Do we need "Harmless" Bayesian Optimization and "First-Order" Bayesian Optimization?

Mohamed Osama Ahmed
University of British Columbia

Bobak Shahriari
University of British Columbia

Mark Schmidt
University of British Columbia

Abstract

A recent empirical study highlighted the shocking result that, for many hyperparameter tuning problems, Bayesian optimization methods can be outperformed by random guessing run for twice as many iterations [1]. This is supported by theoretical results showing the optimality of random search under certain assumptions, but disagrees with other theoretical and empirical results showing that Bayesian optimization can lead to large gains in some situations. In light of this fact, we propose two research directions that we believe the community should pursue. First, we should focus on developing "harmless" Bayesian optimization methods that do no worse than random, and we propose a very simple "harmless" algorithm. Second, we should focus on developing first-order Bayesian optimization algorithms that use gradient information to improve performance for situations where Bayesian optimization already beats random. We empirically show the advantage of both of these ideas in simple simulations. We also propose a simple strategy for reducing the memory and computational requirements of existing first-order Bayesian optimization methods by using directional derivatives instead of full gradients, which can be obtained from analytic functions even when gradient code is not available.

1 Introduction

Bayesian optimization (BO) has a long history and has been used in variety of fields (see [2] for a recent review), with recent interest from the machine learning community in the context of automatic hyperparameter tuning [3]. However, the empirical results of a recent work have called into question the usefulness of Bayesian optimization; Li et al. [1] show on a large number of problems that Bayesian optimization often only gave a small gain over random search and in fact was typically outperformed by running random search for twice as many iterations. On the other hand, for some specific problems BO does in fact handily beat random (see our experiments) and we know that under certain smoothness assumptions BO can be exponentially faster than random [4, Theorem 5].

In the context of these conflicting results, in this work we propose two strategies for improving the performance of Bayesian optimization methods. First, in the next section we introduce the concept of a "harmless" Bayesian optimization algorithm. This is an algorithm that does no worse than random in the worst case, but that might still take advantage of BO for functions with a high degree of smoothness. Second, since smooth functions are one class of function where BO has a large advantage over random, we propose to re-explore the idea of "first-order" Bayesian optimization (FOBO) methods that use gradient information to improve performance. We show how gradient information can lead to improved performance, and also propose FOBO strategies that use directional derivatives. The advantage of using directional derivatives is that they reduce the cost and memory

requirements of FOBO methods, while directional derivatives can be obtained by automatic differentiation for any analytic function (so can be used even in many "black box" situations where we don't have gradient code available).

2 Harmless Bayesian Optimization

We consider the problem of minimizing a real-valued function f over a domain $\mathcal{X}$ consisting of finite lower and upper bounds on every variable,

$$\underset{x \in \mathcal{X}}{\operatorname{argmin}} f(x). \tag{1}$$

We first consider a zero-order oracle where on iteration t the algorithm can learn about the function by choosing a parameter value x^t and receiving the corresponding function value $f(x^t)$. In this model, we consider the goal of trying to minimize the number of iterations t before we can guarantee that we find a parameter value $\hat{x}$ such that $f(\hat{x}) - f^* \leq \epsilon$ (where f^* is the minimum of f and ϵ is some positive constant). In this work we will primarily be interested in BO methods based on Gaussian processes and will consider deterministic functions, but we expect that these observations also hold in other settings.

Firstly, we note that finding such an $\hat{x}$ is *impossible in any finite number of oracle calls* unless we make assumptions about f. Because the real numbers are uncountable, for any deterministic algorithms we can construct a function that achieves an arbitrarily low function value at a parameter setting that the algorithm will never try (and similar arguments hold for stochastic algorithms). Thus, to say anything in this setting we need some sort of continuity assumptions about the function f.

One of the most natural assumption about f that we might make is that f is Lipschitz-continuous, meaning that the amount that f can change as we change x is bounded by a constant times the change in x. Under this assumption, it is known that *any* algorithm requires $\Omega(1/\epsilon^d)$ iterations to reach an accuracy of ϵ [5, Theorem 1.1.2]. Further, an $O(1/\epsilon^d)$ worst-case rate can be achieved by using a grid-based search [5, Corollary 1.1.1] or in expectation simply by trying random values for the x^t (because the probability that a random x is an ϵ-optimal solution under this asumption is $\Omega(\epsilon^d)$). Thus, in some sense random search is an *optimal* algorithm. Further, this $O(1/\epsilon^d)$ rate is faster than the best known rates for BO in many settings, in the sense that the exponent of ϵ for BO can be larger than d [4, Theorems 1-4].[1]

However, for functions with a sufficient amount of smoothness BO can beat random search. In particular, under additional assumptions and using ν as a measure of smoothness of f, Bull shows in a fascinating result that a variant of BO only requires $\tilde{O}(1/\epsilon^{d/\nu})$ iterations [4, Theorem 5]. Thus, for "nice" functions where ν is greater than 1 BO can be exponentially faster than random search. This gives support for the empirical observation that in some settings BO soundly beats random search.

Unfortunately, in the black-box setting we typically don't know ν and so we don't know if it is greater or less than 1. This motivates us to consider "harmless" BO:

- A "harmless" BO algorithm is a BO method that requires at most $O(1/\epsilon^d)$ iterations to achieve an accuracy of ϵ on a Lipschitz-continuous function, and thus up to a constant factor performs as well as random search in the worst case.

It is quite simple to achieve a "harmless" BO method by combining an existing BO method with an existing "harmless" method like random search. We simply alternate between performing the iterations of the two algorithms. For example, if on odd iterations we evaluate a random x^t and on even iterations we apply a BO method, this would constitute a "harmless" BO method. This is actually quite similar to the "ϵ-greedy" approach that Bull requires just for convergence of BO (and which is a common way to address exploration-exploitation issues). Further, if our BO method is the method analyzed by Bull then this simple harmless BO method achieves the best known rate of $\tilde{O}(1/\epsilon^{\min\{d, d/\nu\}})$. This assumes that the BO method ignores the random search iterations, but in practice we would likely do better by giving the BO method access to these iterates. Indeed,

[1] We state results in terms in terms of the number of needed iterations, but we could equivalently state results in terms of the error after a fixed number of iterations. For example, if we require $O(1/\epsilon^d)$ iterations to reach an accuracy of ϵ then we can guarantee a sub-optimality of $O(1/t^{1/d})$ after t iterations.

we conjecture that better harmless methods are possible which *locally* try to exploit smoothness in certain regions by adapting to the properties of the black-box function. Similarly, it might be possible to improve the method by biasing the random samples away from the points sampled by the GP.

3 First Order Bayesian Optimization

Making a BO method harmless protects it from obtaining poor performance on functions with a low degree of smoothness, but often we are faced with minimizing a function with a high degree of smoothness. In this differentiable case we should expect to improve performance by incorporating knowledge of the derivatives at the guesses x^t. We call methods based on this extra information first-order Bayesian optimization (FOBO) methods.

Incorporating derivative observations in Gaussian processes is not a new idea [6–9], and there has been preliminary work on using derivative information specifically for the purpose of Bayesian optimization on two-dimensional problems [10, Section 5.2.4]. In this section we revisit this idea in higher dimensions and also explore how directional derivatives can reduce the time/memory as well as the need to write gradient code when all that we have available is a zero-order black box.

The usual BO assumption is that the values of f are jointly Gaussian, being generated by a Gaussian process. We can incorporate derivative information by also assuming that derivatives are being generated by a Gaussian process. In particular, we'll assume that the function values and all first derivatives of f are jointly Gaussian and that the covariance kernel is twice differentiable. In this setting, the additional elements of the covariance kernel involving partial derivatives are given by [11, 12]

$$\text{cov}(f(x^i), \partial_p f(x^j)) = \partial_p k(x^i, x^j), \text{ and} \tag{2}$$

$$\text{cov}(\partial_p f(x^i), \partial_q f(x^j)) = \partial_p \partial_q k(x^i, x^j), \tag{3}$$

where $\partial_p f$ denotes the partial derivative of f with respect to direction p. Therefore, as long as the kernel is twice differentiable and the gradient can be evaluated, the GP can be extended to include gradient observations. Further, we conjecture that including gradient observations can further reduce the exponent in the iteration complexity of BO methods.

While the cost of computing gradients for analytic functions cannot be more than a constant factor more expensive than the computing the function value, the memory and time requirement of the GP model increase if we use the full gradient of each x^t. In particular, the memory is increased from $O(t^2)$ to $O(t^2d^2)$ when we have d variables. Similarly, the cost of the GP is increased from $O(t^3)$ to $O(t^3d^3)$ (assuming we use an exact solver, for illustration purposes). If this extra memory requirement is too large, then we propose to instead of modelling the gradient as a GP to instead model a *directional derivative* $\partial_p f(x^t)$ for a particular direction p. The most logical direction p to consider is the gradient direction, $\nabla f(x^t)$. This directional derivative still provides information about how the function changes, but since it is a scalar it only increases the time/memory by a constant factor.

In many scenarios where BO is applied it is possible to compute gradients. For example, Bengio as well as Maclaurin et al. have shown how to compute gradients with respect to hyper-parameters of machine learning models [13, 14]. It is likely that these existing methods could be improved through the use of FOBO. On the other hand, sometimes we really have a "black box" and cannot get access to the gradient. In these settings, as long as the function is analytic it is always possible to use automatic differentiation to obtain directional derivatives $\partial_p f(x^t)$ for a given direction p for a small increase in the cost of evaluating the black box. A particularly nice way of doing this is with the complex-step trick [15]. Our experiments indicate that simply setting p to a random direction can be more effective than standard BO methods for smooth functions.

4 Simulation Results

We performed a small set of numerical experiments on the following two classes of functions to illustrate the two ideas explored in this paper:

1. We generated datasets by applying a kernel smoother to examples generated uniformly across a 10-dimensional domain where the function values were generated from a student

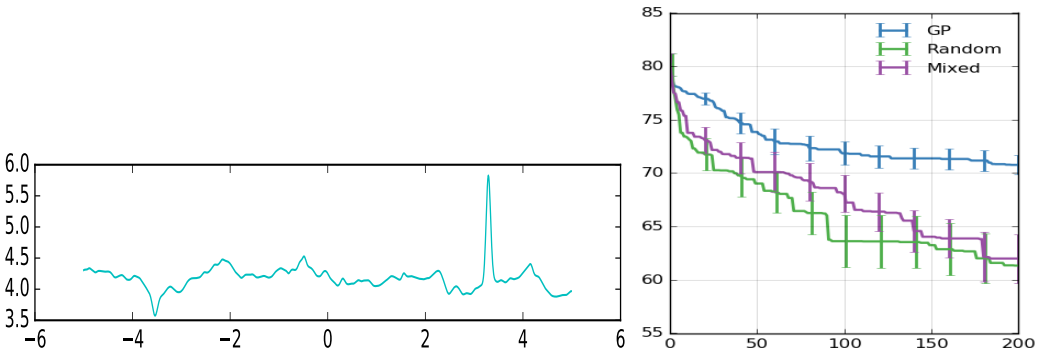

Figure 1: *Example of a 1-D oscillatory function used to test the harmless strategy by applying a kernel smoother to student t data (left) and performance of different algorithms on a 10-dimensional variant.*

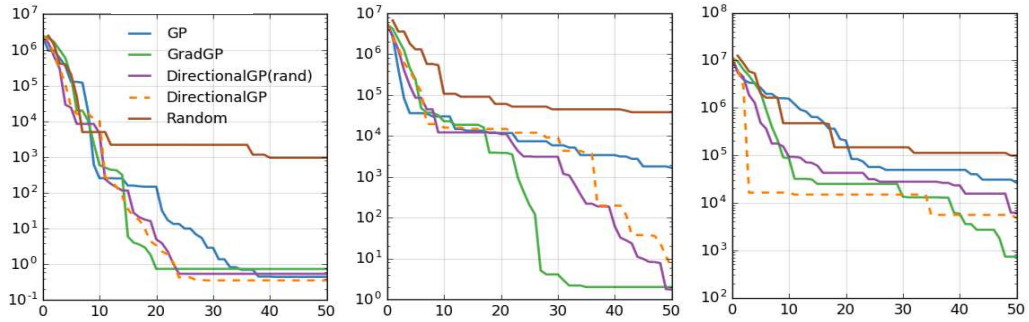

Figure 2: *Sub-optimality against number of iterations for different global optimization strategies for the Rosenbrock function in 2D(left), 3D (center) and 4D (right).*

t-distribution. This yields a function that is differentiable, but where the smoothness assumptions required for BO to outperform random are not satisfied.

2. To examine the effects of incorporating gradient information on a (relatively) "nice" function, we used the Rosenbrock function $f(x_1 \cdots x_d) = \sum_{i=1}^{d-1}(100(x_i^2 - x_{i+1})^2 + (1-x_i)^2)$ for different dimensions d.

We used the squared exponential covariance function, set the mean function to be constant or a quadratic, and used the expected improvement acquisition function. Our experiments consider the difference between the true maximum and the best value found by the algorithm.

Our first experiment in Figure 1 plots the performance on the student t of BO (GP) against random search (Random) and the "harmless" BO algorithm from Section 2 that combines these. On this dataset the random method significantly outperforms the non-harmless BO method. However, the harmless strategy allows the GP model to obtain similar performance to random.

Our second experiment in Figure 2 compares GP and Random to different variants of FOBO on the Rosenbrock function. In particular, we compared to FOBO methods that use the full gradient information (GradGP), use the directional derivative in the gradient direction (DirectionalGP), or that use a random directional derivative (DirectionalGP(random). On these smooth functions, random search is clearly beaten by the BO methods. But note that the standard BO method is clearly beat by many of the FOBO methods. In particular, using the full gradient information performed close to the best in all experiments but in all cases even the directional derivative information offered an advantage.

References

[1] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Efficient hyperparameter optimization and infinitely many armed bandits. *arXiv preprint arXiv:1603.06560*, 2016.

[2] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[3] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 2012.

[4] Adam D Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct):2879–2904, 2011.

[5] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[6] Max D Morris, Toby J Mitchell, and Donald Ylvisaker. Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

[7] Ercan Solak, Roderick Murray-Smith, William E Leithead, Douglas J Leith, and Carl Edward Rasmussen. Derivative observations in gaussian process models of dynamic systems. *Advances in Neural Information Processing Systems*, 2003.

[8] Carl Edward Rasmussen and Christopher Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

[9] Michael Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature*. PhD thesis, University of Oxford, 2010.

[10] Daniel Lizotte. *Practical Bayesian optimization*. PhD thesis, University of Alberta, 2008.

[11] Athanasios Papoulis and S Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[12] Robert J Adler. *The geometry of random fields*, volume 62. Siam, 2010.

[13] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

[14] Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Gradient-based hyperparameter optimization through reversible learning. *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[15] Joaquim RRA Martins, Peter Sturdza, and Juan J Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.