SCIENCE CHINA
Information Sciences

• NEWS & VIEWS •

October 2022, Vol. 65 207101:1–207101:2
https://doi.org/10.1007/s11432-021-3416-y

ZOOpt: a toolbox for derivative-free optimization

Yu-Ren LIU[1], Yi-Qi HU[1], Hong QIAN[2], Chao QIAN[1] & Yang YU[1*]

[1]*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023*, China;*
[2]*School of Computer Science and Technology, East China Normal University, Shanghai* 200062*, China*

Received 11 September 2021/Revised 21 November 2021/Accepted 9 December 2021/Published online 22 September 2022

Citation Liu Y-R, Hu Y-Q, Qian H, et al. ZOOpt: a toolbox for derivative-free optimization. Sci China Inf Sci, 2022, 65(10): 207101, https://doi.org/10.1007/s11432-021-3416-y

Recent advances in derivative-free optimization allow for efficient approximation of the global optimal solutions of sophisticated functions, such as those with many local optima, non-differentiable and non-continuous functions. This article describes the ZOOpt (zeroth order optimization) toolbox that provides efficient derivative-free solvers that are designed for easy use. ZOOpt provides single-machine parallel optimization based on the python core and multi-machine distributed optimization for time consuming tasks by incorporating the Ray framework [1] — a famous platform for building distributed applications. ZOOpt particularly focuses on the optimization problems in machine learning, addressing high-dimensional and noisy problems such as hyperparameter tuning and direct policy search. The toolbox is developed as a ready-to-use tool in real-world machine learning tasks.

Classification-based optimization. Derivative-free optimization, also termed zeroth-order or black-box optimization, involves a class of optimization algorithms that do not rely on gradient information. Model-based derivative-free optimization algorithms share a framework that iteratively learns a model for promising search areas and sample solutions from the model. Different kinds of methods usually vary in the model's design. Classification-based optimization algorithms learn a particular type of model: classification model, leading to theoretically grounded properties of optimization performance. A classification model learns to classify solutions into two categories, good or bad. Then solutions are sampled from the good areas. SRacos [2] is a recently proposed classification-based optimization algorithm. Unlike other model-based optimization algorithms, the sampling region of SRacos is learned by a simple classifier, which maintains an axis-parallel rectangle to cover all positive but no negative solutions. SRacos shows outstanding performance in empirical studies. To support machine learning tasks, ZOOpt implements a set of classification-based methods that are efficient and performance-guaranteed, with add-ons handling noise and high-dimensionality.

Optimization in the continuous, discrete, or hybrid space. We implement SRacos [2] as the default optimization method, which has shown high efficiency in a range of learning tasks. Optional methods are Racos [3] and ASRacos [4], respectively, which are the batch and asynchronous versions of SRacos. A routine is in place to set up the default parameters of the two methods, while users can override them. Benefitting from the compatibility of the classifier with multiple data types, classification-based optimization naturally supports optimization in the continuous, discrete (categorical), or hybrid space.

Optimization in the binary vector space with constraint. If the optimization task is in the binary vector space with constraints, such as the subset selection problem, POSS [5] is the default optimization method. POSS treats the subset selection task as a bi-objective optimization problem that simultaneously optimizes some given criterion and the subset size. POSS has been proven to have the best-so-far approximation quality on these problems. PPOSS [6] is the parallel version of the POSS algorithm.

Noise handling. Noise has a great impact on the performance of derivative-free optimization. Resampling is the most straightforward method to reduce noise, which evaluates one sample several times to obtain a stable mean value. Besides resampling, more efficient methods, including value suppression [7] and threshold selection [8] are implemented.

High-dimensionality handling. An increase in the search space dimensionality badly injures the performance of derivative-free optimization. When a high dimensional search space has a low effective dimension, random embedding [9] is an effective way to improve the efficiency. Moreover, the sequential random embeddings [10] can be used when there is no clear low effective dimension.

Distributed optimization. Evaluating a sampled solution is usually time-consuming for many real-world optimization tasks, such as hyperparameter tuning in large-scale machine learning projects. By incorporating the Ray framework [1], ZOOpt implements an efficient distributed optimization module that enables users to parallelize single-machine code with little to zero code changes.

Usage. A key point we considered when designing ZOOpt was to make its usage as easy as possible. After defining a user-specified objective function and the correspond-

* Corresponding author (email: yuy@nju.edu.cn)

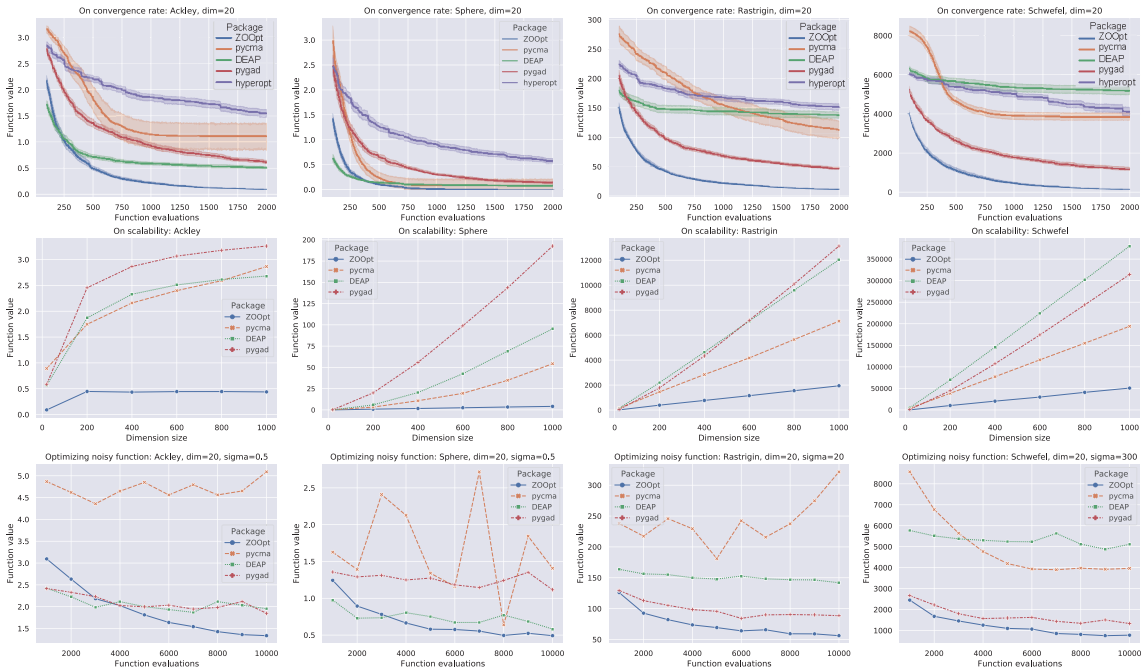

Figure 1 (Color online) Evaluate the optimization (minimization) performance of ZOOpt on four synthetic benchmark functions. Among these, the Ackley, Rastrigin, and Schwefel functions are highly nonconvex whereas the Sphere function is convex. The optimal values of the four functions are all zero. For all toolboxes, we choose the recommended parameters according to their tutorials. The top row shows the convergence rate of the tested toolboxes. The middle row shows the scalability of the tested toolbox as the dimension size increases (the number of evaluations is set to be 20 $\times$ dim_size for each run). The bottom row demonstrates the performance of optimizing noisy functions (a Gaussian noise $N(0, \sigma^2)$ is added when evaluating the objective functions). Each experiment is repeated 30 times, and mean values and 95% confidence intervals are recorded. For experiments on scalability and optimizing noisy functions, the confidence interval is omitted for clarity. It can be observed that ZOOpt achieves the best performance in all tasks.

ing search space, only one line of code was needed to conduct optimization using the ZOOpt optimization interface Opt.min. To use some advanced functionality, such as the noise handler and the high-dimensionality handler, a more detailed Parameter object can be defined and passed as an attribute to Opt.min. Users can customize their optimizer by performing fine-grained control on these parameters. Distributed optimization in ZOOpt is implemented by incorporating Ray. Currently, ZOOpt is an optional optimization tool in Ray.tune — a library for fast hyperparameter tuning at any scale. Users only need to provide extra server IP addresses to distribute the optimization without caring about the communication infrastructure. For concrete examples, we refer readers to the project homepage[1)].

Access methods. ZOOpt can be downloaded from the website[1)] or be installed directly from PyPi (pip install zoopt). The full version of this paper can be found from the website[2)].

References

1 Moritz P, Nishihara R, Wang S, et al. Ray: a distributed framework for emerging AI applications. In: Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation, Carlsbad, 2018. 561–577

2 Hu Y Q, Qian H, Yu Y. Sequential classification-based optimization for direct policy search. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, 2017. 2029–2035

3 Yu Y, Qian H, Hu Y Q. Derivative-free optimization via classification. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, 2016. 2286–2292

4 Liu Y R, Hu Y Q, Qian H, et al. Asynchronous classification-based optimization. In: Proceedings of the 1st International Conference on Distributed Artificial Intelligence, Beijing, 2019

5 Qian C, Yu Y, Zhou Z H. Subset selection by pareto optimization. In: Proceedings of Advances in Neural Information Processing Systems, Montreal, 2015. 1765–1773

6 Qian C, Shi J C, Yu Y, et al. Parallel pareto optimization for subset selection. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence, New York, 2016. 1939–1945

7 Wang H, Qian H, Yu Y. Noisy derivative-free optimization with value suppression. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, 2018. 1447–1454

8 Qian C, Shi J C, Yu Y, et al. Subset selection under noise. In: Proceedings of Advances in Neural Information Processing Systems, Long Beach, 2017. 3563–3573

9 Wang Z Y, Hutter F, Zoghi M, et al. Bayesian optimization in a billion dimensions via random embeddings. J Artif Intell Res, 2016, 55: 361–387

10 Qian H, Hu Y Q, Yang Yu Y. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence, New York, 2016. 1946–1952

1) https://github.com/polixir/ZOOpt.
2) https://arxiv.org/abs/1801.00329.