Applied Intelligence (2021) 51:1565–1579
https://doi.org/10.1007/s10489-020-01896-w

# Novel First Order Bayesian Optimization with an Application to Reinforcement Learning

Prabuchandran K. J.[1] · Santosh Penubothula[2] · Chandramouli Kamanchi[3] · Shalabh Bhatnagar[3]

Published online: 30 September 2020

**Abstract**
Zeroth Order Bayesian Optimization (ZOBO) methods optimize an unknown function based on its black-box evaluations at the query locations. Unlike most optimization procedures, ZOBO methods fail to utilize gradient information even when it is available. On the other hand, First Order Bayesian Optimization (FOBO) methods exploit the available gradient information to arrive at better solutions faster. However, the existing FOBO methods do not utilize a crucial information that the gradient is zero at the optima. Further, the inherent sequential nature of the FOBO methods incur high computational cost limiting their wide applicability. To alleviate the aforementioned difficulties of FOBO methods, we propose a relaxed statistical model to leverage the gradient information that directly searches for points where gradient vanishes. To accomplish this, we develop novel acquisition algorithms that search for global optima effectively. Unlike the existing FOBO methods, the proposed methods are parallelizable. Through extensive experimentation on standard test functions, we compare the performance of our methods over the existing methods. Furthermore, we explore an application of the proposed FOBO methods in the context of policy gradient reinforcement learning.

**Keywords** Bayesian optimization · Reinforcement learning · First order methods · Policy gradient

## 1 Introduction

Bayesian Optimization (BO) is a sample efficient global optimization paradigm to optimize expensive real-valued functions (the effort required in terms of time/resources to evaluate the function is expensive, see [12]). BO assumes black-box optimization setup, i.e., the analytical form of the function is unknown and only the function evaluations at query points are available (Fig. 1).

The goal of BO is to obtain the global maximum of the function $f$ by implanting minimal number of queries in its domain. In recent times, BO has been applied with great success in machine learning (ML) for automatic tuning of model hyperparameters [33], in robotics for optimizing gait [6, 16] and path planning [20] and in the context of reinforcement learning (RL) [7] for episodic tasks [19, 39].

BO methods search towards global maxima by utilizing two key components namely: a statistical model and an acquisition algorithm (or acquisition/utility function [4]). The statistical model is used for representing the unknown objective function $f(\cdot)$ based on the function evaluations (or samples) observed so far. Gaussian Process (GP) [31] is the statistical model to represent $f(\cdot)$. The acquisition algorithm then utilizes the constructed statistical model to decide where to query the objective function in order to build a better statistical model for future acquisition. As the objective function is queried on its domain based on the point suggested by the acquisition algorithm, a new sample value is observed and thus gets included to the set of so

Prabuchandran K. J. was supported by SGNF research grant from Indian Institute of Technology, Dharwad.

✉ Prabuchandran K. J.
prabukj@iitdh.ac.in

Santosh Penubothula
sapenubo@in.ibm.com

Chandramouli Kamanchi
chandramouli@iisc.ac.in

Shalabh Bhatnagar
shalabh@iisc.ac.in

1 Department of Computer Science and Engineering, Indian Institute of Technology, Dharwad, India

2 IBM Research, Bangalore, India

3 Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

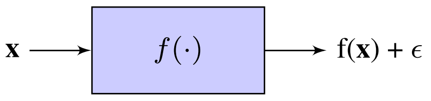


**Fig. 1** Blackbox setup: $\epsilon$ corresponds to noise

far observed samples. Based on the updated set of samples, the statistical model (GP) gets updated (known as posterior Bayesian update). All BO methods have the same posterior update step and differ in the type of acquisition algorithm used to acquire new points. Acquisition algorithms like Maximum Probability of Improvement (MPI), Maximum Expected Improvement (**EI**) [12], Upper Confidence Bound (UCB) [34], Predictive Entropy Search [11] and Knowledge Gradient (**KG**) [9] have been employed to explore the most likely region of maxima. However, none of these algorithms exploit the gradient information in performing the search when it is available.

Recent research [3, 15, 25] has shown that gradient information of the objective function (either directly available or constructed) along with objective function evaluations could be leveraged in BO algorithms. These BO methods that exploit the gradient information are often known as First Order Bayesian Optimization (FOBO) methods. In many applications like solving PDE-constrained problems, gradients can be obtained cheaply via adjoint methods [8, 28]; in optimizing hyperparameters for deep neural networks, gradients can be obtained using reverse mode differentiation; and in determining optimal policy in reinforcement learning [29, 30], the gradient information can be easily constructed using policy gradient methods. Further, FOBO methods have been shown to perform better compared to ZOBO methods for optimizing standard test functions [40], tuning hyperparameters in ML [10, 17, 18] and for performing minimum energy path calculations [14].

Unlike the Zeroth Order Bayesian Optimization (ZOBO) methods that use single-output GP (with function argument $x$ as input and function evaluation $f(x)$ as output), FOBO methods assume a joint multi-output GP (with function argument $x$ as input and the function evaluation and the gradient evaluation $(f(x), \nabla f(x))$ as joint output). Owing to this joint assumption, there underlies a relationship between the function and the gradient GPs captured in terms of dependencies in the kernels of these GPs (see (2) and (3) in [3]). This aspect enables the FOBO methods to perform the posterior update step more effectively than the ZOBO methods (see Section 2.3). However, the effective posterior update comes with a computational cost rendering the existing FOBO methods unsuitable for optimizing high dimensional functions. Another main disadvantage of these FOBO methods is not utilizing a crucial information that the gradient of the function at maxima is zero while performing the acquisition of the new points (see Section 3). This information potentially could improve the search towards global maxima. Note that this crucial fact has been utilized in all the gradient descent based methods for optimization. Furthermore, owing to the joint assumption in the existing FOBO methods, the associated problem of estimating hyperparameters is unreliable in the presence of noisy function and gradient evaluations. All these disadvantages limit the applicability of the FOBO methods.

To mitigate the aforementioned limitations of FOBO methods, in this work, we propose novel FOBO methods that uses independent GP models for the objective function and its partial derivatives (gradient) by relaxing the multi-output joint GP model. In contrast to the existing FOBO methods, our independent modeling assumption renders the proposed FOBO methods computationally efficient and parallelizable. Another important contribution of this work is to develop novel acquistion algorithms for the FOBO methods that exploit the fact “the gradient of a function at the maxima is zero” in performing effective search towards maxima. This is an important fact utilized in many gradient descent based optimization procedures but not in the existing FOBO or ZOBO methods. Thus, our proposed FOBO methods aspire to utilize the advantages of both BO as well as gradient descent methods. In our proposed FOBO methods, we use the partial derivative GP models to suggest points where the gradient of the objective function is zero, i.e., where the objective function attains its maxima. Even though the gradient of the function is zero at minima and saddle points, our FOBO methods avoid such points during the search.[1] Our methods achieve this by utilizing the GP model of the function as described in Section 3. Our methods enable direct utilization of the gradient information at the ‘new point’ acquisition step aligning with the spirit of BO methods, i.e., exploring the search space on the most likely region of maxima. Table 1 summarizes different BO approaches.

Further in this work, we consider an interesting study of the FOBO methods for RL setting. In utilizing BO for RL (BORL) setting, policies (different ways of choosing actions in states) are treated as evaluation points for the black box model which outputs the evaluated performance of the input policies. The goal in the BORL setting is to obtain a globally optimal policy by exploring the policy space based on the performance evaluations of earlier policies.

We now briefly discuss literature that utilize BO for RL tasks and finally present our contributions in this

[1]This observation could be utilized in the existing FOBO methods as well. However, due to the computational burden of the joint GP model in the existing FOBO methods, we propose to to utilize this fact in independent GP modeling . Note further that we do not require joint GP modeling to utilize this fact.

**Table 1** Comparison of BO approaches

| BO approaches | Uses gradient information | GP model | Utilizes $\nabla f(x) = 0$ | Computationally efficient | Parallelizable |
|---|---|---|---|---|---|
| ZOBO | No | Single GP for $f$ | No | Yes | No |
| Existing FOBO | Yes | Joint GP model for $f$ and $\nabla f$ | No | No | No |
| Our FOBO | Yes | Independent GP models for $f$ and $\nabla f$ | Yes | Yes | Yes |

setting. In [15], Monte-Carlo estimates of the policy using sample trajectory information have been used to measure the policy performance. All other information present in the trajectories was ignored. In [7], the unknown transition function of the underlying Markov decision process is modeled as a Gaussian Process (GP) and the information from the sample trajectories has been used to perform the Bayesian posterior update. Using the updated model, policy evaluation and policy improvement steps were performed. In [39], unlike the prior works of BO in RL which used standard squared exponential kernel for the GP, based on the closeness between policies kernel tailored to RL have been proposed. By comparing the sampled trajectories under different policies, the distance metric for the kernel is obtained and improvements over the classical kernel in the RL setting have been shown. In [39], to identify high quality policies, a BORL algorithm that combines the learned domain models (the transition and reward functions) with the GP models of [7] has been proposed. In [19], kernels have been designed for modeling the non-stationary control problems. This is carried out by partitioning the input space into different regions and considering local GP model in each of the regions showing positive results for the design of wing shape of a UAV.

All the BORL methods discussed so far are model-based that typically use **EI** acquisition algorithm to explore the policy space. Moreover, none of the existing BORL methods exploit the gradient information that could be obtained based on the policy gradients computed from the sample trajectories. In this work, we study and compare FOBO and ZOBO methods in the episodic RL setting. Unlike the prior model-based BORL methods, the FOBO methods adapted for RL setting are model-free. It is important to note that earlier BORL methods were zero-order and utilized only **EI** acquisition algorithm. Further, for the first time our study explores other zero-order and first-order acquisition algorithms like **KG**, **dKG**, **dEI** (see Section 2.2) for the RL setting. We finally present an interesting application of BO methods as an initializing mechanism for the policy gradient (PG) methods [38]. We now summarize the contributions of our work.

## 1.1 Our contributions

We now summarize the contribution of our work and discuss its salient features. We have two major contributions in this work:

- A Novel FOBO approach that exploits the crucial fact *the gradient at maxima is zero*. Note that this fact is central to continuous optimization paradigm, however, it has not been utilized in any of the existing BO approaches. We further illustrate the benefit of incorporating this observation into BO methods through numerical experiments on standard test functions, where we show the performance and computational gain of the proposed FOBO methods over the existing BO methods.
- Exploiting FOBO methods for the policy gradient (PG) Reinforcement Learning (RL) setting. In the context of episodic RL setting, for the first time, we explore FOBO methods that leverage policy gradient information for policy exploration. With the help of multiple benchmark RL tasks, we experimentally study various BORL algorithms. Further, we illustrate an advantage of policies learnt by BORL algorithms in the initialization step of the vanilla PG algorithm. This mitigates the sample inefficiency of the PG algorithm.

The salient features of our novel FOBO approach are:

- Independent GP models for utilizing gradient information effectively over the existing joint GP model.
- Construction of acquisition algorithms that capitalize gradient information to perform direct and effective search better than ZOBO methods.
- Computational efficiency and parallelizability compared to the existing FOBO methods.

## 1.2 Organization of the paper

The rest of the paper is organized as follows. In Section 2, we describe ZOBO and FOBO frameworks. In Section 3, we present our relaxed independent GP models and the proposed FOBO methods that utilise these statistical models. In Section 4, we describe the episodic RL setting and discuss how BO methods can be utilized for episodic RL tasks. In Section 5, we discuss numerical results on standard test functions and benchmark RL tasks. Finally, in Section 6, we provide concluding remarks and discuss interesting future research directions.

## 2 Bayesian optimization

In this section, we present the classical ZOBO methods and existing FOBO methods that attempt to leverage the gradient information.

ZOBO is a general purpose sample efficient global optimization paradigm to solve for the extremum, in particular the maximum of any expensive real-valued function $f$, i.e.,

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in D\subset\mathbb{R}^d} f(\mathbf{x}),$$

where $D$ is the domain of the function $f$. Note that in many ML and RL applications the cost of evaluating certain objective functions is typically high (see [39, 40]).

### 2.1 Surrogate statistical model

ZOBO methods construct a surrogate statistical model for the unknown function $f(\cdot)$ by having a GP prior, i.e., we assume

$$f(\cdot) \sim GP(\mu(\cdot), K(\cdot,\cdot)). \tag{1}$$

GP is a stochastic process that is completely determined by its mean function $\mu(\mathbf{x})$ and the kernel function $K(\mathbf{x},\mathbf{x}')$. The squared exponential function is a kernel that is typically used for parameterizing the covariance function.

$$K(\mathbf{x},\mathbf{x}') = \sigma^2 \exp(-1/2||\mathbf{x}-\mathbf{x}'||_H^2) + \sigma_n^2\delta_{\mathbf{x},\mathbf{x}'},$$

where the norm $||\cdot||_H$ is defined as

$$||u||_H^2 = u'Hu, \quad H = \mathrm{diag}(h_1,\dots,h_d).$$

The $d+2$ variables $\sigma^2,\ \sigma_n^2,\ h_1,\dots,h_d$ are referred as the hyperparameters of the GP model [31].

In addition to function evaluations, if we have access to gradient evaluations $\nabla f(\cdot)$, we could resort to FOBO methods. FOBO methods utilize a well known fact [31] that the gradient of a GP is also a GP. From this fact, FOBO methods model the tuple $(f(\cdot),\nabla f(\cdot))$ as a sample from a multi-output GP with mean function $\tilde{\mu}(\mathbf{x})$ and $\tilde{K}(\mathbf{x},\mathbf{x}')$ defined as follows:

$$\tilde{\mu}(\mathbf{x}) = (\mu(\mathbf{x}),\nabla\mu(\mathbf{x}))^T, \quad \tilde{\mu}\in\mathbb{R}^{d+1}, \tag{2}$$

$$\tilde{K}(\mathbf{x},\mathbf{x}') = \begin{pmatrix} K(\mathbf{x},\mathbf{x}') & J(\mathbf{x},\mathbf{x}') \\ J(\mathbf{x},\mathbf{x}')^T & H(\mathbf{x},\mathbf{x}') \end{pmatrix}, \quad \tilde{K}\in\mathbb{R}^{d+1\times d+1} \tag{3}$$

where $J(\mathbf{x},\mathbf{x}') = \left(\frac{\partial K(\mathbf{x},\mathbf{x}')}{\partial\mathbf{x}'(1)}, \frac{\partial K(\mathbf{x},\mathbf{x}')}{\partial\mathbf{x}'(2)}, \cdots, \frac{\partial K(\mathbf{x},\mathbf{x}')}{\partial\mathbf{x}'(d)}\right)$ and $H(\mathbf{x},\mathbf{x}')$ is the $d\times d$ dimensional Hessian of $K(\mathbf{x},\mathbf{x}')$. Note that the kernel function of the multi-output GP has a dependency in its sub-kernel functions due to the joint assumption. As one only gets estimates of the function and its gradient rather than the true function and its gradient, we model this noisy objective and gradient as perturbations of the true objective and the true gradient by zero mean independent noise with finite variance. Thus, at a point $\mathbf{x}$, the tuple of noise corrupted objective value and its gradient $(\hat{f}(\mathbf{x}),\widehat{\nabla f}(\mathbf{x}))$ is normally distributed, i.e.,

$$\hat{f}(\mathbf{x}),\widehat{\nabla f}(\mathbf{x}) \Big| f(\mathbf{x}),\nabla f(\mathbf{x}) \sim \mathcal{N}\left((f(\mathbf{x}),\nabla f(\mathbf{x})), \mathrm{diag}\left(\lambda^2(\mathbf{x})\right)\right),$$

where $\lambda^2 : D\to\mathbb{R}_{\geq 0}^{d+1}$ specifies the noise variance.

Let us suppose we have evaluated the objective function and its gradient at $n$ points $\mathbf{X} = \{\mathbf{x}^1,\cdots,\mathbf{x}^n\}$. We then obtain the corresponding noisy function and gradient evaluations as $(\hat{f},\widehat{\nabla f})^{(1:n)} = \{(\hat{f}(\mathbf{x}^1),\widehat{\nabla f}(\mathbf{x}^1)),\cdots,(\hat{f}(\mathbf{x}^n),\widehat{\nabla f}(\mathbf{x}^n))\}$. We then modify our prior belief about the GP and incorporate the information $(\hat{f},\widehat{\nabla f})^{(1:n)}$ to obtain the posterior of our GP, i.e., we find the posterior mean $\tilde{\mu}^{(n)}(\mathbf{x})$ and kernel $\tilde{K}^{(n)}(\mathbf{x},\mathbf{x}')$ such that

$$(f(\cdot),\nabla f(\cdot))|(\hat{f},\widehat{\nabla f})^{(1:n)} \sim GP(\tilde{\mu}^{(n)}(\cdot),\tilde{K}^{(n)}(\cdot,\cdot)). \tag{4}$$

The posterior mean function $\tilde{\mu}^{(n)}(\mathbf{x})$ and kernel function $\tilde{K}^{(n)}(\mathbf{x},\mathbf{x}')$ could be obtained as (see [31])

$$\begin{aligned}\tilde{\mu}^{(n)}(\mathbf{x}) = {} & \tilde{\mu}(\mathbf{x}) + \tilde{K}(\mathbf{x},\mathbf{X}) \\ & \left(\tilde{K}(\mathbf{X},\mathbf{X}) + \mathrm{diag}\,\{\lambda^2(\mathbf{x}^1),\cdots,\lambda^2(\mathbf{x}^n)\}\right)^{-1} \\ & \times\left((\hat{f},\widehat{\nabla f})^{(1:n)} - \tilde{\mu}(\mathbf{X})\right),\end{aligned} \tag{5}$$

$$\begin{aligned}\tilde{K}^{(n)}(\mathbf{x},\mathbf{x}') = {} & \tilde{K}(\mathbf{x},\mathbf{x}') - \tilde{K}(\mathbf{x},\mathbf{X}) \\ & \left(\tilde{K}(\mathbf{X},\mathbf{X}) + \mathrm{diag}\{\lambda^2(\mathbf{x}^1),\cdots,\lambda^2(\mathbf{x}^n)\}\right)^{-1}\tilde{K}(\mathbf{X},\mathbf{x}'),\end{aligned} \tag{6}$$

where $\tilde{K}(\mathbf{X},\mathbf{X})$ is a $n(d+1)\times n(d+1)$ matrix whose $(i,j)^{\text{th}}$ $(d+1)\times(d+1)$ matrix is given by $\tilde{K}(\mathbf{x}^i,\mathbf{x}^j)$, $\tilde{K}(\mathbf{x},\mathbf{X})$ is a $(d+1)\times n(d+1)$ matrix whose $i^{\text{th}}$ $(d+1)\times(d+1)$ submatrix is given by $\tilde{K}(\mathbf{x},\mathbf{x}^i)$ and $\tilde{\mu}(\mathbf{X}) = \left(\tilde{\mu}(\mathbf{x}^1),\tilde{\mu}(\mathbf{x}^2),\cdots,\tilde{\mu}(\mathbf{x}^n)\right)$ is a vector of dimension $n(d+1)$. Note that after obtaining the noisy objective and gradient estimates, updating the prior mean $\tilde{\mu}(\cdot)$ and kernel $\tilde{K}(\cdot,\cdot)$ to $\tilde{\mu}^{(n)}(\cdot)$ and $\tilde{K}^{(n)}(\cdot,\cdot)$ correspond to updating the hyperparameters.

### 2.2 Acquisition algorithms

In the previous subsection (Section 2.1), we have seen how FOBO methods construct a statistical model for the objective function and its gradient based on current set of samples. In this subsection, we discuss different acquisition algorithms employed in BO methods that utilize the built statistical model to decide which point to query next in order to guide the search towards finding the global maximum of $f(\cdot)$.

From (4), we know that $f(\cdot)$ is a GP. This means that at any $\mathbf{x}$ the value of the function $f(\mathbf{x})$ is a Gaussian random variable with mean $\mu^{(n)}(\mathbf{x})$ and variance $K^{(n)}(\mathbf{x},\mathbf{x})$. Now let $\mathbb{E}_n[g(f(\mathbf{x}))]$ denote the expectation of the random variable $g(f(\mathbf{x}))$ with respect to the above mentioned distribution. Having set up the notation, we now briefly describe the two acquisition algorithms available in the FOBO literature.

dEI **acquisition** [15] This algorithm determines the next point as the one that gives the maximum expected improvement with respect to the current maximum. Here, the improvement/utility function $I(\mathbf{x})$ (this captures the expected utility of sampling our next point at $\mathbf{x}$) is defined as

$$I(\mathbf{x}) = \mathbb{E}_n[\max\{\hat{f}_{\max}, f(\mathbf{x})\}] - \hat{f}_{\max}, \tag{7}$$

where $\hat{f}_{\max} = \max_{i=1}^{n} \hat{f}(\mathbf{x}^i)$ is the current maximum based on all the point evaluations done so far. Then, the next point $\mathbf{x}^{n+1}$ is determined by maximizing the utility function

$$\mathbf{x}^{n+1} = \arg\max_{\mathbf{x}\in D} I(\mathbf{x}). \tag{8}$$

Note that if the algorithm computes expectation without using the gradient information in (7) then we are using simple **EI** acquisition function [16].

*Remark 1* The expectation in (7) is averaging the randomness in the function value $f(\mathbf{x})$ after conditioning on the noisy function values $\hat{f}(\mathbf{x}^1), \hat{f}(\mathbf{x}^2), \ldots, \hat{f}(\mathbf{x}^n)$. We detail this point below.

– Before obtaining any samples, the unknown function value at a point $\mathbf{x}$ is a Gaussian random variable, i.e., $f(\mathbf{x})$ is a normally distributed with mean $\mu^{(0)}(\mathbf{x})$ and variance $K^{(0)}(\mathbf{x},\mathbf{x})$.
– After we obtain noisy evaluations of the function at the domain points $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$, the unknown function value at a point $\mathbf{x}$ is still a Gaussian random variable but with mean and variance updated to $\mu^{(n)}(\mathbf{x})$ and $K^{(n)}(\mathbf{x},\mathbf{x})$ (see (5) and (6) respectively). Note that the function values $f(\mathbf{x}^1), f(\mathbf{x}^2), \ldots, f(\mathbf{x}^n)$ would be random variables even at the points $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$ where we have already evaluated the function. This is because when we evaluate the function at a point, we do not get the exact value of the function rather we only get noise corrupted version of the function's original value.
– Therefore, after noisy function evaluation at the domain points $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$, $f(\mathbf{x})$ is still a normally distributed random variable but with updated mean $\mu^{(n)}(\mathbf{x})$ and updated variance $K^{(n)}(\mathbf{x},\mathbf{x})$. The notation $\mathbb{E}_n[g(f(\mathbf{x}))]$ thus refers to expectation w.r.t. the randomness in $f(\mathbf{x})$ after conditioning on the noisy function values $\hat{f}(\mathbf{x}^1), \hat{f}(\mathbf{x}^2), \ldots, \hat{f}(\mathbf{x}^n)$.

For example:

$$\mathbb{E}_n[f(\mathbf{x})] = \mu^{(n)}(\mathbf{x}) \tag{9}$$

$$\mathbb{E}_n[f(\mathbf{x}) - \mu^{(n)}(\mathbf{x})]^2 = K^{(n)}(\mathbf{x},\mathbf{x}) \tag{10}$$

Further, for any general function $g(\cdot)$,

$$\mathbb{E}_n[g(f(\mathbf{x})] = \int_{-\infty}^{\infty} g(f(\mathbf{x}))\mathcal{N}(\mathbf{x};\mu^{(n)}(\mathbf{x}),K^{(n)}(\mathbf{x},\mathbf{x}))d\mathbf{x}, \tag{11}$$

where $\mathcal{N}(\mathbf{x};\mu^{(n)}(\mathbf{x}),K^{(n)}(\mathbf{x},\mathbf{x}))$ denotes the normal density, i.e.,

$$\mathcal{N}(\mathbf{x};\mu^{(n)}(\mathbf{x}),K^{(n)}(\mathbf{x},\mathbf{x})) = \frac{\exp\left(-\left(\frac{f(\mathbf{x})-\mu^{(n)}(\mathbf{x})}{2K^{(n)}(\mathbf{x},\mathbf{x})}\right)^2\right)}{2K^{(n)}(\mathbf{x},\mathbf{x})\sqrt{2\pi}}.$$

dKG **acquisition** [40] This algorithm determines the next point as the one that has the highest expected knowledge gradient [9].

$$\mathbf{x}^{n+1} = \arg\max_{z\in D} dKG(z) \tag{12}$$

where $dKG(z)$ is defined as;

$$dKG(z) \triangleq \mathbb{E}_n\left[\max_{\mathbf{x}\in D} \tilde{\mu}_1^{(n+1)}(\mathbf{x}) \,\middle|\, \mathbf{x}^{n+1} = \mathbf{z}\right] - \max_{\mathbf{x}\in D} \tilde{\mu}_1^{(n)}(\mathbf{x}). \tag{13}$$

From the evolution of $\tilde{\mu}^{(n+1)}(\mathbf{x})$ in (14), we can determine the random variable $\tilde{\mu}_1^{(n+1)}(\mathbf{x})$ (the first component of $\tilde{\mu}^{(n+1)}(\mathbf{x})$, $\tilde{\mu}_1^{(n+1)}(\mathbf{x})$ is essentially $\mu^{(n+1)}(\mathbf{x})$ see (2)).

$$\tilde{\mu}^{(n+1)}(\mathbf{x}) = \tilde{\mu}^{(n)}(\mathbf{x}) + \tilde{K}(\mathbf{x},\mathbf{z})\left(\tilde{K}(\mathbf{z},\mathbf{z}) + \operatorname{diag}\{\lambda^2(\mathbf{z})\}\right)^{-1} \left((\hat{f}(\mathbf{z}), \widehat{\nabla f}(\mathbf{z})) - \tilde{\mu}^{(n)}(\mathbf{z})\right). \tag{14}$$

Note that if the algorithm computes expectation without using the gradient information then we are using **KG** acquisition function [9].

Both the acquisition algorithms **dEI** and **dKG** use one-step lookahead rule to determine the next point. And each of these optimizations in (8) and (13) can be solved efficiently [9]. Note that one could generalize these acquisition algorithms to incorporate multi-step look ahead instead of one-step look ahead (see [40]). The complete description of the existing FOBO methods is given in Algorithm 1.

*Remark 2* We briefly describe the **dKG** algorithm. Let $\mathbf{x}^1, \mathbf{x}^2, \ldots \mathbf{x}^n$ correspond to the $n$ query locations and let the corresponding noisy function evaluation at these locations be $\hat{f}(\mathbf{x}^1), \hat{f}(\mathbf{x}^2), \ldots, \hat{f}(\mathbf{x}^n)$. Then our current best bet for the location of maxima would be $\arg\max_{\mathbf{x}\in D}\tilde{\mu}_1^{(n)}(\mathbf{x})$, as $\tilde{\mu}_1^{(n)}(\mathbf{x})$ is the surrogate function for $f(\cdot)$ based on our observations. Suppose we are given an additional chance of querying at a new location $\mathbf{x}^{n+1}$ then by similar reasoning it would be $\arg\max_{\mathbf{x}\in D}\tilde{\mu}_1^{(n+1)}(\mathbf{x})$. However, the mean of $\tilde{\mu}_1^{(n+1)}(\mathbf{x})$ is a random variable as we have not really evaluated the function and gradient at $\mathbf{x}^{n+1} = \mathbf{z}$ and thus we need to take expectation w.r.t. $\mathbb{E}_n$. Therefore, in the **dKG** acquisition algorithm, we would like to evaluate our function and gradient at a new location $\mathbf{z}$ where we get maximum gain given by (13). We now describe why $\tilde{\mu}_1^{(n+1)}(\mathbf{x})$ is a random variable: The (14) corresponds to the posterior update step of the mean function when an additional noisy function and gradient evaluation at point $\mathbf{z}$ is given. Note that the R.H.S of (14) involves the term $(\hat{f}(\mathbf{z}), \widehat{\nabla f}(\mathbf{z}))$. Since, we have not really evaluated the function at $\mathbf{z}$, there is a probability distribution associated with function and gradient value at $\mathbf{z}$ given by mean $\tilde{\mu}^{(n)}(\mathbf{z})$ and the covariance function $\tilde{K}^{(n)}(\mathbf{z},\mathbf{z})$. Thus, to average this randomness we take the expectation $\mathbb{E}_n$ of $\tilde{\mu}_1^{(n+1)}(\mathbf{x})$ in (13).

**Algorithm 1** Existing FOBO methods.

```
1: Input:
   $N \leftarrow$ Budget on number of evaluations
   $(\hat{f}(\cdot), \widehat{\nabla f(\cdot)})$ - noisy function and gradient evaluator
2: Output: $\mathbf{x}^* \approx \arg\max_{\mathbf{x}} f(\mathbf{x})$
3: Algorithm:
4: $\mathcal{D}_0 = \{\emptyset\}$
5: for $n = 0$ to $N-1$ do
6:     Find new point $\mathbf{x}^{n+1}$ using standard acquisition algorithm (E.g. dEI or dKG) for the function GP model.
7:     Evaluate $(\hat{f}(\mathbf{x}^{n+1}), \widehat{\nabla f}(\mathbf{x}^{n+1}))$.
8:     Augment the data:
       $\mathcal{D}^{n+1} = \mathcal{D}^n \cup \{(\hat{f}(\mathbf{x}^{n+1}), \widehat{\nabla f}(\mathbf{x}^{n+1}))\}$.
9:     Update the function and partial derivative joint GPs using the augmented data $\mathcal{D}^{n+1}$ according to (5) and (6).
10: end for
11: return $\mathbf{x}^* = \arg\max_{\mathbf{x}} \mu^{(N)}(\mathbf{x})$, the maximum of posterior mean function of the function GP model (see (1)).
```

## 2.3 Pros and cons of the existing FOBO methods

The advantages of the existing FOBO methods over the ZOBO methods stem from their ability to utilize the gradient information to perform the posterior update step effectively. In the posterior update step, FOBO methods effectively filters out functions in the GP that do not conform to both the earlier function and gradient evaluations. Further, this advantage gets propagated into their acquisition step as their improvement function utilizes conditional expectation with respect to the computed posterior ($\mathbb{E}_n$ in acquisition algorithms **dEI** and **dKG** computes conditional expectation w.r.t. posterior). However, performing the posterior update is computationally challenging. For instance, if we have obtained $n$ evaluations of the objective function and the corresponding gradients, the posterior update in FOBO methods requires inverting the kernel matrix of dimension $n(d+1)$, requiring $O((n(d+1))^3)$ computational effort as opposed to $O(n^3)$ in the ZOBO methods. Note that the ZOBO methods enjoy this computational benefit at the cost of not utilizing the gradient information. Further, in the acquisition step of the FOBO methods, the computation of conditional expectation would involve matrix-vector multiplication of complexity $O(n^2(d+1)^2)$. Thus, optimizing the utility function involves several matrix operations resulting in increased computational effort. The aforementioned reasons present a serious bottleneck in utilizing the FOBO methods. To mitigate the computational burden, [3, 40] proposes to utilize only the information from the best directional derivative instead of all the partial derivatives. However, this does not utilize the full potential of the available gradient information. In the next section, we discuss our model and FOBO methods that utilize the complete gradient information with significantly less computational requirement.

## 3 Proposed model and algorithms

In Section 2.3, we discussed how the existing FOBO methods utilize the gradient information in the posterior update, i.e., essentially filtering (or assigning low probabilities to) functions in the GP model that do not conform to earlier function and gradient evaluations. However, this is an indirect utilization of the available gradient information. A direct use of the gradient information would be to utilize the gradient GP model to search for points at which the gradient of the function vanishes, i.e., search for $\mathbf{x}$ such that $\nabla f(\mathbf{x}) = 0$ along with the search for global maximum carried out by the function GP model (see (1)). To achieve this, we strictly do not require a joint GP model between function and its partial derivatives. It would suffice if we have models for the partial derivatives which would help us in performing the search.

### 3.1 Independent surrogate statistical models

In our FOBO methods, we relax the joint assumption (as modeled in existing FOBO methods) between the objective function and its partial derivatives and treat each one of them as independent GPs. Along with the function GP model (1), we model each of the partial derivatives of the objective function as independent GPs with separate mean and kernel functions, i.e.,

$$\begin{aligned} f(\cdot) &\sim GP(\mu(\cdot), K(\cdot,\cdot)),\\ \frac{\partial f(\cdot)}{\partial \mathbf{x}(i)} &\sim GP(\mu_i(\cdot), K_i(\cdot,\cdot)),\\ & \qquad i \in \{1,\dots,d\}. \end{aligned} \tag{15}$$

Unlike the posterior update step in existing FOBO methods that utilize all the information $(\hat{f}, \widehat{\nabla f})^{(1:n)}$, the posterior update in each of the GP models is carried out independently by appropriately segregating the information in $(\hat{f}, \widehat{\nabla f})^{(1:n)}$. In our FOBO methods, due to the independence assumption, the $i$th partial derivative GP model only utilizes $(\frac{\partial \hat{f}}{\partial \mathbf{x}(i)})^{(1:n)}$. This constructs posterior GP models (similar to (4)) for all the partial derivatives and the objective function. Note that we have $(d+1)$ GP models.

## 3.2 Acquisition algorithms

Now we describe how to utilize the above constructed GP models to acquire the next point. The important fact $\nabla f(\mathbf{x}) = 0$ at the point of maxima suggests that the acquisition algorithm should search for points at which gradient vanishes. Thus, in each of the partial derivative GP models the novel acquisition algorithm that we construct search for points at which $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)} = 0,\ i \in \{1,\dots,d\}$. This is accomplished in a way similar to the **EI** acquisition which looks for next point that has highest expected improvement. In our setting, the acquisition algorithm searches for next points at which expected value of the absolute partial derivative is minimum. In order to capture this, we define utility functions corresponding to each of the partial derivative GP models as

$$I_i(\mathbf{x}) = \mathbb{E}_n\left(\left|\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)}\right|\right) \quad i \in \{1,\dots,d\}. \tag{16}$$

The expectation in (16) could be easily calculated (see Lemma 1) as $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)}$ is a Gaussian random variable with mean $\mu_i^{(n)}(\mathbf{x})$ and variance $K_i^{(n)}(\mathbf{x},\mathbf{x})$. Then, the next point suggested by the utility function of the $i$th partial derivative GP model is given by

$$\mathbf{x}_i^{n+1} = \arg\min_{\mathbf{x}\in D} I_i(\mathbf{x}),\ i \in \{1,\dots,d\}. \tag{17}$$

To simplify the expressions we have abused the notation $\mathbb{E}_n$ in (16) and (17). This actually denotes the conditional expectation taken with respect to posterior distribution of the corresponding GP models.

*Remark 3* We clarify the abuse of notation in (16). As we have assumed independent GP models for the function $f$ as well as its partial derivatives $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)},\ i = \{1,2,\dots,d\}$, the abuse of notation $\mathbb{E}_n$ refers to taking the expectation w.r.t. the posterior distribution of the $i$th partial derivative $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)},\ i \in \{1,2,\dots,d\}$ when we obtain the noisy function and partial derivative (gradient) evaluations $(\hat{f}(\mathbf{x}^1), \nabla\hat{f}(\mathbf{x}^1)), (\hat{f}(\mathbf{x}^2), \nabla\hat{f}(\mathbf{x}^2)), \dots, (\hat{f}(\mathbf{x}^n), \nabla\hat{f}(\mathbf{x}^n))$, i.e.,

$$\left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)}\right| \frac{\partial \hat{f}(\mathbf{x}^1)}{\partial \mathbf{x}(i)}, \frac{\partial \hat{f}(\mathbf{x}^2)}{\partial \mathbf{x}(i)}, \dots, \frac{\partial \hat{f}(\mathbf{x}^n)}{\partial \mathbf{x}(i)}$$
$$\sim \mathcal{N}(\mathbf{x}; \mu_i^{(n)}(\mathbf{x}), K_i^{(n)}(\mathbf{x},\mathbf{x}))i \in \{1,2,\dots,d\}$$

where $\mathcal{N}(\mathbf{x};\cdot,\cdot)$ denotes the Gaussian or Normal distribution. Thus, in (16) as $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)}$ is a Gaussian random variable with mean $\mu_i^{(n)}(\mathbf{x})$ and variance $K_i^{(n)}(\mathbf{x},\mathbf{x})$ and the expectation is calculated w.r.t to this distribution. We have used the same notation $\mathbb{E}_n$ for all indices $i \in \{1,2,\dots,d\}$ instead of $\mathbb{E}_n^i$ to make the notation simpler.

Before we proceed further to describe how to utilize points suggested by our acquisition algorithm, we first present the Lemma 1 that helps us to evaluate the expectation in (16).

**Lemma 1** *Let Z be a Gaussian random variable with mean $\mu \le 0$ and variance $\sigma^2$. Then, $\mathbb{E}[|z|] = 2\sigma\phi(-\frac{\mu}{\sigma}) + \mu[\Phi(\frac{\mu}{\sigma}) - \Phi(\frac{-\mu}{\sigma})]$ where $\phi(\cdot)$ and $\Phi(\cdot)$ correpond to the normal density function and the normal cumulative distribution function, respectively.*

*Proof*

$$\begin{aligned}
\mathbb{E}[|Z|] &= \int_{-\infty}^{\infty} |y|\phi(y)dy\\
&= \int_{-\infty}^{0} -y\phi(y)dy + \int_{0}^{\infty} y\phi(y)dy\\
&= \int_{-\infty}^{\frac{-\mu}{\sigma}} -(\sigma x+\mu)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)dx\\
&\quad + \int_{\frac{-\mu}{\sigma}}^{\infty} (\sigma x+\mu)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)dx\\
&= 2\sigma\int_{\frac{-\mu}{\sigma}}^{\infty} \frac{1}{\sqrt{2\pi}} x\exp\left(-\frac{x^2}{2}\right)dx\\
&\quad +\mu\left[1 - 2\int_{-\infty}^{\frac{-\mu}{\sigma}} \frac{1}{\sqrt{2\pi}} x\exp\left(-\frac{x^2}{2}\right)dx\right]\\
&= 2\sigma\frac{1}{\sqrt{2\pi}}\int_{\frac{\mu^2}{\sigma^2}}^{\infty} \exp(-t)dt + \mu\left[1-2\Phi\left(-\frac{\mu}{\sigma}\right)\right]\\
&= 2\sigma\frac{1}{\sqrt{2\pi}}\phi\left(\frac{\mu}{\sigma}\right) + \mu\left[1-2\Phi\left(-\frac{\mu}{\sigma}\right)\right] \qquad \square
\end{aligned}$$

**Corollary 1** *Lemma 1 holds for $\mu \ge 0$.*

*Proof* This simply follows from symmetry. □

We now describe how to utilize the points suggested by our gradient acquisition algorithm. Let $\mathbf{x}_i^{n+1}$ denote the point suggested by the acquisition algorithm of the function GP model using any one of the ZOBO acquisition algorithms. $P_{n+1} = \{\mathbf{x}_0^{n+1}, \mathbf{x}_1^{n+1}, \ldots, \mathbf{x}_d^{n+1}\}$ denote the potential locations to sample at next step $n+1$. Note that the points suggested by any of these partial derivative models may correspond to minima or saddle points. Thus, it is important to validate the significance of each of these points before utilizing them in the search for the global maximum. This is achieved by utilizing the function GP model.

In our FOBO methods, we measure the significance of each of these points by evaluating the mean of the function GP at these locations, i.e., the significance of the candidate points $S_{n+1} = \{\mu^{(n)}(\mathbf{x}_0^{n+1}), \mu^{(n)}(\mathbf{x}_1^{n+1}), \ldots, \mu^{(n)}(\mathbf{x}_d^{n+1})\}$, where $\mu^{(n)}(\cdot)$ denotes the posterior mean function of the function GP after acquisition of $n$ points. Each of these values in $S_{n+1}$ suggest the utility of sampling at each of these locations. Now, using this information, our method decides the next location to implant the query in two possible ways.

In the first way, the information is aggregated by taking a weighted convex combination of the points in $P_{n+1}$,

$$\mathbf{x}^{n+1} = \sum_{i=0}^{d} \frac{\exp(\mu^{(n)}(\mathbf{x}_i^{n+1}))}{\sum_{i=0}^{d} \exp(\mu^{(n)}(\mathbf{x}_i^{n+1}))} \mathbf{x}_i^{n+1}. \tag{18}$$

In the second way, the information is aggregated by taking the point that has the maximum significance,

$$\begin{aligned} i^* &= \arg\max_i \mu^{(n)}(\mathbf{x}_i^{n+1}), \\ \mathbf{x}^{n+1} &= \mathbf{x}_{i^*}^{n+1}. \end{aligned} \tag{19}$$

The complete description of our proposed FOBO methods is given in Algorithm 2. Note that the choice of acquisition algorithm for the function GP model (see step 6 of the Algorithm 2) gives rise to distinct FOBO methods. In our experiments, we considered both **EI** and **KG** acquisition algorithms for the function GP model to compare against other BO methods.

## 3.3 Computational complexity

In the posterior update step the computational effort of the proposed independent GP models is $O((d+1) * n^3)$ as opposed to $O((d+1)^3 * n^3)$ in the earlier joint GP model. This is a significant computational saving. Even though we assumed independent models for the function and partial derivative GPs, we are able to completely utilize the gradient information due to our novel acquisition functions that search for new points $\{\mathbf{x} : \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}(i)} = 0,\ i \in \{1, \ldots, d\}\}$. Further, each of these acquisitions in the partial derivative GPs could be carried out in a parallel manner making our FOBO methods faster.

**Algorithm 2** Proposed FOBO methods.

```
1:  Input:
    $N \leftarrow$ Budget on number of evaluations
    $(\hat{f}(\cdot), \widehat{\nabla f}(\cdot))$ - noisy function and gradient evaluator
2:  Output: $\mathbf{x}^* \approx \arg\max_{\mathbf{x}} f(\mathbf{x})$
3:  Algorithm:
4:  $\mathcal{D}_0 = \{\emptyset\}$
5:  for $n = 0$ to $N-1$ do
6:      Find new point $\mathbf{x}_0^{n+1}$ using standard acquisition
        algorithm (EI or KG) for the function GP model.
7:      Obtain new points $\mathbf{x}_i^{n+1},\ i \in \{1, \ldots, d\}$ by
        optimizing the acquisition function (see (17)) of the
        partial derivative GP models.
8:      Find the next point $\mathbf{x}^{n+1}$ by aggregating the new
        points $\mathbf{x}_i^{n+1},\ i \in \{0, \ldots, d\}$ utilizing (18), (19).
9:      Evaluate $(\hat{f}(\mathbf{x}^{n+1}), \widehat{\nabla f}(\mathbf{x}^{n+1}))$.
10:     Augment the data:
        $\mathcal{D}^{n+1} = \mathcal{D}^n \cup \{(\hat{f}(\mathbf{x}^{n+1}), \widehat{\nabla f}(\mathbf{x}^{n+1}))\}$.
11:     Update the function and partial derivative indepen-
        dent GPs using the augmented data $\mathcal{D}^{n+1}$.
12: end for
13: return $\mathbf{x}^* = \arg\max_{\mathbf{x}} \mu^{(N)}(\mathbf{x})$, the maximum of
    posterior mean function of the function GP model.
```

*Remark 4* Here we briefly outline the asymptotic convergence of the proposed model. The rigorous analysis of the same will be considered in our future work along the lines in [34]. For ease of exposition, we will assume the function and gradient evaluations are not corrupted by noise. Similar reasoning can be carried out even if there is noise. Further, we assume that there are finite extrema points (local maxima, minima and saddle points), i.e, there are finite points where gradient vanishes. We first discuss the asymptotic convergence results of BO algorithms available in the literature. Based on this, we then outline the asymptotic convergence of our model.

In [37], the convergence of EI algorithm with fixed mean and covariance kernel is given. The condition required for convergence is that the acquisition algorithm should produce dense sequence of evaluation points in the domain. They show that if the domain is compact and the covariance kernel of Gaussian Process (GP) is smooth then EI algorithm converges to global maxima. Further, when the parameters are estimated sequentially from the observed data, the convergence of the EI algorithm is given in [5].

Now we discuss the convergence of the proposed model. In our proposed model, we considered two approaches for combining the points suggested by the gradient GP model (see (18) and (19)).

We consider the second approach suggested by (19) first. This approach considers that point that has the highest function value (by evaluating on the function GP model) among the points suggested by each of the partial derivative models. When we query for the value of the function at this point, if the point is not one of the global maxima, then we will lose a function evaluation, however, we would know the behaviour of the function around that extremum point. As there are only finite number of such extrema points, after finite time the behaviour of our proposed model will then follow the standard EI algorithm for which the asymptotic convergence has already been established. This shows the asymptotic convergence of our second approach.

In our first approach (18) for asymptotic convergence, we need to introduce a temperature parameter $T$ when we perform the convex combination of the points suggested by partial derivative models, i.e.,

$$\mathbf{x}^{n+1} = \sum_{i=0}^{d} \frac{\exp(\mu^{(n)}(\mathbf{x}_i^{n+1})/T)}{\sum_{i=0}^{d}\exp(\mu^{(n)}(\mathbf{x}_i^{n+1})/T)}\mathbf{x}_i^{n+1}. \qquad (20)$$

The parameter $T = 1$ corresponds to the default weighting as in (18). For large parameter value of $T$, i.e., as $T \to \infty$, we give equal importance to all the points and for small parameter value of $T$, i.e., as $T \to 0$ it essentially becomes the second approach (19). Thus, in this approach, we start with large value for $T$ and gradually reduce, i.e., we let $T \to 0$ which corresponds to the second approach (19) and thus ensures asymptotic convergence. This is the standard practice for achieving both exploration and exploitation.

## 4 Episodic reinforcement learning

In this section, we briefly describe the episodic Reinforcement Learning (RL) setting and how one could utilize BO methods for RL tasks.

In an RL problem, an agent explores the world state or an environment by choosing actions in a trial and error manner with the goal of maximizing a long-term return. Markov decision processes (MDPs) form the mathematical framework for studying RL problems. An MDP is described by the tuple $(S, A, \mathcal{P}, \mathcal{R}, d)$. Here, $S$ and $A$ correspond to state and action spaces respectively. Each element $i \in S$ denotes the state of the environment and each action $a \in A$ denotes the choice available to an RL agent. The transition function $\mathcal{P}$ describes the state evolution of the environment based on the action chosen by the agent, i.e., $\mathcal{P}(j \mid i, a)$ specifies the conditional probability of transitioning to state $j$ when the action $a$ is taken given the current state $i$. The reward function $\mathcal{R}(i, a, j)$ denotes the single stage reward that is obtained for taking action $a$ in state $i$ and transitioning to $j$. $d(\cdot)$ denotes the initial distribution of the environment state. In the episodic RL setting, we have a special terminal state $s^*$ which has the property $\mathcal{P}(s^* \mid s^*, a) = 1$ and $\mathcal{R}(s^*, a, i) = 0\ \forall a \in A,\ i \in S$. Every episode begins with initial state chosen according to the distribution $d(\cdot)$ and terminates at the state $s^*$.

It is often convenient to view actions as being chosen from a given policy, i.e., a rule or method for selecting actions. A parameterized stochastic policy specifies a mapping from states to probability distribution over actions, i.e., given parameter $\mathbf{x} \in \mathcal{R}^k$, $\pi_{\mathbf{x}}(\cdot \mid i)$ gives a probability distribution over the action space $A$ for the state $i$. A random trajectory $\tau = (s_0, a_0, s_1, a_1, \cdots, a_{T-1}, s_T)$ produced by a policy $\pi_{\mathbf{x}}$ is the sequence of states and actions that is obtained when we choose the initial state $s_0 \sim d(\cdot)$ and the actions $a_t \sim \pi_{\mathbf{x}}(\cdot \mid s_t)$ where $T$ denotes the random termination time of the episode. The probability of this trajectory $\tau$ is given by

$$P(\tau|\mathbf{x}) = d(s_0)\prod_{t=1}^{T}\mathcal{P}(s_t|s_{t-1}, a_{t-1})\pi_{\mathbf{x}}(a_{t-1} \mid s_{t-1}),$$

and the corresponding reward obtained in the trajectory $\tau$ is,

$$\sigma(\tau) = \sum_{t=0}^{T-1}\mathcal{R}(s_t, a_t, s_{t+1}). \qquad (21)$$

Observe that $\sigma(\tau)$ is the sum of single stage rewards. We assume that $T < \infty$ for all trajectories $\tau$. This ensures that $\sigma(\tau)$ is finite for all trajectories. The performance measure of the policy $\pi_{\mathbf{x}}$ is the expected return $f(\mathbf{x})$ gained by following the policy $\pi_{\mathbf{x}}$,

$$f(\mathbf{x}) = \mathbb{E}_\tau[\sigma(\tau) \mid \mathbf{x}] = \int \sigma(\tau)P(\tau|\mathbf{x})d\tau.$$

The objective in the episodic RL setting is to determine the policy parameters that maximize the performance measure,

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in D\subset\mathbb{R}^L} f(\mathbf{x}). \qquad (22)$$

where $D$ denotes the space of the possible parameter values.

An important solution approach for solving (22) under model free RL setting utilizes the policy-based methods that directly search the policy space [27, 32, 38]. These methods have better convergence guarantees and are very effective in high dimensional settings or continuous action spaces [24]. Moreover, the policy-based methods have also been used successfully in many machine learning applications [26, 41]. The success of policy-based methods can be attributed to the use of gradient information to explore the policy space in a principled way as opposed to value-based methods [22]. However, they suffer from instability. This can be attributed to the presence of noise (high variance) in the gradient estimates and the sensitivity of the policy parameter updates to the choice of step size schedule. Many variance reduction methods proposed in the literature [27] have ameliorated the instability issue. Also, better methods for choosing step sizes have evolved [13, 36]. However, it is unclear which amongst these methods is the best for step size selection and which therefore, requires highly experienced hands in practice to design a stable policy gradient method. Furthermore, the slow and local convergence owing to the step size choice restricts its wider applicability.

In the conventional policy-based method, the goal of maximizing the $f(\mathbf{x})$ is accomplished by performing a gradient search in the parameter $\mathbf{x}$. The gradient $\nabla f(\mathbf{x})$ of the expected return $f(\mathbf{x})$ can be computed as (see [38])

$$\nabla f(\mathbf{x}) = \mathbb{E}_\tau[\nabla \log(P(\tau|\mathbf{x}))\sigma(\tau)] \tag{23}$$

$$= \mathbb{E}_\tau\left[\left(\sum_{t=0}^{T-1} \nabla \log(\pi_{\mathbf{x}}(a_t \mid s_t))\right)\sigma(\tau)\right]. \tag{24}$$

A simple calculation reveals (24) follows from (23). Now one can use (24) to obtain gradient estimate $\widehat{\nabla f}(\mathbf{x})$ by simulating multiple trajectories, computing the quantities in (24) and averaging them (see Algorithm 3). Then, in the standard policy gradient (PG) (policy-based) method, $\mathbf{x}$ gets updated along the positive estimated gradient direction,

$$\mathbf{x}^{n+1} = \mathbf{x}^n + c_n\widehat{\nabla f}(\mathbf{x}^n),$$

where $c_n$ is the step-size parameter. Assuming stability of iterates $\{\mathbf{x}^n\}_{n\geq 0}$ above, under reasonable conditions the updates $\mathbf{x}^n$ converge almost surely to a parameter $\mathbf{x}_l^*$ such that $\pi_{\mathbf{x}_l^*}$ is a locally optimal policy. As mentioned earlier, the main difficulty with this approach however is the choice of the step size sequence when one obtains noisy gradient estimates. This leads to instability in practice and sometimes even when we are fortunate to have stable iterates, we get stuck at local minima. These problems can be totally avoided by utilizing BO paradigm.

In the BORL setting, policies are treated as evaluation points for the BO black box. The output of the black box is the expected return (expected total reward collected during an episode) of the policy and its gradient. Note that we obtain sample trajectories as the intermediate output of the blackbox. The expected return can be easily constructed by averaging the total reward over multiple sample trajectories and the gradient estimates can be constructed by utilizing the policy gradient theorem for episodic RL tasks [38]. We now provide description of the algorithm for obtaining expected return and its gradient for the episodic RL tasks for a given policy. This can be utilized in the BO paradigm to optimize the expected return over the space of policies.

In Algorithm 3, at a given new point $\mathbf{x}$, we let the RL agent to follow the fixed policy $\pi_{\mathbf{x}}$ to obtain the estimates of the objective function $\hat{f}(\mathbf{x})$ and its gradient $\widehat{\nabla f}(\mathbf{x})$. In steps 4 to 17, the agent follows the policy $\mathbf{x}$ for $E$ number of episodes and collects the sample trajectory information from these runs. Using this information, the agent estimates the objective and gradient for the current policy $\pi_{\mathbf{x}}$ in step 18.

**Algorithm 3** Expected return and policy gradient estimation algorithm.

```
1:  Input:
      $\mathbf{x} \leftarrow$ Policy parameters
      $\pi_{\mathbf{x}} \leftarrow$ Parametrized set of policies
      $E \leftarrow$ Number of episodes to average over to get estimates of $f(\mathbf{x})$ and $\nabla f(\mathbf{x})$
2:  Output: $(\hat{f}(\mathbf{x}), \widehat{\nabla f}(\mathbf{x}))$
3:  Algorithm:
4:  $\mathcal{D}_0 = \{\emptyset\}$
5:  $R \leftarrow 0; \Psi \leftarrow 0$
6:  for $e = 1$ to $E$ do
7:      $R_e \leftarrow 0;\ \Psi_e \leftarrow 0;$
8:      obtain $s_0^e \sim d(.)$
9:      for $t = 0$ to $T-1$ do
10:         choose action $a_t^e \sim \pi_{\mathbf{x}}(\cdot \mid s_t)$
11:         obtain $s_{t+1}^e \sim P(.|s_t^e, a_t^e)$ and
12:         obtain $R(s_t^e, a_t^e, s_{t+1}^e)$
13:         $R_e \leftarrow R_e + R(s_t^e, a_t^e, s_{t+1}^e)$
14:         $\Psi_e \leftarrow \Psi_e + \nabla \log \pi_{\mathbf{x}}(a_t^e \mid s_t^e)$
15:     end for
16:     $R \leftarrow R + R_e;\ \Psi \leftarrow \Psi + \Psi_e R_e$
17: end for
18: $\hat{f}(\mathbf{x}) \leftarrow R/E;\ \widehat{\nabla f}(\mathbf{x}) \leftarrow \Psi/E$
19: return $(\hat{f}(\mathbf{x}), \widehat{\nabla f}(\mathbf{x}))$
```

## 5 Performance evaluation and results

In this section, we compare the performance and running time of the proposed FOBO methods with existing BO methods. In the following subsections, we provide details of the experimental setup and discuss the results obtained.

### 5.1 Experimental setup

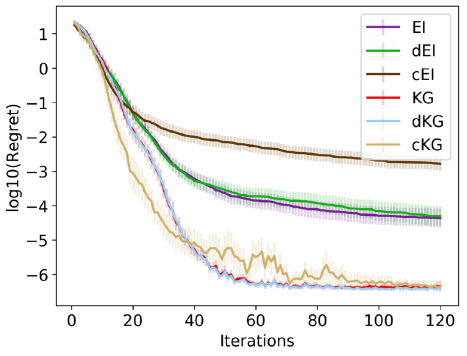
We compare the performance of proposed FOBO methods against the existing BO methods using five synthetic test functions and three benchmark RL tasks. We refer the proposed methods as **cEI** and **cKG** based on the type of acquisition algorithm used for the function GP model (in step 4 for Algorithm 2). Note that **cEI** and **cKG** use EI [12] and KG [9] acquisition algorithms respectively.

We refer the existing ZOBO methods as **EI** (utilizes EI acquisition algorithm) and **KG** (KG acquisition algorithm) and the existing FOBO methods as **dEI** (dEI acquisition algorithm (8)) and **dKG** (dKG acquisition algorithm (13)). In all the BO methods, we utilize Bayesian treatment of hyperparameters similar to [40] as this blends the acquisition step and the posterior step in a natural manner [33]. We used MPI (Multi Processing Interface) to run all our experiments on a multi-node cluster. Each node has 24 cores (2.5 GHz) and primary memory of 128GB which is shared by them.

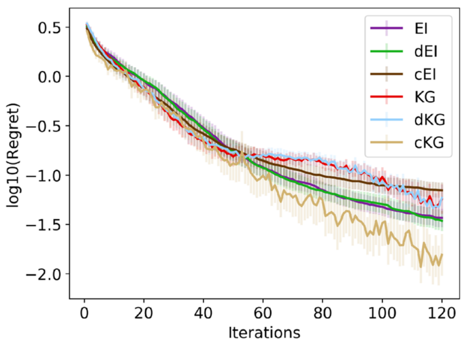
### 5.1.1 Synthetic test functions

We consider five synthetic test functions for performance evaluation as in [40] namely Branin (dimension of the domain $d = 2$), Levy ($d = 4$), Rosenbrock ($d = 4$), Ackley ($d = 5$) and Hartmann ($d = 6$) functions. In order to test the robustness of the BO methods, we add random Gaussian noise of 0 mean and 0.25 variance to objective functions and their derivate evaluations in all the aforementioned functions. We consider only one-step look-ahead in the acquisition algorithms of all the methods.

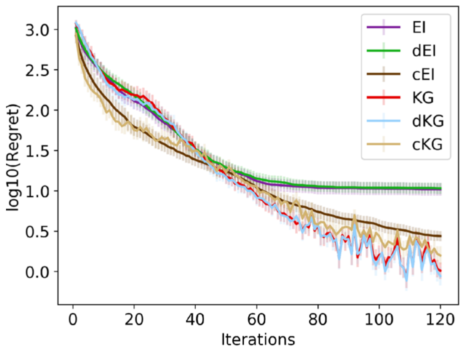
**Fig. 2** Performance comparison of BO methods on synthetic test functions



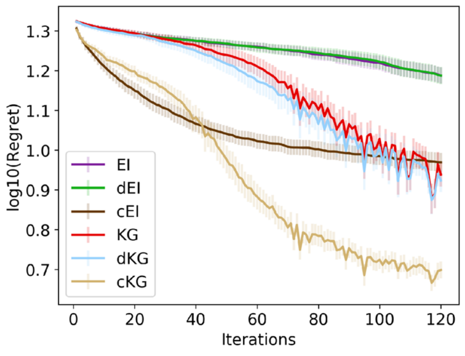
(a) Branin2 function: cKG converges closer to optima quickly compared to other algorithms and reaches error of order 0.00001.



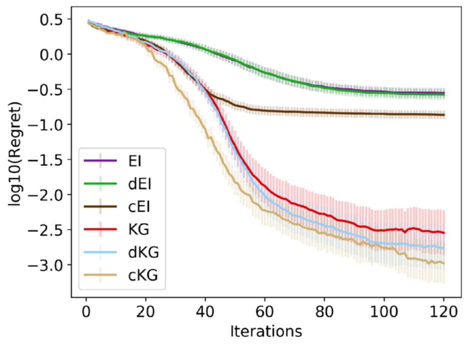
(b) Levy4 function: cEI outperforms EI in the initial phase and cKG outperforms all algorithms.



(c) Rosenbrock4 function: cEI and cKG outperforms EI, dEI and KG, dKG respectively



(d) Ackley5 function: cEI and cKG significantly outperforms existing methods.



(e) Hartmann6 function: cEI and cKG outperforms EI and KG respeectively.

**Table 2** Running time of algorithms in hours

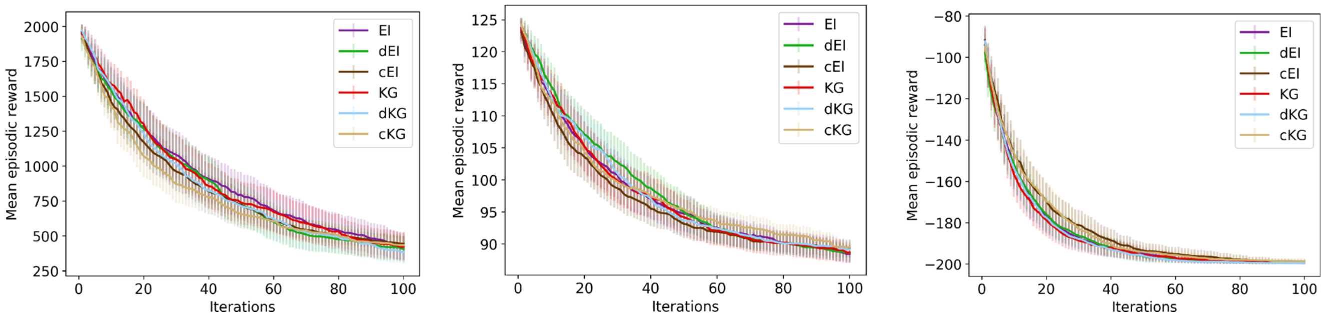
| Method (iterations) | EI | dEI | cEI | KG | dKG | cKG |
|---|---|---|---|---|---|---|
| Branin2 (120) | 0.19 | 0.19 | 0.20 | 12.08 | 12.12 | 12.13 |
| Levy4 (120) | 0.22 | 0.22 | 0.23 | 42.65 | 42.83 | 42.71 |
| Rosenbrock4 (120) | 0.21 | 0.21 | 0.23 | 41.92 | 42.13 | 42.15 |
| Ackley5 (120) | 0.20 | 0.20 | 0.22 | 28.77 | 30.23 | 28.83 |
| Hartmann6 (120) | 0.23 | 0.23 | 0.26 | 38.41 | 39.22 | 38.57 |
| MountainCar (100) | 12.11 | 12.21 | 12.13 | 15.49 | 16.03 | 15.57 |
| Acrobot (100) | 13.88 | 13.95 | 13.93 | 17.57 | 18.78 | 17.69 |
| CartPole (100) | 3.43 | 3.54 | 3.49 | 7.80 | 7.93 | 7.83 |

As the computational cost of the joint GP models grows with the dimension, we could not utilize all the partial derivative information in the existing FOBO methods. We thus randomly choose 3 partial derivatives for the gradient GP model in **dEI** and **dKG** methods. In plots Fig. 2a–e, x-axis corresponds to number of iterations and y-axis corresponds to logarithm (log10) of *immediate regret*. Note that immediate regret is defined as the absolute value of the difference between global optimum and the solution obtained by the BO method. In the plots, we average the results of all the BO methods over 250 runs.

The utility of the candidate points suggested by the acquisition algorithm where partial derivatives vanish can be seen from the plots (b–e) in Fig. 2. Here, **cEI** clearly outperforms **EI** algorithm and similarly **cKG** outperforms **KG** algorithm. This shows that the candidate points suggested by our acquisition algorithm are indeed useful, otherwise the performance would be similar to **EI** and **KG** algorithm. In the case of Branin2 function Fig. 2a, where the performance of **cEI** is not as good as **EI**, it is important to note that both algorithms have closed around optima, which gives a regret of 0.01, i.e., error between the true value and algorithm's solution value differ only in two decimal places. **cEI** explores in this scenario because Branin function has flat regions (gradient is close to zero) around the region of global maxima and thus **EI** wins over **cEI** as it only cares about the function value and not the gradient. However, again in the case of Branin function, **cKG** explores faster and thus better than **KG** algorithm in the initial phase. So the benefit of our acquisition algorithm will also depend on the kind of acquisition function used in the function GP model. Essentially, we emphasize that utlizing gradient information (like in our proposed approach) ensures BO algorithms to quickly converge close to the region of optima. This is very desirable behaviour as we only get to perform limited evaluations. If there are flat regions around optima, our FOBO algorithms will incur very small regret. To summarize, for Rosenbrock, Ackley and Hartmann functions (Fig. 2c, d, and e), **cEI** and **cKG** perform better than their counterparts by effectively capitalizing the gradient information. The KG family of algorithms (**KG**, **dKG** and **cKG**) performs better than the EI family (**EI**, **dEI** and **cEI**) for most of the test functions that we have considered. However, this comes with a high computational overhead and is clearly illustrated in Table 2. Note the numbers in the table are only an approximate measure of the computational effort required by different algorithms. From Table 2, it can be seen that the EI based acquisition methods are nearly 100 times faster than the KG methods. In summary, by utilizing our novel acquisition algorithm (16)–(17), **cEI** is able to achieve faster exploration, i.e., fast decrease in regret with significantly less computational requirement and **cKG** performs better than all the existing BO methods.

### 5.1.2 RL tasks

We have considered three benchmark episodic RL tasks namely CartPole, AcrobotSwing and MountainCar. In CartPole task, the goal is to balance a pole as long as possible by suitably moving the cart left or right. In the AcrobotSwing task, the goal of the controller is to optimally (in the minimum time possible) swing the foot of the acrobot over a specified threshold by suitably applying torque. In the MountainCar task, the goal is to optimally (in the minimum



(a) MountainCar task (b) Acrobot task (c) CartPole task

**Fig. 3** Performance comparison of BO methods on benchmark RL tasks

time possible) drive an under-powered car starting from a position at the base of a hill to its apex by leveraging the potential energy. For more details on these tasks, please refer to [35]. In all these tasks, the goal of the agent is to choose state dependent actions (typically known as policy) that maximizes the expected total reward collected during an episode. We evaluate the performance of various BO methods on these tasks.

For the RL tasks, we parameterize our policies using a single hidden layer neural network (5 neurons) and use ReLU nonlinearity [23] for activations in the hidden neurons. For each policy parameter update, we obtain estimates of expected return and its gradient by averaging over $E = 2000$ episodes. Under this setup, we measure and compare the mean episodic reward of various BO methods. In plots Fig. 3a–c, the x-axis corresponds to the number of iterations and the y-axis corresponds to the mean episodic reward. We have used the openAI gym environments for evaluating BO methods on all the three tasks.

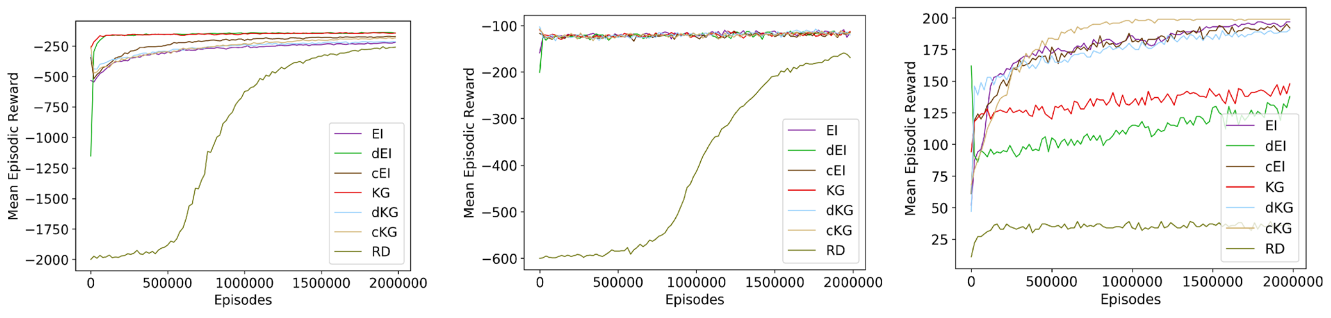
In RL tasks, all BO methods perform equally well. Our BO methods **cKG** and **cEI** marginally perform better than other methods in the MountainCar and AcrobotSwing tasks respectively. Note that the RL tasks that we have considered have 'highly noisy' gradient information which limits the advantage of the FOBO methods that exploit gradient information. However, all the BO methods converge quickly (in few iterations) towards the optimal solution using less number of function evaluations (samples).

Policy gradient (PG) methods (that utilizes simple gradient descent on the policy gradient information) are highly sample inefficient and their convergence is sensitive to step-size schedules and initial sample points. Thus, we propose to utilize the policies learnt by the BO algorithms after 10 iterations as initializing policies for the PG method. Figure 4a, b and c depict the performance of the PG method with different initializations (based on the output of the BO methods) on the MountainCar, AcrobotSwing and CartPole tasks respectively. With random initialization (denoted as **RD** in Fig. 4a, b and c), PG method can be seen to remain almost constant for many episodes. Other initializations of PG with BO methods show significant performance improvement over **RD**. To summarize, BO methods are sample efficient but time consuming. On the other hand, PG methods are fast but sample inefficient. Thus, one can strike a balance by utilizing the policies obtained from the BO methods as an initialization for the PG method.

## 6 Conclusions & future work

In this paper, we presented a relaxed statistical model for utilizing gradient information in the framework of BO. Further, utilizing our model we developed FOBO methods that are computationally faster and parallelizable. Through experiments on standard test functions, we compared the proposed FOBO methods (**cEI** and **cKG**) with existing BO methods. We observed that **cEI** runs faster and performs reasonably well and **cKG** outperforms the existing ZOBO and FOBO methods with computational overhead. Thus, it would be interesting to combine our FOBO methods with local optimization methods as in [21] for optimal exploration and exploitation. Another interesting line of research would be to devise more effective acquisition algorithms utilizing noisy gradient information directly without subjecting to computational burden.

In this work, we presented an interesting application of the BO methods to the episodic RL setting by utilizing policy gradient information. Our experiments indicate BO methods could be a promising initialization mechanism for many gradient based RL methods like PG. An interesting future research direction in the RL setting would be to extend our method for the long-run average reward objective criterion. Further, it would be more challenging to develop BORL methods based on other variants of the policy-based method like actor-critic algorithms, natural actor-critic and trust region policy optimization algorithms as the main challenge here would be to suitably transform these on-policy methods to the off-policy setting. An



(a) PG on MountainCar task

(b) PG on Acrobot task

(c) PG on CartPole task

**Fig. 4** Performance of Policy Gradient (PG) method with different initializations on different RL tasks

important and challenging research direction would be to perform regret analysis of the proposed method that uses gradient information along the lines similar to [34]. Another interesting research direction would be to utilize the proposed BO methods for text representation as in [42] and utilize the same in text clustering applications [1, 2].

## References

1. Abualigah LMQ (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Springer, Berlin
2. Abualigah LM, Khader AT, Hanandeh ES (2018) Hybrid clustering analysis using improved krill herd algorithm. Appl Intell 48(11):4047–4071
3. Ahmed MO, Shahriarim B, Schmidt M (2016) Do we need "harmless" bayesian optimization and "first-order" bayesian optimization. NIPS BayesOpt
4. Brochu E, Cora VM, De Freitas N (2010) A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv:1012.2599
5. Bull AD (2011) Convergence rates of efficient global optimization algorithms. J Mach Learn Res 12(10):1–27
6. Calandra R, Seyfarth A, Peters J, Deisenroth MP (2016) Bayesian optimization for learning gaits under uncertainty. Ann Math Artif Intell 76(1–2):5–23
7. Deisenroth M, Rasmussen CE (2011) Pilco: a model-based and data-efficient approach to policy search. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 465–472
8. Duffy AC (2009) An introduction to gradient computation by the discrete adjoint method, Tech. Rep.
9. Frazier PI, Powell WB, Dayanik S (2008) A knowledge-gradient policy for sequential information collection. SIAM J Control Optim 47(5):2410–2439
10. Fu J, Luo H, Feng J, Chua T-S (2016) Distilling reverse-mode automatic differentiation (drmad) for optimizing hyperparameters of deep neural networks, arXiv:1601.00917
11. Hernández-Lobato JM, Hoffman MW, Ghahramani Z (2014) Predictive entropy search for efficient global optimization of black-box functions. In: Advances in neural information processing systems, pp 918–926
12. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13(4):455–492
13. Kingma D, Adam JBA (2014) A method for stochastic optimization. arXiv:1412.6980
14. Koistinen OP, Maras E, Vehtari A, Jónsson H (2016) Minimum energy path calculations with Gaussian process regression. Nanosyst Phys Chem Math 7(6):925
15. Lizotte DJ (2008) Practical Bayesian optimization. University of Alberta
16. Lizotte DJ, Wang T, Bowling MH, Schuurmans D (2007) Automatic gait optimization with Gaussian process regression. In: IJCAI, vol 7, pp 944–949
17. Luketina J, Berglund M, Greff K, Raiko T (2016) Scalable gradient-based tuning of continuous regularization hyperparameters. In: International conference on machine learning, pp 2952–2960
18. Maclaurin D, Duvenaud D, Adams R (2015) Gradient-based hyperparameter optimization through reversible learning. In: International conference on machine learning, pp 2113–2122
19. Martinez-Cantin R (2017) Bayesian optimization with adaptive kernels for robot control. In: IEEE International conference on robotics and automation (ICRA). IEEE, pp 3350–3356
20. Martinez-Cantin R, de Freitas N, Brochu E, Castellanos J, Doucet A (2009) A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. Auton Rob 27(2):93–103
21. McLeod M, Osborne MA, Roberts SJ (2018) Optimization, fast and slow: optimally switching between local and Bayesian optimization, arXiv:1805.08610
22. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
23. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
24. O'Donoghue B, Munos R, Kavukcuoglu K, Mnih V (2016) Combining policy gradient and Q-learning. arXiv:1611.01626
25. Osborne MA, Garnett R, Roberts SJ (2009) Gaussian processes for global optimization. In: 3rd international conference on learning and intelligent optimization (LION3), pp 1–15
26. Peters J, Schaal S (2006) Policy gradient methods for robotics. In: IEEE/RSJ international conference on intelligent robots and systems, pp 2219–2225
27. Peters J, Schaal S (2008) Natural actor-critic. Neurocomputing 71(7):1180–1190
28. Plessix R-E (2006) A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. Geophys J Int 167(2):459–503
29. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms, arXiv:1707.06347
30. Sutton RS, McAllester DA, Singh SP, Mansour Y (2000) Policy gradient methods for reinforcement learning with function approximation. In: Advances in neural information processing systems, pp 1057–1063
31. Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning, vol 1. MIT Press, Cambridge
32. Rückstiess T, Sehnke F, Schaul T, Wierstra D, Sun Y, Schmidhuber J (2010) Exploring parameter space in reinforcement learning. Paladyn 1(1):14–24
33. Snoek J, Larochelle H, Adams RP (2012) Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp 2951–2959
34. Srinivas N, Krause A, Seeger M, Kakade SM (2010) Gaussian process optimization in the bandit setting: no regret and experimental design. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 1015–1022
35. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press, Cambridge
36. Tieleman T, Hinton G (2012) Lecture 6.5-rmsprop: divide the gradient by a running average of its recent magnitude. COURSERA: Neural Netw Mach Learn 4(2):26–31
37. Vazquez E, Bect J (2010) Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. J Stat Plan Inference 140(11):3088–3095
38. Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8(3–4):229–256
39. Wilson A, Fern A, Tadepalli P (2014) Using trajectory data to improve Bayesian optimization for reinforcement learning. J Mach Learn Res 15(1):253–282
40. Wu J, Poloczek M, Wilson AG, Frazier PI (2017) Bayesian optimization with gradients. In: Advances in neural information processing systems, pp 5267–5278

41. Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y (2015) Show attend and tell: neural image caption generation with visual attention. In: International conference on machine learning, pp 2048–2057
42. Yogatama D, Kong L, Smith NA (2015) Bayesian optimization of text representations. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 2100–2105

**Prabuchandran K. J.** is an Assistant Professor at IIT Dharwad. He completed Ph.D. from Department of Computer Science and Automation, IISc in the area of Reinforcement Learning. Post his PhD, Prabuchandran worked as Research Scientist at IBM Research Labs, India for an year and half on change detection algorithms for multivariate compositional data. After that he pursued his postdoctoral research at IISc, Bangalore as an Amazon-IISc Postdoctoral scholar for an year and half on Multi-agent Reinforcement Learning and Stochastic Optimization algorithms. His research lies in the intersection of reinforcement learning, stochastic control & optimization, Machine Learning, Bayesian Optimization and stochastic approximation algorithms. His research interest also focusses on utilizing techniques from these fields in solving problems arising in applications like wireless sensor networks, traffic signal control and social networks.

**Santosh Penubothula** is a Advisory Research Engineer at IBM Research Labs, India. He completed M.E. from Department of Computer Science and Automation, IISc in the area of Databases. Post his M.E., Santosh has been working at IBM Research Labs. His research lies in the intersection of Algorithms, Databases, Cryptography, Security and Machine Learning.

**Chandramouli Kamanchi** received the B.Tech degree in computer science and engineering from Indian Institute of Technology, Delhi. He received the M.S degree in mathematics from Indian Institute of Science, Bangalore. He is currently a graduate student in the Department of Computer Science and Automation at Indian Institute of Science, Bangalore. His research interests are in the areas of Optimization, Reinforcement Learning, Stochastic Approximation.

**Shalabh Bhatnagar** is currently a Professor in the Department of Computer Science and Automation with a joint appointment with the Robert Bosch Centre for Cyber Physical Systems at the Indian Institute of Science, Bangalore. His research interests are in stochastic approximation algorithms, simulation-based/ datadriven optimization and reinforcement learning. He is also interested in the applications of these algorithms and techniques to autonomous systems, control of micro-grids, as well as vehicular, communication and wireless networks. He is a Fellow of the Indian National Science Academy, the Indian Academy of Sciences, the Indian National Academy of Engineering, and is a J.C.Bose National Fellow.