had a limited budget of 300 zero-order noised function evaluations. The noise was Gaussian distributed with standard deviation $\sigma = 0.1$. A more detailed description of the experiments, including the true global maximum search and an out-of-model comparison, is given in Appendix A.5.

We compared our algorithm GIBO to ARS, CMA-ES [29] and standard BO with expected improvement [30] as acquisition function ('Vanilla BO'). To ensure a fair comparison, domain knowledge was passed to the ARS and CMA-ES algorithms by scaling the space-dependent hyperparameters with the mean of the lengthscale distribution $\ell(d)$. For details about the hyperparameters see Appendix A.9.

Fig. 3 shows the normalized difference between the global function optimum and the function values of the optimizer's best guesses. The within-model comparison shows that our algorithm outperforms

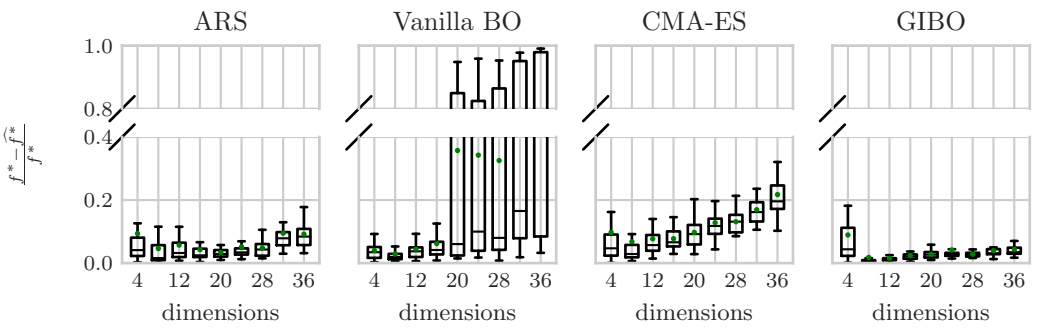

Figure 4: Within-model comparison: Boxplots show the normalized distance of optimizers' best found values after 300 function evaluations from the true global maximum. The whiskers lengths are 1.5 of the interquartile range; the black horizontal lines represent medians, green dots the means.

vanilla BO on all test functions, except for the 4-dimensional domain. The proposed method GIBO outperforms the other benchmarks in terms of sample complexity, especially in higher dimensions. Further, GIBO is able to reduce the variance of obtained regret significantly, as shown in Figure 4, which indicates a consistently better performance.

4.2 Linear Quadratic Regulator

The classic LQR with known dynamics is a fundamental problem in control theory. In this setting, an agent seeks to control a linear dynamical system while minimizing a quadratic cost. With available dynamics, the LQR problem has an efficiently determinable optimal solution. LQR with unknown dynamics, on the other hand, is less well understood. As argued in Mania et al. [1], this offers a new type of benchmark problems, where one can set up LQR problems with challenging dynamics, and compare model-free methods to known optimal costs. We compare GIBO against ARS and LSPI [31] on a challenging LQR instance with unknown dynamics, known from [32, 1, 31]. The reader is referred to Appendix A.6 for a complete introduction to the setup.

The Fig. 5 shows the frequency of stable controllers found and the cost compared to the optimal cost for GIBO, ARS, and LSPI. On the left in Fig. 5 we observe that GIBO requires significantly fewer samples than ARS, equivalent to LSPI, to find a stabilizing controller. But we note that LSPI requires an initial controller K_0, which stabilizes a discounted version of the LQR problem. Neither GIBO nor ARS require any special initialization. All algorithms achieve similar regrets.

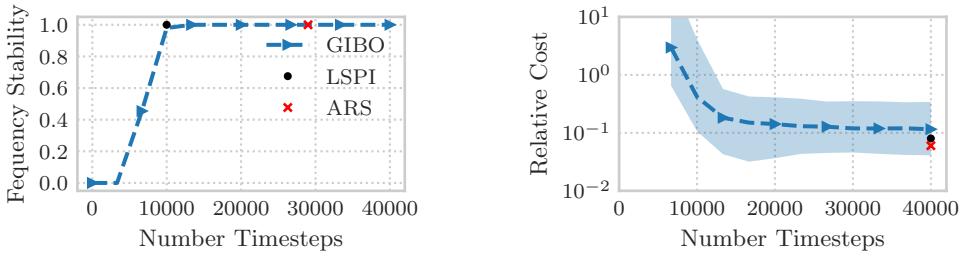

Figure 5: Results for the LQR experiment. Left: How frequently GIBO found stabilizing controllers in comparison to ARS and LSPI. The frequencies are estimated from 100 trials. Right: The sub-optimality gap of the controllers produced by GIBO compared to ARS and LSPI. The points along the dashed line denote the median cost, and the shaded region covers 2-nd to 98-th percentile out of 100 trials. Values for the benchmark methods in both images are estimated from [1].

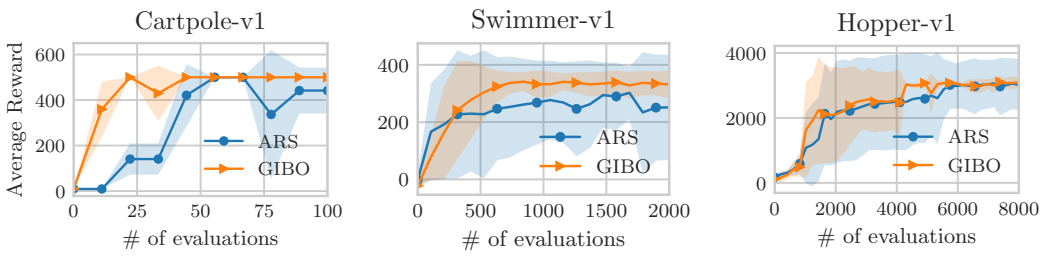

Figure 6: Training curves of GIBO and ARS for classic control and MuJoCo tasks, averaged over 3 trails. The shaded regions show the standard deviation.

4.3 Gym and MuJoCo

Lastly, we evaluate the performance of GIBO on classic control and MuJoCo tasks included in the OpenAI Gym [25, 26]. The OpenAI Gym provides benchmark reward functions that we use to evaluate our policies' performance compared to policies trained by ARS. Mania et al. [1] showed that deterministic linear policies, $\pi_\theta : \mathbb{R}^p \to \mathbb{R}^m$, $\pi_\theta(s) = As + b$, are sufficiently expressive for MuJoCo locomotion tasks. Consequently, we define our parameter space by $\theta = (A, b) \in \mathbb{R}^{p \times m+m}$. For the CartPole-v1 we need 4, for the Swimmer-v1 16 and for the Hopper-v1 36-dimensions. For all environments, we normalize the reward axis. For the Hopper environment, we additionally subtract the survival bonus and use state normalization; find further details in Appendix A.7.

In the following, we plot the reward against the number of function evaluations (calls of RL environment). We averaged the reported policy rewards over three trials. In Fig. 6 we observe that GIBO reaches the reward thresholds faster and with significantly less variance than ARS.

5 Conclusion

We introduce GIBO, a gradient-based optimization algorithm with a BO-type active sampling strategy to improve gradient estimates for black-box optimization problems. When the model assumptions of BO are satisfied, we show that the algorithm is significantly more sample-efficient, especially in higher dimensions, compared to baseline algorithms for black-box optimization.

Additionally, we show the benefits of active sampling and probabilistic gradient estimates with GIBO by solving popular RL benchmarks for which the model assumptions do not hold exactly. When compared to random sampling, GIBO is still more sample efficient and has lower variance. Yet, the performance benefits are less pronounced in the RL task. This highlights that GIBO especially shines when prior knowledge is available while it still performs reasonably otherwise. Nonetheless, we want to remark that the prior biases the gradient estimates and wrong assumptions about the objective function can deteriorate performance. However, in some sense, all hyperparameters in RL algorithms encode some form of prior knowledge about the problem at hand. In our view, explicit probabilistic priors are an appropriate and intuitive form of prior knowledge to obtain, e.g., from domain knowledge or available data from prior experiments.

Since it is straightforward to include derivative observations into GIBO, we expect similar improvements for other existing RL methods when integrating our method as an additional layer between gradient estimators and optimizers. The proposed framework can suggest different exploration policies and combine all available data into a posterior belief over the Jacobian. For future research, we want to utilize GIBO with state-of-the-art actor-critic algorithms to improve sample complexity of these methods.

In a more general context, our active sampling methodology makes a step towards autonomous decision-making. GIBO decides on a learning experiment for the autonomous agent. Whenever a decision process is automated, the responsibility for legal and ethical consequences of these decisions must be resolved. However, we do not discuss how the decision-maker, GIBO, can be constrained to ensure compliance with regulatory requirements, which is a relevant aspect for future research.

References

[1] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31*, pages 1800–1809. 2018.

[2] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[4] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[5] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[6] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[7] Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[8] Daniel J. Lizotte, Tao Wang, Michael H. Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *International Joint Conferences on Artificial Intelligence*, volume 7, pages 944–949, 2007.

[9] Aaron Wilson, Alan Fern, and Prasad Tadepalli. Using Trajectory Data to Improve Bayesian Optimization for Reinforcement Learning. *Journal of Machine Learning Research*, pages 253–282, 2014.

[10] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe. Automatic LQR tuning based on Gaussian process global optimization. In *IEEE International Conference on Robotics and Automation*, pages 270–277, 2016.

[11] Ruben Martinez-Cantin. Bayesian optimization with adaptive kernels for robot control. In *IEEE International Conference on Robotics and Automation*, pages 3350–3356, 2017.

[12] Alexander von Rohr, Sebastian Trimpe, Alonso Marco, Peer Fischer, and Stefano Palagi. Gait learning for soft microrobots controlled by light fields. In *International Conference on Intelligent Robots and Systems*, pages 6199–6206, 2018.

[13] Michael A. Osborne, Roman Garnett, and Stephen J. Roberts. Gaussian processes for global optimization. In *3rd International Conference on Learning and Intelligent Optimization*, pages 1–15, 2009.

[14] Mohamed O. Ahmed, Bobak Shahriari, and Mark Schmidt. Do we need "harmless" bayesian optimization and "first-order" bayesian optimization. In *NeurIPS Workshop on Bayesian Optimization*, 2016.

[15] Jian Wu, Matthias Poloczek, Andrew G. Wilson, and Peter Frazier. Bayesian Optimization with Gradients. In *Advances in Neural Information Processing Systems*, pages 5267–5278, 2017.

[16] K. J. Prabuchandran, Santosh Penubothula, Chandramouli Kamanchi, and S. Bhatnagar. Novel First Order Bayesian Optimization with an Application to Reinforcement Learning. *Applied Intelligence*, pages 1565–1579, 2021.

[17] Shubhanshu Shekhar and Tara Javidi. Significance of gradient information in bayesian optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2836–2844. PMLR, 2021.

[18] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9. PMLR, 2013.

[19] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, 2019.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations*, 2015.

[21] 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*, pages 148–175, 2016.

[22] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. Sparse gaussian processes for bayesian optimization. In *Thirty-Second Conference on Uncertainty in Artificial Intelligence*, page 517–526, 2016.

[23] B. P. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, pages 419–420, 1962.

[24] Philipp Hennig and Christian J. Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, pages 1809 – 1837, 2012.

[25] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[26] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[27] José Miguel Hernández-Lobato, Michael A. Gelbart, Ryan P. Adams, Matthew W. Hoffman, and Zoubin Ghahramani. A General Framework for Constrained Bayesian Optimization using Information-based Search. *Journal of Machine Learning Research*, pages 1–53, 2016.

[28] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient Bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3627–3635, 2017.

[29] Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, pages 159–195, 2001.

[30] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, pages 345–383, 2001.

[31] Stephen Tu and Benjamin Recht. Least-squares temporal difference learning for the linear quadratic regulator. In *International Conference on Machine Learning*, pages 5005–5014. PMLR, 2018.

[32] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the Sample Complexity of the Linear Quadratic Regulator. *Foundations of Computational Mathematics*, pages 633–679, 2020.

[33] Michael A. Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimization and Quadrature*. PhD thesis, Oxford University, UK, 2010.

[34] Geoffrey Hinton. Lecture: Neural Networks for Machine Learning, 2012. `https://www.cs.toronto.edu/~hinton/nntut.html`.

[35] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, pages 533–536, 1986.

[36] John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, pages 2121–2159, November 2011.

[37] R. S. Anderssen, R. P. Brent, D. J. Daley, and P. A. P. Moran. Concerning $\int_0^1 \cdots \int_0^1 \left(x_1^2 + \cdots + x_k^2\right)^{1/2} dx_1 \ldots dx_k$ and a Taylor Series Method. *SIAM Journal on Applied Mathematics*, pages 22–30, 1976.

[38] Stephen Tu. Sample Complexity Bounds for the Linear Quadratic Regulator. Technical Report UCB/EECS-2019-42, University of California at Berkeley, 2019.

[39] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[40] 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. In *Advances in Neural Information Processing Systems 33*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We point out during the paper and especially in the conclusion Sec. 5 that the performance of GIBO hinges on prior knowledge of the objective function in the form of GP hyperparamters.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Our work doesn't describe a specific application, but a general optimization algorithm. We shortly discuss automated decision making in the conclusion Sec. 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] `https://github.com/gibo-neurips-2021/GIBO.git`
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix A.9.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec. 4 for the variance after 300 query points. We choose not to plot error bars in the learning curves for readability reasons. Since the results are plotted on a log scale the error bars are hard to read.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] Nothing fancy, virtual machine of internal cluster, 16 GB RAM, 8 VCPUs, total disk 160 GB, no GPU.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use the OpenAI Gym environments as well as the MuJoCo engine.
   (b) Did you mention the license of the assets? [Yes] See Appendix A.8.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]