The Journal of Open Source Software

PyBADS: Fast and robust black-box optimization in Python

Gurjeet Sangra Singh [1,3¶] **and Luigi Acerbi** [2¶]

1 University of Geneva **2** University of Helsinki **3** University of Applied Sciences and Arts Western Switzerland (HES-SO) ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Rachel Kurchin

Reviewers:

- @gaxler
- @jungtaekkim

Submitted: 11 June 2023
Published: unpublished

Summary

PyBADS is a Python implementation of the Bayesian Adaptive Direct Search (BADS) algorithm for fast and robust *black-box* optimization (Acerbi & Ma, 2017), first developed in MATLAB[1]. BADS is an optimization algorithm designed to efficiently solve difficult optimization problems where the objective function is rough (non-convex, non-smooth), mildly expensive (e.g., the function evaluation requires more than 0.1 seconds), possibly noisy, and gradient information is unavailable. With BADS, these issues are well addressed, making it an excellent choice for fitting computational models using methods such as maximum-likelihood estimation. The algorithm scales efficiently to black-box functions with up to $D \approx 20$ continuous input parameters and supports bounds or no constraints. PyBADS comes along from the previous MATLAB code-base implementation with an easy-to-use Pythonic interface for running the algorithm and inspecting its results. PyBADS only requires the user to provide a Python function for evaluating the target function, and optionally other constraints.

Extensive benchmarks on both artificial test problems and large real model-fitting problems models drawn from cognitive, behavioral and computational neuroscience, show that BADS performs on par with or better than many other common and state-of-the-art optimizers (Acerbi & Ma, 2017), making it a general model-fitting tool which provides fast and robust solutions.

Statement of need

Many optimization problems in science and engineering involve complex and expensive simulations or numerical approximations such that the target function can only be evaluated at a point with moderate to high cost, possibly yielding stochastic outcomes, and gradients are unavailable (or exceedingly expensive) – the typical *black-box* scenario. There is a large landscape of derivative-free optimization algorithms for tackling black-box problems (Rios & Sahinidis, 2013), many of which follow variants of direct-search methods (Abramson et al., 2009; Audet et al., 2021; Audet & Dennis, 2006; Deng & Ferris, 2006). Despite their theoretical guarantees, direct search methods require a large number of function evaluations and have limited support for handling stochastic targets.

Conversely, Bayesian Optimization (BayesOpt) is a recently popular family of methods that has shown effectiveness in solving *very* costly black-box problems in machine learning and engineering with very few, possibly noisy, function evaluations (Agnihotri & Batra, 2020; Garnett, 2023; Shahriari et al., 2016). However, classical BayesOpt requires specific technical knowledge to be implemented or tuned beyond simple tasks, since vanilla BayesOpt applied to complex real-world problems can be strongly affected by deviations from the algorithm's

[1] github.com/acerbilab/bads

assumptions (model misspecification), a problem rarely dealt with in current implementations. Moreover, traditional BayesOpt methods assume *highly expensive* target functions (e.g., with evaluation costs of hours or more), whereas many computational models might only have a *moderate* evaluation cost (e.g., from a fraction of a second to a few seconds), meaning that the optimization algorithm should add only a relatively small overhead.

PyBADS addresses all these problems as a fast hybrid algorithm that combines the strengths of BayesOpt and the Mesh Adaptive Direct Search (Audet & Dennis, 2006) method. In contrast to other black-box optimization algorithms, PyBADS is both *fast* in terms of wall-clock time and *sample-efficient* in terms of the number of target evaluations (typically of the order of a few hundred), with support for noisy targets. Moreover, PyBADS does not require any specific tuning and runs off-the-shelf with its well-modularized Python API.

Method

PyBADS follows the Mesh Adaptive Direct Search (MADS, Audet & Dennis, 2006) schema for minimizing the given objective function. The algorithm alternates between a series of fast local Bayesian Optimization (BayesOpt) steps, referred to as *search stage*, and systematic exploration of the mesh space in a neighborhood of the current point, known as *poll stage*, based on the MADS poll method (Audet & Dennis, 2006); see Figure 1. Briefly:

- In the poll stage, points are evaluated on a mesh by taking steps in one (non-orthogonal) direction at a time, until an improvement is found or all directions have been tried. The step size is doubled in case of success, halved otherwise.
- In the search stage, a Gaussian process (GP) surrogate model (Rasmussen & Williams, 2006) of the target function is fit to a local subset of the points evaluated so far. New points to evaluate are quickly chosen according to a *lower confidence bound* strategy that trades off between exploration of uncertain regions (high GP uncertainty) and exploitation of promising solutions (low GP mean). The search switches back to the poll stage after repeated failures to find an improvement over the current point.

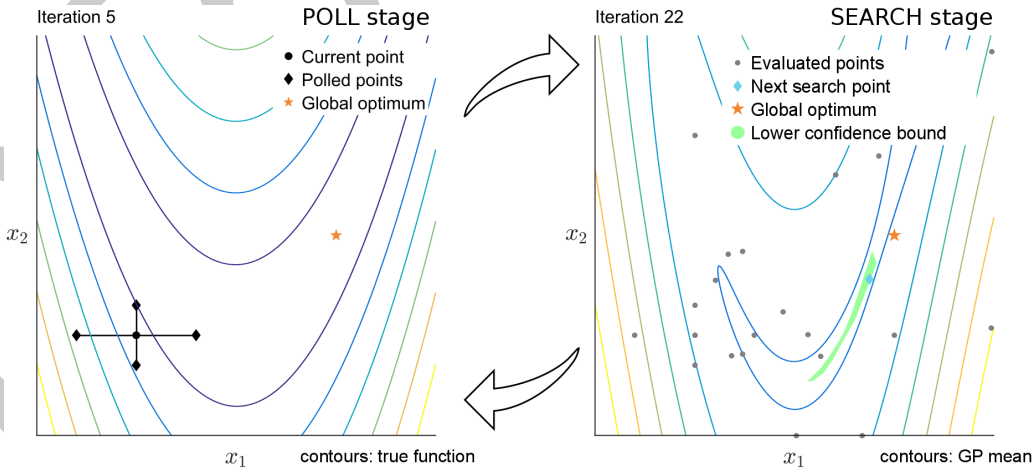

Figure 1: Contour plots and PyBADS exploration of a two-dimensional Rosenbrock function. Lines represent the contours of the true target (left) and of the GP surrogate model built during the search stage (right). The solid black diamonds indicate new points chosen by the poll method (here for simplicity a simple orthogonal poll), the grey circles represent the previously sampled points, and the orange solid star represents the global minimum of the function. When switching to the search stage, the blue diamond describes the point selected by the active sampling method based on the lower confidence bound obtained from the GP surrogate model (green region).

This alternation between the two stages makes BADS uniquely robust and effective. The poll stage follows a slow but steady "model-free" optimization with theoretical guarantees.

Conversely, the search stage exploits a powerful "model-based" GP surrogate to propose potentially large steps, which can be extremely effective if the surrogate is able to approximate the target well. Notably, this strategy is fail-safe in that if the GP fails to locally model the target, the search will fail and PyBADS will fall back to the safer poll method. The points acquired during the poll will afford a construction of a better surrogate at the next search, and so on. In addition, when the target is noisy, BADS follows some effective heuristics for calibrating the surrogate model, checking the reliability of the predictions, and reassessing the estimated value of the current point in light of the new points.

Thanks to these techniques, our algorithm has demonstrated high robustness and effectiveness in solving optimization problems with noisy and complex non-convex objective functions.

Related work

Similarly to PyBADS, relevant libraries have been developed over the years in the area of Bayesian Optimization, such as BoTorch (Balandat et al., 2020), GPflowOpt (Knudde et al., 2017), Spearmint (Snoek et al., 2014), among others. Instead, NOMAD (Audet et al., 2022) is the main reference library for *pattern search* algorithms, and it implements several variants of MADS in C++, by providing Python and Julia interface bindings.

Differently to these algorithms, PyBADS comes with a unique hybrid, fast and robust combination of direct search (MADS) and Bayesian Optimization. This combination of strategies protects against failures of the GP surrogate models – whereas vanilla BayesOpt does not have such fail-safe mechanisms, and can be strongly affected by misspecification of the surrogate GP model. PyBADS has also been designed to avoid problem-specific tuning, making it a generic tool for model fitting. Compared to other approaches, PyBADS also has the advantage of natively accommodating target functions with heteroskedastic (input-dependent) observation noise. The results of our approach demonstrate that a hybrid Bayesian approach to optimization can be beneficial beyond the domain of costly black-box functions. Finally, unlike most other BayesOpt packages, targeted to an audience of machine learning researchers, PyBADS comes with a neat API library and well-structured, user-friendly documentation.

PyBADS was developed in parallel to PyVBMC, a new software for sample-efficient Bayesian inference (Acerbi, 2018, 2019, 2020; Huggins et al., 2023). PyBADS can be used in combination with PyVBMC, by providing an effective way of initializing the inference algorithm at the maximum-a-posteriori (MAP) solution.

Applications and usage

The BADS algorithm, in its MATLAB implementation, has already been applied in multiple fields, especially in neuroscience where it finds a broad audience by efficiently solving difficult model-fitting problems (Cao et al., 2019; Daube et al., 2019; J.-A. Li et al., 2020; Tajima et al., 2019). Other fields in which BADS has been successfully applied include control engineering (Stenger & Abel, 2022), electrical engineering (M. Li et al., 2022), material engineering (Ren et al., 2021), robotics (Ren et al., 2020), petroleum science (Feng et al., 2022), environmental economics (Nobel et al., 2020), and cognitive science (Stengård et al., 2022; van Opheusden et al., 2023). Moreover, BADS has been shown to perform best in most settings of a black-box optimization benchmark for control engineering (Stenger & Abel, 2022), highlighting the effectiveness of our algorithm compared to other BayesOpt and direct-search approaches. With PyBADS, we bring the same sample-efficient and robust optimization to the wider open-source Python community, while improving the interface, test coverage, and documentation.

The package is available on both PyPI (`pip install pybads`) and `conda-forge`, and provides an idiomatic and accessible interface, only depending on standard, widely available scientific Python packages (Harris et al., 2020; Virtanen et al., 2020). The user only needs to give a few basic details about the objective function and its parameter space, and PyBADS handles the rest of the optimization task. PyBADS includes automatic handling of bounded variables,

robust termination conditions, sensible default settings, and does not need tunable parameters. At the same time, experienced users can easily supply their own options. We have extensively tested the algorithm and implementation details for correctness and performance. We provide detailed tutorials, so that PyBADS may be accessible to those not already familiar with black-box optimization, and our comprehensive documentation will aid not only new users but future contributors as well.

Acknowledgments

We thank Bobby Huggins, Chengkun Li, Marlon Tobaben and Mikko Aarnos for helpful comments and feedback. Work on the PyBADS package is supported by the Academy of Finland Flagship programme: Finnish Center for Artificial Intelligence FCAI.

References

Abramson, M. A., Audet, C., & Digabel, S. L. (2009). OrthoMADS: A deterministic MADS instance with orthogonal directions. *SIAM Journal on Optimization*, *20*(2), 948–966. https://doi.org/10.1137/080716980

Acerbi, L. (2018). Variational Bayesian Monte Carlo. *Advances in Neural Information Processing Systems*, *31*, 8222–8232. https://proceedings.neurips.cc/paper_files/paper/2018/file/747c1bcceb6109a4ef936bc70cfe67de-Paper.pdf

Acerbi, L. (2019). An exploration of acquisition and mean functions in Variational Bayesian Monte Carlo. *PMLR*, *96*, 1–10. https://proceedings.mlr.press/v96/acerbi19a.html

Acerbi, L. (2020). Variational Bayesian Monte Carlo with noisy likelihoods. *Advances in Neural Information Processing Systems*, *33*, 8211–8222. https://proceedings.neurips.cc/paper_files/paper/2020/file/5d40954183d62a82257835477ccad3d2-Paper.pdf

Acerbi, L., & Ma, W. J. (2017). Practical Bayesian optimization for model fitting with Bayesian adaptive direct search. *Advances in Neural Information Processing Systems*, *30*, 1834–1844. https://proceedings.neurips.cc/paper_files/paper/2017/file/df0aab058ce179e4f7ab135ed4e641a9-Paper.pdf

Agnihotri, A., & Batra, N. (2020). Exploring bayesian optimization. *Distill*. https://doi.org/10.23915/distill.00026

Audet, C., & Dennis, J. (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, *17*, 188–217. https://doi.org/10.1137/040603371

Audet, C., Dzahini, K. J., Kokkolaras, M., & Le Digabel, S. (2021). Stochastic mesh adaptive direct search for blackbox optimization using probabilistic estimates. *Computational Optimization and Applications*, *79*(1), 1–34. https://doi.org/10.1007/s10589-020-00249-0

Audet, C., Le Digabel, S., Montplaisir, V. R., & Tribes, C. (2022). Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, *48*(3). https://doi.org/10.1145/3544489

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 33*. http://arxiv.org/abs/1910.06403

Cao, Y., Summerfield, C., Park, H., Giordano, B. L., & Kayser, C. (2019). Causal inference in the multisensory brain. *Neuron*, *102*(5), 1076–1087.e8. https://doi.org/10.1016/j.neuron.2019.03.043

Daube, C., Ince, R. A. A., & Gross, J. (2019). Simple acoustic features can explain phoneme-based predictions of cortical responses to speech. *Current Biology*, *29*(12), 1924–1937.e9. https://doi.org/10.1016/j.cub.2019.04.067

Deng, G., & Ferris, M. C. (2006). Adaptation of the UOBYQA algorithm for noisy functions. *Proceedings of the 2006 Winter Simulation Conference*, 312–319. https://doi.org/10.1109/wsc.2006.323088

Feng, Q.-H., Li, S.-S., Zhang, X.-M., Gao, X.-F., & Ni, J.-H. (2022). Well production optimization using streamline features-based objective function and bayesian adaptive direct search algorithm. *Petroleum Science*, *19*(6), 2879–2894. https://doi.org/10.1016/j.petsci.2022.06.016

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

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362.

Huggins, B., Li, C., Tobaben, M., Aarnos, M. J., & Acerbi, L. (2023). PyVBMC: Efficient bayesian inference in python. *Journal of Open Source Software*, *8*(86), 5428. https://doi.org/10.21105/joss.05428

Knudde, N., van der Herten, J., Dhaene, T., & Couckuyt, I. (2017). GPflowOpt: A Bayesian Optimization Library using TensorFlow. *arXiv Preprint – arXiv:1711.03845*. https://arxiv.org/abs/1711.03845

Li, J.-A., Dong, D., Wei, Z., Liu, Y., Pan, Y., Nori, F., & Zhang, X. (2020). Quantum reinforcement learning during human decision-making. *Nature Human Behaviour*, *4*(3), 294–307. https://doi.org/10.1038/s41562-019-0804-2

Li, M., Yu, P., Wang, Y., Sun, Z., & Chen, Z. (2022). Topology comparison and sensitivity analysis of fuel cell hybrid systems for electric vehicles. *IEEE Transactions on Transportation Electrification*. https://doi.org/10.1109/TTE.2022.3218341

Nobel, A., Lizin, S., Witters, N., Rineau, F., & Malina, R. (2020). The impact of wildfires on the recreational value of heathland: A discrete factor approach with adjustment for on-site sampling. *Journal of Environmental Economics and Management*, *101*, 102317. https://doi.org/10.1016/j.jeem.2020.102317

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press. ISBN: 978-0-262-18253-9

Ren, X., Chai, Z., Xu, J., Zhang, X., He, Y., Chen, H., & Chen, X. (2020). A new method to achieve dynamic heat input monitoring in robotic belt grinding of inconel 718. *Journal of Manufacturing Processes*, *57*, 575–588. https://doi.org/10.1016/j.jmapro.2020.07.018

Ren, X., Huang, X., Feng, H., Chai, Z., He, Y., Chen, H., & Chen, X. (2021). A novel energy partition model for belt grinding of inconel 718. *Journal of Manufacturing Processes*, *64*, 1296–1306. https://doi.org/10.1016/j.jmapro.2021.02.052

Rios, L. M., & Sahinidis, N. V. (2013). Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization*, *56*(3), 1247–1293. https://doi.org/10.1007/s10898-012-9951-y

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & Freitas, N. de. (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

Snoek, J., Swersky, K., Zemel, R., & Adams, R. (2014). Input warping for bayesian optimization of non-stationary functions. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st*

international conference on machine learning (Vol. 32, pp. 1674–1682). PMLR.

Stengård, E., Juslin, P., Hahn, U., & van den Berg, R. (2022). On the generality and cognitive basis of base-rate neglect. *Cognition*, *226*, 105160. https://doi.org/10.1101/2021.03.11.434913

Stenger, D., & Abel, D. (2022). *Benchmark of bayesian optimization and metaheuristics for control engineering tuning problems with crash constraints*. https://arxiv.org/abs/2211.02571

Tajima, S., Drugowitsch, J., Patel, N., & Pouget, A. (2019). Optimal policy for multi-alternative decisions. *Nature Neuroscience*, *22*(9), 1503–1511. https://doi.org/10.1038/s41593-019-0453-9

van Opheusden, B., Kuperwajs, I., Galbiati, G., Bnaya, Z., Li, Y., & Ma, W. J. (2023). Expertise increases planning depth in human gameplay. *Nature*. https://doi.org/10.1038/s41586-023-06124-2

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Contributors, S. 1.0. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2