BayesO: A Bayesian optimization framework in Python

Jungtaek Kim and Seungjin Choi

1 University of Pittsburgh, USA 2 Intellicode, South Korea

Summary

Bayesian optimization is a sample-efficient method for solving the optimization of a black-box function. In particular, it successfully shows its effectiveness in diverse applications such as hyperparameter optimization, automated machine learning, material design, sequential assembly, and chemical reaction optimization. In this paper we present an easy-to-use Bayesian optimization framework, referred to as BayesO, which is written in Python and licensed under the MIT license. To briefly introduce our software, we describe the functionality of BayesO and various components for software development.

Statement of need

Bayesian optimization (Brochu et al., 2010; Garnett, 2023; Shahriari et al., 2016) is a sample-efficient method for solving the optimization of a black-box function $f$:

$$x^* = \arg\min_{x \in \mathcal{X}} f(x) \quad \text{or} \quad x^* = \arg\max_{x \in \mathcal{X}} f(x),$$

(1)

where $\mathcal{X} \subset \mathbb{R}^d$ is a $d$-dimensional space. In general, finding a solution $x^*$ of Equation 1, i.e., a global optimum on $\mathcal{X}$, is time-consuming since we cannot employ any knowledge, e.g., gradients and Hessians, in solving this problem. Compared to other possible approaches, e.g., random search and evolutionary algorithm, Bayesian optimization successfully shows its effectiveness by utilizing a probabilistic regression model and an acquisition function. In particular, the sample-efficient approach of our interest enables us to apply it in various real-world applications such as hyperparameter optimization (Snoek et al., 2012), automated machine learning (Feurer et al., 2015, 2022), neural architecture search (Kandasamy et al., 2018), material design (Frazier & Wang, 2016), free-electron laser configuration search (Duris et al., 2020), organic molecule synthesis (Korovina et al., 2020), sequential assembly (Kim et al., 2020), and chemical reaction optimization (Shields et al., 2021).

In this paper, we present an easy-to-use Bayesian optimization framework, referred to as BayesO (pronounced “bayes-o”), to effortlessly utilize Bayesian optimization in the problems of interest to practitioners. Our BayesO is written in one of the most popular programming languages, Python, and licensed under the MIT license. Moreover, it provides various features including different types of input variables (e.g., vectors and sets (Kim et al., 2021)) and different surrogate models (e.g., Gaussian process regression (Rasmussen & Williams, 2006) and Student-t process regression (Shah et al., 2014)). Along with the description of such functionality, we cover various components for software development in the BayesO project.

We hope that this BayesO project encourages researchers and practitioners to readily utilize the powerful black-box optimization technique in diverse academic and industrial fields.
Bayesian optimization

As discussed in the work (Brochu et al., 2010; Garnett, 2023; Shahriari et al., 2016), supposing that a target objective function is black-box, Bayesian optimization is an approach to optimizing the objective in a sample-efficient manner. It repeats three primary steps:

1. Building a probabilistic regression model, which is capable of estimating the degrees of exploration and exploitation;
2. Optimizing an acquisition function, which is defined with the probabilistic regression model;
3. Evaluating the target objective at a query point, which is determined by optimizing the acquisition function,

until a predefined stopping criterion, e.g., an iteration budget or a budget of wall-clock time, is satisfied. Eventually, the best solution among the queries evaluated is selected by considering the function evaluations. As shown in Figure 1, Bayesian optimization iteratively finds a candidate of global optimum, repeating the aforementioned steps. Note that, for this example, an objective function is

\[ f(x) = 2 \sin(x) + 2 \cos(2x) + 0.05x, \]

(2)

where \( x \in [-5, 5] \), and Gaussian process regression and expected improvement are used as a surrogate model and an acquisition function, respectively.

Several related projects on Bayesian optimization have been proposed. GPyOpt (The GPyOpt authors, 2016), built on GPy (GPy, 2012), has been implemented. SMAC3 (Lindauer et al., 2022) has been developed using random forests (Breiman, 2001). Moreover, by making use of modern automatic differentiation tools, i.e., TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019), GPflowOpt (Knudde et al., 2017) and BoTorch (Balandat et al., 2020) have been introduced.
Overview of BayesO

In this section we cover the overview of BayesO including probabilistic regression models and acquisition functions, by referring to BayesO v0.5.4. For higher versions of BayesO, see official documentation. Note that the BayesO logo is shown in Figure 2.

BayesO supports the following probabilistic regression models:

- Gaussian process regression (Rasmussen & Williams, 2006);
- Student-t process regression (Shah et al., 2014);
- Random forest regression (Breiman, 2001).

Although random forest regression is not a probabilistic model inherently, we can compute its mean and variance functions as reported by Hutter et al. (2014).

We implement several acquisition functions:

- pure exploration;
- pure exploitation;
- probability of improvement (Kushner, 1964);
- expected improvement (Jones et al., 1998);
- augmented expected improvement (Huang et al., 2006);
- Gaussian process upper confidence bound (Srinivas et al., 2010).

As a strategy to optimize an acquisition function, we suggest the following options: DIRECT (Jones et al., 1993), CMA-ES (Hansen & Ostermeier, 1997), and L-BFGS-B (Byrd et al., 1995). In addition, we also include Thompson sampling (Thompson, 1933) in BayesO.

Furthermore, to support an easy-to-use interface, we implement the wrappers of Bayesian optimization for particular scenarios:

- a run with randomly-chosen initial inputs;
- a run with initial inputs provided;
- a run with initial inputs provided and their evaluations.

Software development for BayesO

To manage BayesO productively, we actively utilize external development management packages.

- Code analysis: Our software is monitored and inspected to satisfy the code conventions predefined in our software.
- Type hints: As supported in Python 3, we provide type hints for any arguments.
- Unit tests: Unit tests for our software are included. We have achieved 100% coverage. In addition, unit tests for measuring execution time are also provided.
- Dependency: Our package depends on NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), a quasi-Monte Carlo submodule in SciPy (Roy et al., 2023), pycma (Hansen et al., 2019), and tqdm (The tqdm authors, 2016).
- Installation: Our software is released via the Python Package Index (PyPI) meaning users can easily install BayesO into their environment.
- Documentation: We create official documentation with docstring.
Conclusion

In this work we have presented a Bayesian optimization framework, named BayesO. We hope that our project enables many researchers to suggest a new algorithm by modifying BayesO and many practitioners to utilize Bayesian optimization in their applications.

Acknowledgements

The BayesO project has been started when JK and SC were with POSTECH, and it has been mostly developed at POSTECH.

References


