BayesO: A Bayesian optimization framework in Python

Jungtaek Kim[1] **and Seungjin Choi**[2]

1 University of Pittsburgh, USA **2** Intellicode, South Korea

DOI: 10.21105/joss.05320

Software

- Review
- Repository
- Archive

Editor: Vincent Knight

Reviewers:

- @salrm8
- @thomaspinder

Submitted: 02 January 2023
Published: 09 October 2023

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:

$$\mathbf{x}^\star = \underset{\mathbf{x}\in\mathcal{X}}{\arg\min} f(\mathbf{x}) \quad \text{or} \quad \mathbf{x}^\star = \underset{\mathbf{x}\in\mathcal{X}}{\arg\max} f(\mathbf{x}), \tag{1}$$

where $\mathcal{X} \subset \mathbb{R}^d$ is a d-dimensional space. In general, finding a solution $\mathbf{x}^\star$ 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.

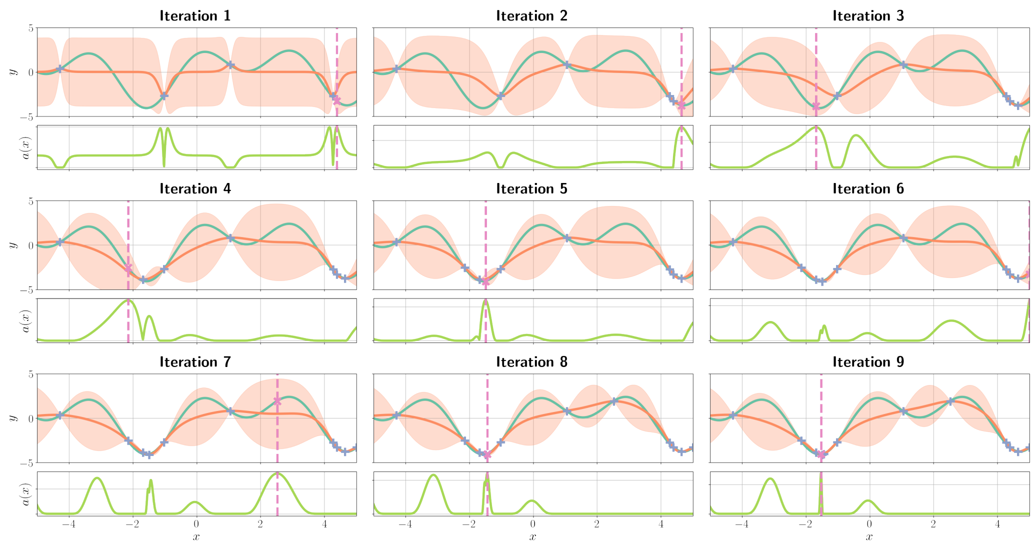

Figure 1: Visualization of Bayesian optimization procedure. Given an objective function, Equation 2 (colored by turquoise) and four initial points (denoted as light blue + at iteration 1), a query point (denoted as pink x) is determined by constructing a surrogate model (colored by orange) and maximizing an acquisition function (colored by light green) every iteration.

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

$$f(x) = 2\sin(x) + 2\cos(2x) + 0.05x, \qquad (2)$$

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

BayesO

Figure 2: Logo 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

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 265–283.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, *33*, 21524–21538.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brochu, E., Cora, V. M., & de Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv Preprint arXiv:1012.2599*.

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, *16*(5), 1190–1208. https://doi.org/10.1137/0916069

Duris, J., Kennedy, D., Hanuka, A., Shtalenkova, J., Edelen, A., Baxevanis, P., Egger, A., Cope, T., McIntire, M., Ermon, S., & Ratner, D. (2020). Bayesian optimization of a free-electron laser. *Physical Review Letters*, *124*(12), 124801. https://doi.org/10.1103/PhysRevLett.124.124801

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2022). Auto-Sklearn 2.0: Hands-free AutoML via meta-learning. *Journal of Machine Learning Research*, *23*(261), 1–61.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. *Advances in Neural Information Processing Systems (NeurIPS)*, *28*, 2962–2970.

Frazier, P. I., & Wang, J. (2016). Bayesian optimization for materials design. In *Information science for materials discovery and design* (pp. 45–75). Springer. https://doi.org/10.1007/978-3-319-23871-5_3

Garnett, R. (2023). *Bayesian Optimization*. Cambridge University Press.

GPy. (2012). *GPy: A Gaussian process framework in Python*. GitHub. https://github.com/SheffieldML/GPy

Hansen, N., Akimoto, Y., & Baudis, P. (2019). *CMA-ES/pycma on GitHub*. GitHub. https://github.com/CMA-ES/pycma

Hansen, N., & Ostermeier, A. (1997). Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_I, \lambda)$-CMA-ES. *Proceedings of the European Congress on Intelligent Techniques and Soft Computing (EUFIT)*, 650–654.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Huang, D., Allen, T. T., Notz, W. I., & Zheng, N. (2006). Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of Global Optimization*, *34*, 441–466. https://doi.org/10.1007/s10898-005-2454-3

Hutter, F., Xu, L., Hoos, H. H., & Leyton-Brown, K. (2014). Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, *206*, 79–111. https://doi.org/10.1016/j.artint.2013.10.003

Jones, D. R., Perttunen, C. D., & Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications*, *79*(1), 157–181. https://doi.org/10.1007/BF00941892

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*, 455–492. https://doi.org/10.1023/A:1008306431147

Kandasamy, K., Neiswanger, W., Schneider, J., Póczos, B., & Xing, E. P. (2018). Neural architecture search with Bayesian optimisation and optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*, *31*, 2016–2025.

Kim, J., Chung, H., Lee, J., Cho, M., & Park, J. (2020). Combinatorial 3D shape generation via sequential assembly. *Neural Information Processing Systems Workshop on Machine Learning for Engineering Modeling, Simulation, and Design (ML4Eng)*.

Kim, J., McCourt, M., You, T., Kim, S., & Choi, S. (2021). Bayesian optimization with approximate set kernels. *Machine Learning*, *110*(5), 857–879. https://doi.org/10.1007/s10994-021-05949-0

Knudde, N., van der Herten, J., Dhaene, T., & Couckuyt, I. (2017). GPflowOpt: A Bayesian optimization library using TensorFlow. *arXiv Preprint arXiv:1711.03845*.

Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Póczos, B., Schneider, J., & Xing, E. P. (2020). ChemBO: Bayesian optimization of small organic molecules with synthesizable recommendations. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 3393–3403.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, *86*(1), 97–106. https://doi.org/10.1115/1.3653121

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., & Hutter, F. (2022). SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, *23*(54), 1–9.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, *32*, 8026–8037.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Roy, P. T., Owen, A. B., Balandat, M., & Haberland, M. (2023). Quasi-Monte Carlo methods in Python. *Journal of Open Source Software*, *8*(84), 5309. https://doi.org/10.21105/joss.05309

Shah, A., Wilson, A. G., & Ghahramani, Z. (2014). Student-t processes as alternatives to Gaussian processes. *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 877–885.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, *104*(1), 148–175. https://doi.org/10.1109/JPROC.2015.2494218

Shields, B. J., Stevens, J., Li, J., Parasram, M., Damani, F., Alvarado, J. I. M., Janey, J. M., Adams, R. P., & Doyle, A. G. (2021). Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, *590*, 89–96. https://doi.org/10.1038/s41586-021-03213-y

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems (NeurIPS)*, *25*, 2951–2959.

Srinivas, N., Krause, A., Kakade, S., & Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. *Proceedings of the International Conference on Machine Learning (ICML)*, 1015–1022.

The GPyOpt authors. (2016). *GPyOpt: A Bayesian optimization framework in Python*. GitHub. https://github.com/SheffieldML/GPyOpt

The tqdm authors. (2016). *tqdm*. GitHub. https://github.com/tqdm/tqdm

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*(3/4), 285–294. https://doi.org/10.1093/biomet/25.3-4.285

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2