Bayesian Adaptive Direct Search: Hybrid Bayesian Optimization for Model Fitting

Luigi Acerbi
Center for Neural Science
New York University
luigi.acerbi@nyu.edu

Wei Ji Ma
Center for Neural Science & Dept. of Psychology
New York University
weijima@nyu.edu

Abstract

Model fitting in fields such as computational neuroscience often implies a difficult black-box optimization problem over complex, possibly noisy parameter landscapes. Bayesian optimization (BO) has been successfully applied to solving costly black-box problems in machine learning and engineering. Here we explore BO as a general tool for scientific model fitting. First, we present a novel hybrid BO algorithm, Bayesian adaptive direct search (BADS), that combines BO with a direct search framework. BADS achieves competitive performance at a small computational cost. We then perform an extensive benchmark of BADS vs. many common and state-of-the-art derivative-free optimizers, on a set of real model-fitting problems from computational neuroscience. With default settings, BADS generally outperforms other methods, including 'vanilla' BO, showing great promise for advanced BO techniques, and BADS in particular, as a general model-fitting tool in computational neuroscience and related fields.

1 Introduction

Many complex, nonlinear computational models in fields such as behaviorial, cognitive, and computational neuroscience cannot be evaluated analytically, but require moderately expensive numerical approximations or simulations [1].[1] In these cases, finding the maximum-likelihood (ML) solution – for parameter estimation, or model selection – requires the costly exploration of a rough or noisy nonconvex landscape, in which gradients are often unavailable to guide the search. Formally, we consider the problem of finding the (global) optimum $\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x}\in\mathcal{X}}\mathbb{E}\left[f(\boldsymbol{x})\right]$ of a possibly noisy *objective* f over a (bounded) domain $\mathcal{X} \subseteq \mathbb{R}^D$, where the black-box function f can be intended as the (negative) log likelihood of a parameter vector $\boldsymbol{x}$ for a given dataset and model.

Bayesian optimization (BO) is a state-of-the-art machine learning framework for optimizing expensive and possibly noisy black-box functions [2, 3, 4]. This makes it an ideal candidate for solving difficult model-fitting problems. Yet there are several obstacles to a widespread usage of BO as a general tool for scientific model fitting. First, standard BO methods target *very* costly problems, such as hyperparameter tuning [5], whereas, for instance, typical behavioral models might only have a moderate computational cost (e.g., 0.1-10 s per evaluation). This implies major differences in what is considered an acceptable algorithmic overhead, and in the maximum number of allowed function evaluations (e.g., hundreds vs. thousands). Second, it is unclear how BO methods would fare in this regime – in terms of efficiency, usability, and robustness – against commonly used and state-of-the-art, non-Bayesian optimizers. Finally, BO might be perceived by non-practitioners as an advanced tool that requires specific technical knowledge to be implemented or tuned.

We addressed these issues by developing a novel hybrid BO algorithm, Bayesian Adaptive Direct Search (BADS), that achieves competitive performance at a small computational cost. We tested

[1] This paper is a shorter version of [1], modified for the NIPS workshop on Bayesian optimization (*BayesOpt*).

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

BADS, together with a wide array of commonly used optimizers, on a novel benchmark set of model-fitting problems with real data and models drawn from studies in cognitive, behaviorial and computational neuroscience. Finally, we made BADS available as a free MATLAB package that can be used out-of-the-box with no tuning.[2] BADS proves to be highly competitive on both artificial functions and real-world model-fitting problems, showing promise as a general tool for model fitting in computational neuroscience and potentially other scientific fields.

Related work There is a large literature about (Bayesian) optimization of expensive, possibly stochastic, computer simulations, mostly used in machine learning [3, 4, 5] or engineering (known as *kriging-based* optimization) [6, 7, 8]. Recent work has combined MADS with treed Gaussian process (GP) models for constrained optimization (TGP-MADS [8]). Crucially, these methods have large overheads and may require problem-specific tuning, making them impractical as a generic tool for model fitting. Cheaper but less precise surrogate models than GPs have been proposed, such as random forests [9], Parzen estimators [10], and dynamic trees [11]. In this paper, we focus on BO based on traditional GP surrogates, leaving the analysis of alternative models for future work.

2 Bayesian adaptive direct search (BADS)

BADS is a *hybrid* BO method in that it combines the mesh adaptive direct search (MADS) framework [12] with BO based on a local GP surrogate, implemented via a number of heuristics for efficiency. BADS alternates between a series of fast, local BO steps (the SEARCH stage of MADS) and a systematic, slower exploration of the mesh grid (POLL stage). See Algorithm 1, and [1] for details.

Mesh adaptive direct search (MADS) The MADS algorithm is a directional direct search framework for nonlinear optimization [12, 13]. Briefly, MADS seeks to improve the current solution by testing points in the neighborhood of the current point (the *incumbent*), by moving one step in each direction on an iteration-dependent mesh (POLL stage). The typical step size at iteration k is given by Δ_k^{poll}, whereas Δ_k^{mesh} represents the mesh resolution. In addition, the MADS framework can incorporate in the optimization any arbitrary exploration strategy which proposes additional test points that lie on the mesh (SEARCH stage). In BADS, we exploit the freedom of SEARCH to perform Bayesian optimization in the neighborhood of the incumbent. See [12, 1] for details.

Bayesian optimization The standard form of BO [2] builds a GP [14] approximation of the objective f, which is used as a relatively inexpensive surrogate to guide the search towards regions that are promising (low GP mean) and/or unknown (high GP uncertainty), according to an *acquisition function* that formalizes the exploitation-exploration trade-off. See [3, 4] for a tutorial on BO.

2.1 Algorithm overview

Initialization The algorithm is initialized by providing a starting point $\boldsymbol{x}_0$, *hard* lower/upper bounds `LB`, `UB`, and optional *plausible* lower/upper bounds `PLB`, `PUB`. Plausible bounds identify a smaller region in parameter space where most solutions are expected to lie. During optimization, variables are linearly rescaled to the standardized box $[-1, 1]^D$ such that the box bounds correspond to $[\texttt{PLB}, \texttt{PUB}]$ in the original space. BADS supports bound or no constraints, and optionally other known constraints via a provided *barrier* function $c(\boldsymbol{x})$ that returns constraint violations. The initial design consists of $\boldsymbol{x}_0$ plus $n_{\text{init}} = D$ points chosen via a space-filling Sobol sequence [15] in the standardized box.

GP model in BADS The GP model is specified by a constant mean function $m \in \mathbb{R}$, a smooth automatic relevance determination (ARD) *rational quadratic* (RQ) kernel, and we use the GP *lower confidence bound* (a_{LCB}) [16] as acquisition function. The GP training set consists of a subset of the points evaluated so far, selected to build a *local* approximation of the objective, and contains up to $50 + 10D$ points in the neighborhood of the incumbent $\boldsymbol{x}_k$. Newly evaluated points are added incrementally to the set, using fast rank-one updates of the GP posterior. The training set is rebuilt any time the incumbent is moved. We impose an empirical Bayes prior on the GP hyperparameters based on the current GP training set, and select $\boldsymbol{\theta}$ via maximum a posteriori (MAP) estimation. We refit the hyperparameters every $2D$ to $5D$ function evaluations; more often earlier in the optimization, and whenever the current GP is particularly inaccurate at predicting new points.

[2] Code available at `https://github.com/lacerbi/bads`.

Algorithm 1 Bayesian Adaptive Direct Search

Input: objective function f, starting point $\boldsymbol{x}_\mathbf{0}$, hard bounds `LB`, `UB`, (*optional*: plausible bounds `PLB`, `PUB`, barrier function c, additional `options`)

Initialization: $\Delta_0^{\text{mesh}} \leftarrow 2^{-10}$, $\Delta_0^{\text{poll}} \leftarrow 1$, $k \leftarrow 0$, evaluate f on initial design
repeat
  (update GP approximation at any step; refit hyperparameters if necessary)
  for $1 \dots n_{\text{search}}$ **do** ▷ SEARCH stage
    $\boldsymbol{x}_{\text{search}} \leftarrow$ SEARCHORACLE ▷ local Bayesian optimization step
    Evaluate f on $\boldsymbol{x}_{\text{search}}$, **if** improvement is *sufficient* **then break**
  if SEARCH is NOT *successful* **then** ▷ optional POLL stage
    evaluate opportunistically f on the POLL set sorted by acquisition function
  if iteration k is *successful* **then**
    update incumbent $\boldsymbol{x}_{k+1}$
    if POLL was *successful* **then** $\Delta_k^{\text{mesh}} \leftarrow 2\Delta_k^{\text{mesh}}$, $\Delta_k^{\text{poll}} \leftarrow 2\Delta_k^{\text{poll}}$
  else
    $\Delta_k^{\text{mesh}} \leftarrow \frac{1}{2}\Delta_k^{\text{mesh}}$, $\Delta_k^{\text{poll}} \leftarrow \frac{1}{2}\Delta_k^{\text{poll}}$
  $k \leftarrow k + 1$
until `fevals` $>$ `MaxFunEvals` **or** $\Delta_k^{\text{poll}} < 10^{-6}$ **or** stalling ▷ stopping criteria
return $\boldsymbol{x}_{\text{end}} = \arg\min_k f(\boldsymbol{x}_k)$ (or $\boldsymbol{x}_{\text{end}} = \arg\min_k q_\beta(\boldsymbol{x}_k)$ for noisy objectives, see [1])

Implementation of the MADS framework We adopt an aggressive, repeated SEARCH strategy that consists of up to $n_{\text{search}} = \max\{D, \lfloor 3 + D/2 \rfloor\}$ unsuccessful SEARCH steps. In each step, we use a *search oracle*, based on a local BO with the current GP, to produce a search point $\boldsymbol{x}_{\text{search}}$. We choose $\boldsymbol{x}_{\text{search}}$ via a fast, approximate evolutionary optimization inspired by CMA-ES [17] (see [1] for details). We evaluate $f(\boldsymbol{x}_{\text{search}})$ and add it to the training set. If the improvement in objective value is none or *insufficient*, that is less than $(\Delta_k^{\text{poll}})^{3/2}$, we continue searching, or switch to POLL after n_{search} steps. Otherwise, we call it a *success* and start a new SEARCH from scratch, centered on the updated incumbent. We incorporate the GP approximation in the POLL in two ways: when constructing the set of polling directions, by rescaling them proportionally to the GP length scales, and when choosing the polling order, by polling vectors according to the ranking given by the acquisition function [8]. In case of a noisy objective, we adjust several algorithm parameters for increased robustness (see [1]).

3 Experiments

We tested BADS and 16 other optimizers for MATLAB (R2015b, R2017a) on a large set of artificial and real optimization problems (see [1] for details). In particular, to verify the advantage of BADS' hybrid approach to BO, we also tested a standard, 'vanilla' version of BO (`bayesopt`, R2017a; similar to [5] but without hyperparameter marginalization). All algorithms used default settings.

Problem sets First, we considered a standard benchmark set of artificial, noiseless functions (BBOB09 [18], 24 functions) in dimensions $D \in \{3, 6, 10, 15\}$, for a total of 96 test functions. We also created 'noisy' versions of the same set. On this benchmark, BADS performed better or on par with state-of-the-art methods (data not shown; see [1]). Second, we collected model-fitting problems from six studies in cognitive and computational neuroscience (CCN17; see [1]). The objectives here are negative log likelihood functions of an input parameter vector, for specified datasets and models. For each study, we asked its authors for six real datasets (i.e., subjects or neurons), divided between one or two main models of interest; collecting a total of 36 test functions with $D \in \{6, 9, 10, 12, 13\}$.

Procedure We ran 50 independent runs of each algorithm on each test function, with randomized starting points and a budget of $500D$ function evaluations ($200D$ for noisy problems). If an algorithm terminated before depleting the budget, it was restarted from a new random point. We consider a run *successful* if the current best (or returned, for noisy problems) function value is within a given *error tolerance* $\varepsilon > 0$ from the true optimum f_{min} (or our best estimate thereof). For noiseless problems, we compute the fraction of successful runs as a function of number of objective evaluations, averaged over datasets/functions and over $\varepsilon \in [0.01, 10]$ (log spaced). This is a realistic range of ε, for model fitting purposes. For noisy problems, what matters most is the solution $\boldsymbol{x}_{\text{end}}$ that the algorithm *actually* returns at the end of the optimization. Thus, we plot the fraction of successful runs at $200D$ function evaluations as a function of ε, for $\varepsilon \in [0.1, 10]$, and averaged over datasets/functions.

Results (CCN17) The objectives are deterministic (e.g., computed via numerical approximation) for three studies (Fig 1), and noisy (e.g., evaluated via simulation) for the other three (Fig 2).

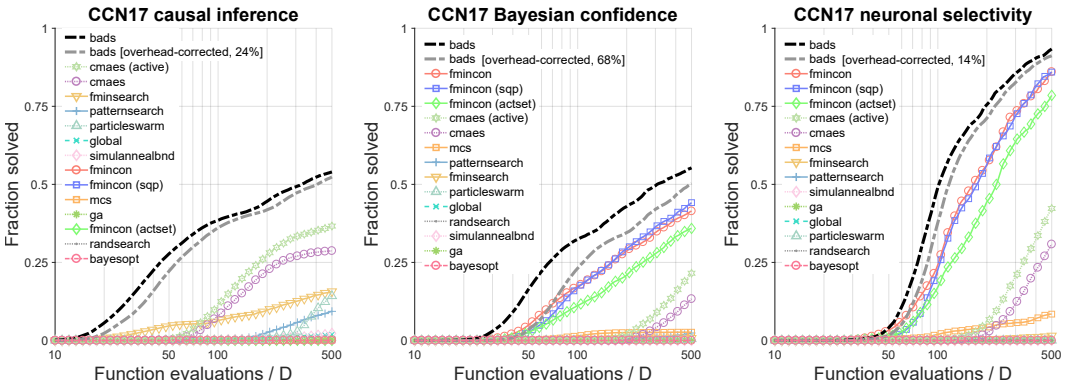

Figure 1: **Real model-fitting problems (CCN17, deterministic).** Fraction of successful runs ($\varepsilon \in [0.01, 10]$) vs. # function evaluations per # dimensions. *Left*: Causal inference in visuo-vestibular perception [19] (6 subjects, $D = 10$). *Middle*: Bayesian confidence in perceptual categorization [20] (6 subjects, $D = 13$). *Right*: Neural model of orientation selectivity [21] (6 neurons, $D = 12$).

In all problems, BADS consistently performs on par with or outperforms all other tested optimizers, even when accounting for its extra algorithmic cost of $\sim$ 0.03-0.15 s per function evaluation, depending on D (see entry for overhead-corrected BADS in Fig 1). Interestingly, vanilla BO (`bayesopt`) performs poorly on all problems, even without accounting for the much larger overhead of `bayesopt` (on average $\sim$ 8 s per function evaluation, with up to 300 training inputs). More complex forms of BO (e.g., input warping [25], hyperparameter marginalization [5]) might fare better, but would substantially increase the already large overhead. Importantly, we expect this poor perfomance to extend to any package which implements vanilla BO (such as *BayesOpt* [26]).

In conclusion, we have developed a novel BO method and an associated toolbox, BADS, with the goal of fitting mildly expensive computational models, such as those found in many scientific fields, out-of-the-box. On real model-fitting problems, BADS outperforms widely used and state-of-the-art methods for nonconvex, derivative-free optimization, including 'vanilla' BO. We attribute the robust performance of BADS to the alternation between the aggressive, efficient SEARCH strategy, based on local BO, and the failsafe, almost model-free, POLL stage, which protects against failures of the GP surrogate – whereas vanilla BO is vulnerable to model misspecification. The performance of BADS is linked to its ability to obtain a fast approximation of the objective, which generally deteriorates in high dimensions, or for functions with pathological structure. Major directions of future work include extending BADS to problems with higher dimensionality; testing alternative statistical surrogates instead of GPs; and recasting some of its heuristics in terms of approximate inference.

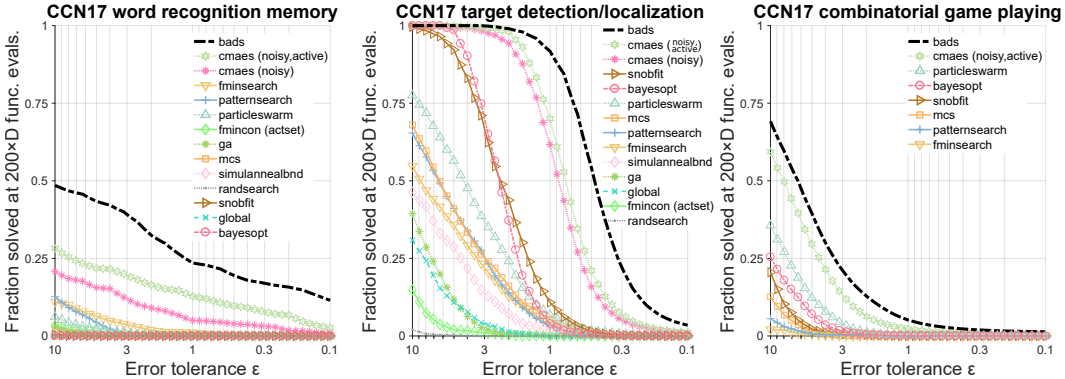

Figure 2: **Real model-fitting problems (CCN17, noisy).** Fraction of successful runs at $200 \times D$ objective evaluations vs. tolerance ε. *Left*: Confidence in word recognition memory [22] (6 subjects, $D = 6, 9$). *Middle*: Target detection and localization [23] (6 subjects, $D = 6$). *Right*: Combinatorial board game playing [24] (6 subjects, $D = 10$).

References

[1] 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.

[2] Jones, D. R., Schonlau, M., & Welch, W. J. (1998) Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* **13**, 455–492.

[3] 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*.

[4] 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**, 148–175.

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

[6] Taddy, M. A., Lee, H. K., Gray, G. A., & Griffin, J. D. (2009) Bayesian guided pattern search for robust local optimization. *Technometrics* **51**, 389–401.

[7] Picheny, V. & Ginsbourger, D. (2014) Noisy kriging-based optimization methods: A unified implementation within the DiceOptim package. *Computational Statistics & Data Analysis* **71**, 1035–1053.

[8] Gramacy, R. B. & Le Digabel, S. (2015) The mesh adaptive direct search algorithm with treed Gaussian process surrogates. *Pacific Journal of Optimization* **11**, 419–447.

[9] Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011) Sequential model-based optimization for general algorithm configuration. *LION* **5**, 507–523.

[10] Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011) *Algorithms for hyper-parameter optimization*. pp. 2546–2554.

[11] Talgorn, B., Le Digabel, S., & Kokkolaras, M. (2015) Statistical surrogate formulations for simulation-based design optimization. *Journal of Mechanical Design* **137**, 021405–1–021405–18.

[12] Audet, C. & Dennis Jr, J. E. (2006) Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization* **17**, 188–217.

[13] Audet, C., Custódio, A., & Dennis Jr, J. E. (2008) Erratum: Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization* **18**, 1501–1503.

[14] Rasmussen, C. & Williams, C. K. I. (2006) *Gaussian Processes for Machine Learning*. (MIT Press).

[15] Bratley, P. & Fox, B. L. (1988) Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)* **14**, 88–100.

[16] Srinivas, N., Krause, A., Seeger, M., & Kakade, S. M. (2010) Gaussian process optimization in the bandit setting: No regret and experimental design. *ICML-10* pp. 1015–1022.

[17] Hansen, N., Müller, S. D., & Koumoutsakos, P. (2003) Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* **11**, 1–18.

[18] Hansen, N., Finck, S., Ros, R., & Auger, A. (2009) Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions.

[19] Acerbi, L., Dokka, K., Angelaki, D. E., & Ma, W. J. (2017) Bayesian comparison of explicit and implicit causal inference strategies in multisensory heading perception. *bioRxiv preprint bioRxiv:150052*.

[20] Adler, W. T. & Ma, W. J. (2017) Human confidence reports account for sensory uncertainty but in a non-Bayesian way. *bioRxiv preprint bioRxiv:093203*.

[21] Goris, R. L., Simoncelli, E. P., & Movshon, J. A. (2015) Origin and function of tuning diversity in macaque visual cortex. *Neuron* **88**, 819–831.

[22] van den Berg, R., Yoo, A. H., & Ma, W. J. (2017) Fechner's law in metacognition: A quantitative model of visual working memory confidence. *Psychological Review* **124**, 197–214.

[23] Mazyar, H., van den Berg, R., & Ma, W. J. (2012) Does precision decrease with set size? *J Vis* **12**, 1–10.

[24] van Opheusden, B., Bnaya, Z., Galbiati, G., & Ma, W. J. (2016) Do people think like computers? *International Conference on Computers and Games* pp. 212–224.

[25] Snoek, J., Swersky, K., Zemel, R., & Adams, R. (2014) *Input warping for Bayesian optimization of non-stationary functions*. pp. 1674–1682.

[26] Martinez-Cantin, R. (2014) BayesOpt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research* **15**, 3735–3739.